Reference Resolution in Context

Jan van Eijck

CWI and ILLC, Amsterdam, Uil-OTS, Utrecht

Abstract

This paper sketches an approach to pronoun reference resolution in context based on a dynamic incremental semantics for NL in polymorphic type theory. Our set-up provides full incrementality of processing, and can handle salience and pronoun resolution in context. An implementation of the system in Haskell, in 'literate programming' style, exists. The full literate source code can be found at http://www.cwi.nl/ jve/papers/02/rric.

1 Salience

An important aspect of natural language text understanding is the resolution of anaphoric links: finding out what the pronouns that occur in the text refer to. Anaphoric reference resolution is incremental in the sense that, in general, the information that is needed to determine the reference of a pronoun in a text can be determined 'on the go', on the basis of:

- Syntactic properties of the sentence that contains the pronoun.
- Information conveyed in the previous discourse (including the introduction of salient referents, with salience determined in part by syntax).
- Background information shared by speaker and hearer in an act of communication (the common ground of speaker and hearer).

Surface syntactic form is an important determinant for salience. It is usually assumed, e.g., that subject is more salient than object. In *John hit Bill. He was upset.*, the first choice for resolving the pronoun *He* is *John*, because *John* is the subject of the preceding sentence. In *Bill got kicked by John. He was upset*, the first choice for resolving *He* is *Bill*, because *Bill* is the subject of the preceding sentence. In both cases, *John* and *Bill* are the two obvious candidates for resolving the reference of the pronoun *He*, because both *John* and *Bill* have been made salient by the preceding text.

There are many ways to increase one's understanding about reference resolution in context. One method, the method to be used in this paper, is to focus on the simplest possible set-up, and to construct a formal picture of what is needed for the process of reference resolution. Just as in logic one can study reasoning by constructing and analyzing reasoning systems, so we can study reference resolution by constructing and analyzing systems for reference resolution. The set-up we focus on is a simplification in many ways, but this should be considered as a virtue rather than a defect. A respectable way to proceed in the formal sciences is to first form a clear picture of an idealized reality, and then gradually proceed to take more factors into account.

We will explore the following idea. Discourse processing is a matter of construction of a context, where a context is simply a list of entities introduced in the (idealized) act of processing the preceeding discourse. The ordering of the context list determines the salience of the discourse objects. As the discourse continues, new salient objects can get introduced, old objects can again become salient, and objects can drift out of focus. The mechanism we propose for context updating is elimination of possibilities plus salience updating of remaining contexts. Contexts that were compatible with the information conveyed up until now may drop out as no longer possible in the light of the new information. Contexts still compatible with the new information may get extended, and the salience ordering of their elements may get adjusted.

The structure of the rest of this paper is as follows. In the next two sections, we introduce contexts and context manipulation by example. Then, we introduce information states, and sketch the basic set-up of meaning representation in context semantics. Some technical issues concerning the property of being a test and the associativity of composition are dealt with. Next we use polymorphic type theory to arrive at a Montague style version of context semantics. Finally, we sketch the reference resolution engine, discuss some examples, and comment on the limitations of our framework.

2 Contexts

The contexts we consider are finite lists of objects mentioned in discourse. A context may consist of princess Maxima and prime minister Wim Kok, in that order.

$$\begin{array}{|c|c|}\hline \text{Maxima} & \text{Kok} \\ \hline\end{array}$$

Context extension is the process of adding items to the context list, e.g., by adding prince Willem Alexander to the context consisting of princess Maxima and prime minister Kok.

$$\begin{array}{|c|c|c|}\hline \text{W-Alex} & \text{Maxima} & \text{Kok} \\ \hline\end{array}$$

Salience update in context is a reshuffle of the order of importance of the items in a context list. Salience update of the example context may make princess Maxima the most salient item.

$$\begin{array}{|c|c|c|}\hline \text{Maxima} & \text{Kok} & \text{W-Alex} \\ \hline\end{array}$$

To allow reshuffling of a context with princess Maxima in it, in such a way that we do not loose track of her, we represent contexts as lists of indexed objects, with the indices running from 0 to the length of the context minus 1:

$$\begin{array}{|c|c|c|}\multicolumn{1}{c}{\scriptstyle 0} & \multicolumn{1}{c}{\scriptstyle 1} & \multicolumn{1}{c}{\scriptstyle 2} \\ \hline \text{Maxima} & \text{Kok} & \text{W-Alex} \\ \hline\end{array}$$

Reshuffle this to make Willem Alexander most salient:

$$\overset{2}{\boxed{\text{W-Alex}}}\overset{0}{\boxed{\text{Maxima}}}\overset{1}{\boxed{\text{Kok}}}$$

Note that the indices $0, \dots, n-1$ determine a permutation of the context list. Call these lists of indexed objects contexts under permutation.

3 Context Manipulation

In a context c, the entity with index i is given by $c[*i]$. E.g.,

$$\left(\overset{2}{\boxed{\text{W-Alex}}}\overset{0}{\boxed{\text{Maxima}}}\overset{1}{\boxed{\text{Kok}}}\right)[*0] = \text{Maxima}$$

If c is a context under permutation, let $(i)c$ be the result of placing the item $(i, c[*i])$ upfront. E.g.,

$$(1)\left(\overset{2}{\boxed{\text{W-Alex}}}\overset{0}{\boxed{\text{Maxima}}}\overset{1}{\boxed{\text{Kok}}}\right) = \overset{1}{\boxed{\text{Kok}}}\overset{2}{\boxed{\text{W-Alex}}}\overset{0}{\boxed{\text{Maxima}}}$$

$$(0)\left(\overset{2}{\boxed{\text{W-Alex}}}\overset{0}{\boxed{\text{Maxima}}}\overset{1}{\boxed{\text{Kok}}}\right) = \overset{0}{\boxed{\text{Maxima}}}\overset{2}{\boxed{\text{W-Alex}}}\overset{1}{\boxed{\text{Kok}}}$$

Thus, $(i)c$ is the result of moving the item with index i to the head position of the context list. It is not hard to see that successive applications of this operation can generate all permutations of a context. A permutation of a finite list can always be decomposed into a finite number of successive swaps in the list.

