Generating Easy References: the Case of Document Deixis

Ivandré Paraboni and **Kees van Deemter**
Information Technology Research Institute
University of Brighton
United Kingdom