If d is an object and c a context, then $d : c$ is the result of putting item $(|c|, d)$ at the head position of the context list. Here $|c|$ refers to the size of the context. Thus,

$$\text{Beatrix} : \left(\overset{2}{\boxed{\text{W-Alex}}}\overset{0}{\boxed{\text{Maxima}}}\overset{1}{\boxed{\text{Kok}}}\right) = \overset{3}{\boxed{\text{Beatrix}}}\overset{1}{\boxed{\text{Kok}}}\overset{2}{\boxed{\text{W-Alex}}}\overset{0}{\boxed{\text{Maxima}}}$$

The operation $(:)$ is used for adding a new element to the context, in most salient position.

Finally, we need a way of cutting context down do size: $\lceil i \rceil\ c$ specifies the result of removing all items with an index $\geq i$ from the context c. E.g.,

$$\lceil 2 \rceil \left(\overset{3}{\boxed{\text{Beatrix}}}\overset{1}{\boxed{\text{Kok}}}\overset{2}{\boxed{\text{W-Alex}}}\overset{0}{\boxed{\text{Maxima}}}\right) = \overset{1}{\boxed{\text{Kok}}}\overset{0}{\boxed{\text{Maxima}}}$$

Note that $\lceil i \rceil\ c$ produces a context of length i.

4 Information States

Let an indexed context be a pair consisting of a world index and a context. Let an information state be a set of indexed contexts, all of the same length, all under the same permutation.

$\Downarrow$	2	0	3	1
w_0	Beatrix	Kok	Claus	Claus
w_0	Maxima	W-Alex	Kok	Bernhard
w_1	Beatrix	Kok	Bernhard	Claus
w_1	Emily	W-Alex	Kok	Claus

Lift the operation $\lceil i \rceil$ to information states S by means of

$$\lceil i \rceil\ S := \{\lceil i \rceil\ c \mid c \in S\}.$$

This gives:

$\lceil 2 \rceil$

$\Downarrow$	2	0	3	1
w_0	Beatrix	Kok	Claus	Claus
w_0	Maxima	W-Alex	Kok	Bernhard
w_1	Beatrix	Kok	Bernhard	Claus
w_1	Emily	W-Alex	Kok	Claus

$=$

$\Downarrow$	0	1
w_0	Kok	Claus
w_0	W-Alex	Bernhard
w_1	Kok	Claus
w_1	W-Alex	Claus

5 Meaning Representation in Context Semantics

Incremental semantics for natural language (Eijck 2001) is based on two ideas:

- replacement of destructive assignment by context extension,
- use of polymorphic type theory to deal with contexts of arbitrary lengths in a uniform way.

The basic (untyped) system of incremental dynamics can be viewed as the one-variable version of the sequence semantics for dynamic predicate logic proposed in (Vermeulen 1993). The definition below extends the system given in (Eijck 2001) with a mechanism for salience ordering.

In standard dynamic semantics, a test on a context or state leaves the context or state unaffected when it succeeds. We now have to modify this to allow for salience reshuffles. For that, we need to define positive and negative relational meaning of the basic context logic.

In possible world semantics, information growth is modelled by elimination of possibilities. Ignorance concerning a fact p is shows up by the presence of worlds where p is true and worlds where p is false in an information state. Learning that p is false boils down to elimination of the worlds where p is true from the information state. We will let contexts be world-dependent by providing every context with a world index c_w. We will assume that all worlds share the same domain D.

Assume i, j to be indices with $i, j < |c|$. Assume $c_w = M$. Then $c \models Pi$, $c \rightmodels Pi$, $c \models Pij$ and $c \rightmodels Pij$ are given by:

$$\begin{array}{rcl} c \models Pi & \Leftrightarrow & c[*i] \in P^M, \\ c \rightmodels Pi & \Leftrightarrow & c[*i] \notin P^M, \\ c \models Pij & \Leftrightarrow & \langle c[*i], c[*j] \rangle \in P^M, \\ c \rightmodels Pij & \Leftrightarrow & \langle c[*i], c[*j] \rangle \notin P^M. \end{array}$$

Using this, we can define positive and negative updates of contexts, as follows:

$$\begin{array}{rcl} [\![\exists]\!]^+(c) & := & \{d : c \mid d \in D\} \\ [\![\exists]\!]^-(c) & := & \emptyset \\ [\![Pi]\!]^+(c) & := & d \begin{cases} \{(i)c\} & \text{if } c \models Pi \\ \emptyset & \text{if } c \rightmodels Pi \end{cases} \\ [\![Pi]\!]^-(c) & := & \begin{cases} \{(i)c\} & \text{if } c \rightmodels Pi \\ \emptyset & \text{if } c \models Pi \end{cases} \\ [\![Pij]\!]^+(c) & := & \begin{cases} \{(i)(j)c\} & \text{if } c \models Pij \\ \emptyset & \text{if } c \rightmodels Pij \end{cases} \\ [\![Pij]\!]^-(c) & := & \begin{cases} \{(i)(j)c\} & \text{if } c \rightmodels Pij \\ \emptyset & \text{if } c \models Pij \end{cases} \\ [\![\neg\phi]\!]^+(c) & := & [\![\phi]\!]^-(c) \\ [\![\neg\phi]\!]^-(c) & := & \lceil |c| \rceil \; [\![\phi]\!]^+(c) \\ [\![\phi;\psi]\!]^+(c) & := & \bigcup\{[\![\psi]\!]^+(c') \mid c' \in [\![\phi]\!]^+(c)\} \\ [\![\phi;\psi]\!]^-(c) & := & \begin{cases} \emptyset \text{ if } \exists c' \in [\![\phi]\!]^+(c) \text{ with } [\![\psi]\!]^+(c') \neq \emptyset \\ [\![\phi]\!]^-(c) \text{ if } [\![\phi]\!]^+(c) = \emptyset \\ \lceil |c| \rceil \bigcup\{[\![\psi]\!]^-(c') \mid c' \in [\![\phi]\!]^+(c)\} \text{ otherwise.} \end{cases} \end{array}$$

The most puzzling part of this definition is perhaps the clause for $[\![\phi;\psi]\!]^-(c)$. Note that there are the following cases:

- In case $[\![\phi;\psi]\!]^+(c) \neq \emptyset$, it is clear that $[\![\phi]\!]^+(c)$ should yield $\emptyset$. This is the first case.
- In case $[\![\phi;\psi]\!]^+(c) = \emptyset$, this can be for two reasons:
  - $[\![\phi]\!]^+(c) = \emptyset$. In this case, the items in $[\![\phi]\!]^-(c)$ are the witnesses. This is the second case.

– $[\![\phi]\!]^+(c) \neq \emptyset$. Now it must be the case that every member c' of $[\![\phi]\!]^+(c)$ makes ψ fail. In this case the members of

$$\lceil |c| \rceil \bigcup \{[\![\psi]\!]^-(c') \mid c' \in [\![\phi]\!]^+(c)$$

are the witnesses. This is the third case.

Some examples may make this still clearer. Take the example $\exists; P0$, with initial empty context c, where $c_w = M$. Suppose there are objects with property P in the model M. Since $\exists$ always succeeds, and there is a d such that $d : c$ satisfies $P0$, we are in the first case, and $[\![\exists; P0]\!]^-(c) = \emptyset$. Suppose c' is an empty context, but with $c'_w = N$, and suppose there a no objects with property P in N. Since $\exists$ always succeeds, and there are no d such that $d : c$ satisfies $P0$, we are in the third case, and $[\![\exists; P0]\!]^-(c) = \{d : c' \mid d : c' =\!\mid P0\}$.

Next, take the example $P0; \neg(\exists Q5; R(0,5))$. Let c be a context of size 5 with $c_w = M$ and $c[*0] = d$, and M a model with Px true of d and $\forall y(Qy \Rightarrow Rxy)$ true of d. Then $[\![\exists; P0]\!]^+(c)$ is non-empty, but for any $c' \in [\![P0]\!]^+(c)$ we will have $[\![\neg(\exists; Q5; R(0,5))]\!]^+(c') = [\![\exists Q5; R(0,5)]\!]^-(c') = \emptyset$. So in this case $[\![P0; \neg(\exists Q5; R(0,5))]\!](c)$ will contain $(0)c$, the result of making the object with index 0 salient.

Here is an example to further illustrate the definitions of $[\![\cdot]\!]^+$ and $[\![\cdot]\!]^-$. Suppose we have a context with a reference to Mary, say, an item $(3, \mathbf{m})$ in a context of size 5. Let us assume the context, with its salience ordering, looks like this (suppressing the world coordinate):

$$[(4, \mathbf{b}), (2, \mathbf{a}), (1, \mathbf{c}), (3, \mathbf{m}), (0, \mathbf{d})]$$

Then *No woman loves Mary* will have translation $\neg(\exists; W5; L5\ 3)$ in this context. Suppose we are interpreting this translation in a model where the sentence is true. Then $\exists; W5; L5\ 3$ should turn out false. For that, every update of the context with an item $(5, w)$, where w is some woman in the model should render $L5\ 3$ false. The processing of this falsity check will have as a result that the context gets reshuffled to

$$[(5, w), (3, \mathbf{m}), (4, \mathbf{b}), (2, \mathbf{a}), (1, \mathbf{c}), (0, \mathbf{d})]$$

Finally, when this is cut down to the original size 5 of the input context, we get:

$$\begin{aligned} \lceil 5 \rceil\ & [(5, w), (3, \mathbf{m}), (4, \mathbf{b}), (2, \mathbf{a}), (1, \mathbf{c}), (0, \mathbf{d})] \\ & = [(3, \mathbf{m}), (4, \mathbf{b}), (2, \mathbf{a}), (1, \mathbf{c}), (0, \mathbf{d})]. \end{aligned}$$

The result is a salience update of the input context, with $(3, \mathbf{m})$ moved to the salient position.

The semantics given above can be rephrased as a function from states to states, as follows:

$$\begin{aligned} [\![\phi]\!]^+(S) &:= \bigcup \{[\![\phi]\!]^+(c) \mid c \in S\} \\ [\![\phi]\!]^-(S) &:= \bigcup \{[\![\phi]\!]^-(c) \mid c \in S\}. \end{aligned}$$

By induction on the structure of ϕ it can be checked that if S is a state (set of world-indexed contexts, all of the same length and under the same permutation), then both $[\![\phi]\!]^+(S)$ and $[\![\phi]\!]^-(S)$ are states as well.

States can also be used as representations of the common ground of speaker and hearer in a discourse, the background information shared by speaker and hearer.

6 Tests modulo permutation

The notion of a test on context now gets refined, as follows. Let $c \sim c' \quad :\Leftrightarrow$ c is a permutation of c'. Then a test-new-style on input context c is a formula ϕ with the property that the following holds:

$$\text{If } c' \in [\![\phi]\!]^+(c) \cup [\![\phi]\!]^-(c) \text{ then } c \sim c'.$$

The following theorem asserts that negations and predications are tests-new-style. The proof is immediate from the definitions.

Theorem 1 *For all ϕ in the language, for all contexts c:*

- *if $c' \in [\![Pi]\!]^+(c)$ then $c \sim c'$, if $c' \in [\![Pi]\!]^-(c)$ then $c \sim c'$,*
- *if $c' \in [\![Pij]\!]^+(c)$ then $c \sim c'$, if $c' \in [\![Pij]\!]^-(c)$ then $c \sim c'$,*
- *if $c' \in [\![\neg\phi]\!]^+(c)$ then $c \sim c'$, if $c' \in [\![\neg\phi]\!]^-(c)$ then $c \sim c'$.*

Note that double negation does change the interpretation of ϕ into a test modulo permutation, for we have:

$$[\![\neg\neg\phi]\!]^+(c) = [\![\neg\phi]\!]^-(c) = \lceil |c| \rceil [\![\phi]\!]^+(c).$$

7 Associativity of sequential composition

The next theorem reassures us that the associativity property of sequential composition still holds.

Theorem 2 *For all ϕ, ψ, χ in the language, for all contexts c:*

$$[\![\phi; (\psi; \chi)]\!]^+(c) = [\![(\phi; \psi); \chi]\!]^+(c),$$

and

$$[\![\phi; (\psi; \chi)]\!]^-(c) = [\![(\phi; \psi); \chi]\!]^-(c).$$

Proof. The case of $[\![\phi; (\psi; \chi)]\!]^+(c) = [\![(\phi; \psi); \chi]\!]^+(c)$ is straightforward. The reasoning for $[\![\phi; (\psi; \chi)]\!]^-(c) = [\![(\phi; \psi); \chi]\!]^-(c)$ is as follows.

First assume $\exists c' \in [\![\phi]\!]^+(c)$ with $[\![\psi; \chi]\!]^+(c') \neq \emptyset$. This is equivalent to $\exists c' \in [\![\phi; \psi]\!]^+(c)$ with $[\![\chi]\!]^+(c') \neq \emptyset$. Therefore, in this case $[\![\phi; (\psi; \chi)]\!]^-(c) = [\![(\phi; \psi); \chi]\!]^-(c) = \emptyset$.

Next assume $[\![\phi]\!]^+(c) = \emptyset$. Then by the definition of $[\![\cdot]\!]^-$ we have that $[\![\phi; (\psi; \chi)]\!]^-(c) = [\![\phi]\!]^-(c)$. From $[\![\phi]\!]^+(c) = \emptyset$ we have that $[\![\phi; \psi]\!]^+(c) =$

$\emptyset$. Therefore $[\![(\phi;\psi);\chi]\!]^-(c) = [\![\phi;\psi]\!]^-(c)$. Again by $[\![\phi]\!]^+(c) = \emptyset$ we get that $[\![\phi;\psi]\!]^-(c) = [\![\phi]\!]^-(c)$. Therefore, in this case, $[\![\phi;(\psi;\chi)]\!]^-(c) = [\![(\phi;\psi);\chi]\!]^-(c) = [\![\phi]\!]^-(c)$.

Finally assume $[\![\phi]\!]^+(c) \neq \emptyset$, and for all $c' \in [\![\phi]\!]^+(c)$ it holds that $[\![\psi;\chi]\!]^+(c') = \emptyset$. Two cases: (i) $[\![\phi;\psi]\!]^+(c) = \emptyset$ or (ii) $[\![\phi;\psi]\!]^+(c) \neq \emptyset$. In case (i),

$$\begin{aligned}
[\![\phi;(\psi;\chi)]\!]^-(c) &= \lceil|c|\rceil \bigcup\{[\![\psi;\chi]\!]^-(c') \mid c' \in [\![\phi]\!]^+(c)\} \\
&= \lceil|c|\rceil \bigcup\{[\![\psi]\!]^-(c') \mid c' \in [\![\phi]\!]^+(c)\} \\
&= [\![\phi;\psi]\!]^-(c) \\
&= [\![(\phi;\psi);\chi]\!]^-(c).
\end{aligned}$$

In case (ii), we get:

$$\begin{aligned}
[\![\phi;(\psi;\chi)]\!]^-(c) &= \lceil|c|\rceil \bigcup\{[\![\psi;\chi]\!]^-(c') \mid c' \in [\![\phi]\!]^+(c)\} \\
&= \lceil|c|\rceil \bigcup\{[\![\chi]\!]^-(c'') \mid c'' \in \bigcup\{[\![\psi]\!]^+(c') \mid c' \in [\![\phi]\!]^+(c)\}\} \\
&= \lceil|c|\rceil \bigcup\{[\![\chi]\!]^-(c') \mid c' \in [\![\phi;\psi]\!]^+(c)\} \\
&= [\![(\phi;\psi);\chi]\!]^-(c).
\end{aligned}$$

This completes the proof. □

8 Lift to Polymorphic Type Theory

Typed logic is the standard means of describing fragments of natural language with a compositional semantics, in the style of Montague. In the present case, we need polymorphically typed logic (see, e.g., (Hindley 1997, Milner 1978)).

We make use of the polymorphism to give a uniform treatment of contexts as lists: lists of entities of different lengths have different types, but they are all in the same class $[e]$. Lists of entities under a permutation are all in the same class $p[e]$ (permuted lists).

In (Eijck 2001), the translation of *a man* looked like this:

$$\lambda Qcc' \cdot \exists x(\text{man}(x) \wedge Qi(c\hat{}x)c') \text{ where } i = |c|.$$

The type of Q is the appropriate type for verb phrased in context semantics, i.e., $\iota \rightarrow$ *Trans*, where *Trans* is the type of context transitions, i.e., $[e] \rightarrow [e] \rightarrow t$. The new element x gets concatenated to the input context, with result $c\hat{}x$. Since c has length i, the item at position i in $c\hat{}x$ is the object x (this depends on the fact that we use 0 as the first index). Thus, i points to the object introduced by the indefinite, and i will continue to point to x in every extension of $c\hat{}x$. In other words, i plays the role of the index into the context that got incremented by the introduction of a referent for the indefinite.

Note that i is of the right type for an index into context $c\hat{}x$. Generally, an index for a context c is a natural number in the range $\{0, \ldots, |c| - 1\}$. If i is an index for c then $c[i]$ points to an entity. Type polymorphism can be used to ensure the fit between indices and contexts.

In the new set-up this gets refined with salience update. The new item x now effects a salience reshuffle, and we get:

$$\lambda Qcc' \cdot \exists x(\text{man}(x) \wedge Qi(x : c)c') \text{ where } i = |c|.$$

The types of the contexts is now $p[e]$, and the type of Q is modified accordingly (it is still $\iota \rightarrow$ *Trans*, but now *Trans* is shorthand for $p[e] \rightarrow p[e] \rightarrow t$). The referent for the indefinite that gets introduced appears in most salient position in the new context. Note that $(x : c)[*i]$ points to the newly introduced referent x, and in any c' that is an extension or permutation of $(x : c)$, $c'[*i]$ will continue to point to x.

It may seem that the integration of salience reshuffling in dynamic semantics precludes an account of the role of surface syntax in establishing salience. This impression is mistaken, for the context semantics is flexible enough to take syntactic effects on salience ordering into account.

Lambda abstraction allows us to make flexible use of the salience updating mechanism we presented in Section 5. In systems of typed logic, predicate argument structure is a feature of the 'surface syntax' of the logic. Consider the difference between the following formulas:

$$(\lambda xy.Kxy)(b)(j) \tag{K1}$$

$$(\lambda xy.Kyx)(j)(b) \tag{K2}$$

$$(\lambda x.Kbx)(j) \tag{K3}$$

All of these reduce to Kbj, but the predicate argument structure is different. Thus, surface predicate argument structure of lambda expressions can be used to encode the relevant salience features of surface syntax, and we can get the right salience effects from the surface word order of examples like the following:

1 *Bill kicked John.*

2 *John got kicked by Bill.*

3 *John, Bill kicked.*

To get the desired salience effect, one as to make sure that the sentence gets translated ionto a lambda expression with the appropriate predicate argument structure. This lambda expression can then be used for the salience update of the appropriate contexts.

In the dynamic semantics, sentences are interpreted as state transformers, noun phrases as functions from discourse predicates (type $\iota \rightarrow$ *Trans*) to state transformers (type *Trans*), and so on. The interpretation of proper names and pronouns involves resolution in context.

For the interpretation of common nouns, intransitive verbs and transitive verbs, we use 'discourse blow up': the types are lifted from $e \rightarrow t$ and $e \rightarrow e \rightarrow t$ to $\iota \rightarrow$ *Trans* and $\iota \rightarrow \iota \rightarrow$ *Trans*.

Interpretation of determiners uses the dynamic semantics of $\exists$, $\neg$ and sequential composition in a style that is very much like dynamic Montague grammar.

Interpretation of VPs consisting of a TV with a reflexive pronoun uses a relation reducer *self* that transforms a two-placed predicate of discourse type $\iota \rightarrow \iota \rightarrow \mathit{Trans}$ to a one-placed predicate (discourse type $\iota \rightarrow \mathit{Trans}$).

In the interpretation rule for $[_{\mathrm{VP}}\mathrm{TV\ NP}]$ (with NP not a reflexive pronoun) we impose a non-coreference constraint. This is necessary to get the right results for examples like *he respected him*. Here the well-known constraint should be imposed that *he* and *him* do not co-refer (see, e.g., (Reinhart 1983)). This constraint on coreference can be imposed with the help of an 'irreflexivizer'. The irreflexivizer maps a two-placed predicate in context semantics, with the appropriate type $\iota \rightarrow \iota \rightarrow \mathit{Trans}$ (a function that takes two indices and forms a state transformer) to another two-placed predicate that is just like it, except for the fact that the two indices are forced to be different. When imposed on $[_{\mathrm{VP}}\mathrm{TV\ NP}]$ where NP is not a reflexive pronoun, this has the desired effect of blocking coreference. In (Reinhart 1983), it is argued that the constraint on coreference is a pragmatic constraint, and that there are exceptions to this rule. Our implementation takes this nicely into account, for the non-coreference constraint is implemented as a constraint on context, not on the underlying reality, where the relation that interprets the TV need not be irreflexive.

9 Reference Resolution

Reference resolution picks the indices of the entities satisfying the appropriate gender constraint from the current state, in order of salience. Thus, the result of reference resolution is a list of indices, in an order of preference determined by the salience ordering of the state. The meaning of a pronoun, given a state, is an invitation to pick indices from the state, and use those indices to link pronouns to entities. This can be further refined in a set-up that also stores syntactic information (about gender, case, and so on) as part of the contexts.

Names are resolved by picking the most salient index to the named entity from the current state, by looking for an index in a state where all the entities at the index are referents for the name. Naming ambiguity is treated as a kind of special case of pronoun reference resolution. If a name has no index in context, the contexts in the state are extended with an index for the named object.

What this reference resolution mechanism provides is an ordering of resolution options determined by syntactic structure, semantic structure, and discourse structure, nothing more, nothing less. The mechanism has to be augmented by modules that take world knowledge into account. The mechanism can be viewed as an extension of pronoun reference resolution mechanisms proposed for DRT (Wada and Asher 1986, Blackburn and Bos 1999). The proposal also takes the so-called 'actor focus' from the *centering theory* of local coherence in discourse (Grosz and Sidner 1986, Grosz, Joshi and Weinstein 1995) into account. Central claim of centering theory is that pronouns are used to signal to the hearer that the speaker continues to talk about the same thing. Our approach demonstrates that reference resolution can be brought within the compass of dynamic semantics in a relatively straightforward way, and that very simple means are enough to implement something useful. Moreover, contexts ordered by salience are a suitable datastructure for further refinement of the reference resolution mechanism by means of modules for discourse focus and world knowledge (Walker, Joshi and Prince 1998).

10 Examples

In this section we will comment on some example runs with the implemented system. Reference resolution is done relative to a context list, with the position at the front of the list considered as salient. Let us start out with the context $[(0, \mathbf{j}), (1, \mathbf{b}), (2, \mathbf{m}), (3, \mathbf{a})]$. In this context **j** is the most salient item.

We will assume that **j** and **b** are men while **m** and **a** are women. We also assume a single background model. Note that this is a simplification. In a more realistic setting one would assume a set of background models, representing all the state of affairs that are compatible with the common knowledge among the participants in the discourse.

4 *He loved some woman.*

In a context where referents for the pronoun are available, *he* can be resolved to any referent that satisfies the property. And this is what we get. Our example context contains two women and two men. The referents `(1,B)` and `(0,J)` in the context are referents for men.

```
ResInContext> eval "He loves some woman."
[[([(1,B),(4,A),(0,J),(2,M),(3,A)],["Love" 1 4]),
  ([(1,B),(4,M),(0,J),(2,M),(3,A)],["Love" 1 4])]]
```

The outermost square brackets are for the list of parses, so in this case there is one parse. The square brackets in the immediate scope of the parse brackets indicate the list of possible output states per parse. In this case, there are two output states. In both states *he* gets resolved to `(1,B)`, because, as it happens, the other man in the initial context, **j**, does not love any women. In one output state *some woman* gets interpreted as `(4,A)`. In the other output state *some woman* gets interpreted as `(4,M)`. In both output state the subject ends up in most salient position, and the object in secondmost salient position.

It may seem unnatural that reference resolution is influenced by what happens to be the case in the model. After all, in the most salient reading of *John loved some woman. He smiled*, the pronoun in the second sentence should pick up a reference to *John*, even if it happens to be the case that John does not love anyone in actual fact. Note, however, that this problem is an artefact of our modelling. In the present implementation, there is just one background model. A more realistic set-up would have a *set* of background models, together constituting the common ground of the discourse participants. If the common ground is available as a resource, it is a reasonable constraint on pronoun resolution that the discourse participants consider the resulting reading possible. This constraint is built into our framework.

The only reason it does not show up properly here is because the representation of the common ground is still too poor.

5 *He hates some thing.*

Here we expect that we get all the new contexts where B or J with their objects of hatred are added, again with the subjects more salient than the objects. And this is what we get:

```
ResInContext> eval "He hates some thing."
[[([(0,J),(4,E),(1,B),(2,M),(3,A)],["H" 0 4]),
  ([(0,J),(4,F),(1,B),(2,M),(3,A)],["H" 0 4]),
  ([(0,J),(4,G),(1,B),(2,M),(3,A)],["H" 0 4]),
  ([(0,J),(4,H),(1,B),(2,M),(3,A)],["H" 0 4]),
  ([(0,J),(4,I),(1,B),(2,M),(3,A)],["H" 0 4]),
  ([(0,J),(4,K),(1,B),(2,M),(3,A)],["H" 0 4]),
  ([(0,J),(4,L),(1,B),(2,M),(3,A)],["H" 0 4]),
  ([(1,B),(4,E),(0,J),(2,M),(3,A)],["H" 1 4]),
  ([(1,B),(4,F),(0,J),(2,M),(3,A)],["H" 1 4]),
  ([(1,B),(4,G),(0,J),(2,M),(3,A)],["H" 1 4]),
  ([(1,B),(4,H),(0,J),(2,M),(3,A)],["H" 1 4]),
  ([(1,B),(4,I),(0,J),(2,M),(3,A)],["H" 1 4]),
  ([(1,B),(4,K),(0,J),(2,M),(3,A)],["H" 1 4]),
  ([(1,B),(4,L),(0,J),(2,M),(3,A)],["H" 1 4])]]
```

6 *Bill smiles. He loves some woman.*

Still working with the same initial context, we get:

```
ResInContext> eval "Bill smiles. He loves some woman."
[[([(1,B),(4,A),(0,J),(2,M),(3,A)],["Love" 1 4,"S" 1]),
  ([(1,B),(4,M),(0,J),(2,M),(3,A)],["Love" 1 4,"S" 1])]]
```

He gets resolved to Bill, for as it happens, in the model Bill is the only man in love. Note that the example still works out with an empty initial context. In that case we get the following outcome:

```
[[([(0,B),(1,A)],["Love" 0 1,"S" 0]),
  ([(0,B),(1,M)],["Love" 0 1,"S" 0])]]
```

The referent of *Bill* does not occur in the context, so it gets added.

7 *Bill smiles. He hates some thing.*

Evaluation in the empty context gives:

```
ResInContext> map (\x->intS x True ([],[])) (parse nex4)
[[([(0,B),(1,E)],["H" 0 1,"S" 0]),
  ([(0,B),(1,F)],["H" 0 1,"S" 0]),
  ([(0,B),(1,G)],["H" 0 1,"S" 0]),
  ([(0,B),(1,H)],["H" 0 1,"S" 0]),
  ([(0,B),(1,I)],["H" 0 1,"S" 0]),
  ([(0,B),(1,K)],["H" 0 1,"S" 0]),
  ([(0,B),(1,L)],["H" 0 1,"S" 0])]]
```

Since Bill is the only referent available for resolution of the pronoun, we get Bill with his objects of hatred as new contexts. In a richer initial context, we may get more, of course:

```
ResInContext> eval nex4
[[([(1,B),(4,E),(0,J),(2,M),(3,A)],["H" 1 4,"S" 1]),
  ([(1,B),(4,F),(0,J),(2,M),(3,A)],["H" 1 4,"S" 1]),
  ([(1,B),(4,G),(0,J),(2,M),(3,A)],["H" 1 4,"S" 1]),
  ([(1,B),(4,H),(0,J),(2,M),(3,A)],["H" 1 4,"S" 1]),
  ([(1,B),(4,I),(0,J),(2,M),(3,A)],["H" 1 4,"S" 1]),
  ([(1,B),(4,K),(0,J),(2,M),(3,A)],["H" 1 4,"S" 1]),
  ([(1,B),(4,L),(0,J),(2,M),(3,A)],["H" 1 4,"S" 1]),
  ([(0,J),(4,E),(1,B),(2,M),(3,A)],["H" 0 4,"S" 1]),
  ([(0,J),(4,F),(1,B),(2,M),(3,A)],["H" 0 4,"S" 1]),
  ([(0,J),(4,G),(1,B),(2,M),(3,A)],["H" 0 4,"S" 1]),
  ([(0,J),(4,H),(1,B),(2,M),(3,A)],["H" 0 4,"S" 1]),
  ([(0,J),(4,I),(1,B),(2,M),(3,A)],["H" 0 4,"S" 1]),
  ([(0,J),(4,K),(1,B),(2,M),(3,A)],["H" 0 4,"S" 1]),
  ([(0,J),(4,L),(1,B),(2,M),(3,A)],["H" 0 4,"S" 1])]]
```

8 *The mouse hates the cat. It smiles.*

We get that *it* can both be resolved to the mouse and to the cat, with a preference for the first resolution, as subject position is more salient.

```
ResInContext> eval nex5
[[([(4,I),(5,K),(0,J),(1,B),(2,M),(3,A)],["S" 4,"H" 4 5]),
  ([(5,K),(4,I),(0,J),(1,B),(2,M),(3,A)],["S" 5,"H" 4 5])]]
```

For examples like (8), (Kameyama 1998) has argued that the other order of resolution is more plausible, for relying on world knowledge (cartoon knowledge?) we can tell that a cat-hating mouse in sight of a cat is less likely to smile than a cat spotting a contemptuous mouse. But note that even that mechanism is taken into account by our little reference resolution engine. We simply take our model of the world as our yardstick for what is likely and what is not. In the model under consideration the mouse *does* smile, and that is a piece of world knowledge that makes the reading with *it* resolved to the mouse plausible.[1]

9 *The mouse respects the cat. It hates it.*

This time, we only get a reading where the mouse hates the cat. This is because it so happens that in the background model the cat does not hate the mouse.[2]

```
ResInContext> eval nex6
[[([(4,I),(5,K),(0,J),(1,B),(2,M),(3,A)],["H" 4 5,"R" 4 5])]]
```

10 *He respects him.*

[1]We may presume object *I* in the model to be Ignatz Mouse.
[2]The reason for this is that *K* in the model happens to be Krazy Kat.

Here the non-coreference constraint comes into play and forces the two pronouns to get resolved to different men in context.

```
ResInContext> eval nex7
[[([(0,J),(1,B),(2,M),(3,A)],["R" 0 1]),
  ([(1,B),(0,J),(2,M),(3,A)],["R" 1 0])]]
```

11 *The mouse respects itself. It hates the cat.*

We get the right outcome, with *it* resolved to the mouse, and the mouse ending up in most salient position in the output context:

```
ResInContext> eval nex8
[[([(4,I),(5,K),(0,J),(1,B),(2,M),(3,A)],["H" 4 5,"R" 4 4])]]
```

12 *Some man respects himself. Some woman loves him.*

Here things go slightly wrong:

```
ResInContext> eval nex9
[[([(5,A),(4,B),(0,J),(1,B),(2,M),(3,A)],["Love" 5 4,"R" 4 4]),
  ([(5,A),(0,J),(4,B),(1,B),(2,M),(3,A)],["Love" 5 0,"R" 4 4]),
  ([(5,A),(1,B),(4,B),(0,J),(2,M),(3,A)],["Love" 5 1,"R" 4 4]),
  ([(5,C),(4,B),(0,J),(1,B),(2,M),(3,A)],["Love" 5 4,"R" 4 4]),
  ([(5,C),(0,J),(4,B),(1,B),(2,M),(3,A)],["Love" 5 0,"R" 4 4]),
  ([(5,C),(1,B),(4,B),(0,J),(2,M),(3,A)],["Love" 5 1,"R" 4 4]),
  ([(5,M),(4,B),(0,J),(1,B),(2,M),(3,A)],["Love" 5 4,"R" 4 4]),
  ([(5,M),(0,J),(4,B),(1,B),(2,M),(3,A)],["Love" 5 0,"R" 4 4]),
  ([(5,M),(1,B),(4,B),(0,J),(2,M),(3,A)],["Love" 5 1,"R" 4 4]),
  ([(5,A),(0,J),(4,D),(1,B),(2,M),(3,A)],["Love" 5 0,"R" 4 4]),
  ([(5,A),(1,B),(4,D),(0,J),(2,M),(3,A)],["Love" 5 1,"R" 4 4]),
  ([(5,C),(0,J),(4,D),(1,B),(2,M),(3,A)],["Love" 5 0,"R" 4 4]),
  ([(5,C),(1,B),(4,D),(0,J),(2,M),(3,A)],["Love" 5 1,"R" 4 4]),
  ([(5,M),(0,J),(4,D),(1,B),(2,M),(3,A)],["Love" 5 0,"R" 4 4]),
  ([(5,M),(1,B),(4,D),(0,J),(2,M),(3,A)],["Love" 5 1,"R" 4 4]),
  ([(5,A),(4,J),(0,J),(1,B),(2,M),(3,A)],["Love" 5 4,"R" 4 4]),
  ([(5,A),(0,J),(4,J),(1,B),(2,M),(3,A)],["Love" 5 0,"R" 4 4]),
  ([(5,A),(1,B),(4,J),(0,J),(2,M),(3,A)],["Love" 5 1,"R" 4 4]),
  ([(5,C),(4,J),(0,J),(1,B),(2,M),(3,A)],["Love" 5 4,"R" 4 4]),
  ([(5,C),(0,J),(4,J),(1,B),(2,M),(3,A)],["Love" 5 0,"R" 4 4]),
  ([(5,C),(1,B),(4,J),(0,J),(2,M),(3,A)],["Love" 5 1,"R" 4 4]),
  ([(5,M),(4,J),(0,J),(1,B),(2,M),(3,A)],["Love" 5 4,"R" 4 4]),
  ([(5,M),(0,J),(4,J),(1,B),(2,M),(3,A)],["Love" 5 0,"R" 4 4]),
  ([(5,M),(1,B),(4,J),(0,J),(2,M),(3,A)],["Love" 5 1,"R" 4 4])]]
```

What we see is that for each man who respects himself we get for every woman that the pronoun resolves to its possible referents in the right order of plausibility, with *the man who respects himself* as the most salient referent for the pronoun. Of course, this is not the right overall order of plausibility. The problem is that choice of reference for indefinites is independent of salience order. This can be remedied by distinguishing between choices made by pronoun resolution steps and choices due to the dynamic semantics for indefinites.

This section was intended to give a taste of what pronoun resolution looks like in a framework based on incremental context semantics. In a more realistic implementation, the single background model should be replaced by a representation of the common ground. We have refrained from doing so in this paper for reasons of exposition. Explicit representation of the common ground on a realistic scale, in the form of a set of possible worlds, is not a feasible matter, but the issue of providing an implicit representation of the common ground, as a theory in a suitable representation language, is outside of our present scope.

11 Further Work

Representation of common ground and further integration of the framework with insights from centering theory are future work. Current work focusses on extending the format to distributive plurals, and to the treatment of dialogue. Dialogue is handled like regular text, but with a special operation for speaker shift, with a semantics that swaps the items in context marked for first and second person.

Acknowledgements The final version of this paper has benefitted from the comments of two anonymous reviewers.

References

Blackburn, P. and Bos, J.(1999), *Representation and Inference for Natural Language; A First Course in Computational Semantics — Two Volumes*, Internet. Electronically available from `http://www.comsem.org/`.

Eijck, J. v.(2001), Incremental dynamics, *Journal of Logic, Language and Information* **10**, 319–351.

Grosz, B. and Sidner, C.(1986), Attention, intentions, and the structure of discourse, *Computational Linguistics* **12**, 175–204.

Grosz, B., Joshi, A. and Weinstein, S.(1995), Centering: A framework for modeling the local coherence of discourse, *Computational Linguistics* **21**, 203–226.

Hindley, J. R.(1997), *Basic Simple Type Theory*, Cambridge University Press.

Kameyama, M.(1998), Intrasentential centering: a case study, *in* M. Walker, A. Joshi and E. Prince (eds), *Centering Theory in Discourse*, Clarendon Press, pp. 89–112.

Milner, R.(1978), A theory of type polymorphism in programming, *Journal of Computer and System Sciences*.

Reinhart, T.(1983), *Anaphora and Semantic Interpretation*, Croom Helm, London.

Vermeulen, C.(1993), Sequence semantics for dynamic predicate logic, *Journal of Logic, Language, and Information* **2**, 217–254.

Wada, H. and Asher, N.(1986), BUILDRS: An implementation of DR theory and LFG, *11th International Conference on Computational Linguistics. Proceedings of Coling '86*, University of Bonn.

Walker, M., Joshi, A. and Prince, E. (eds)(1998), *Centering Theory in Discourse*, Clarendon Press.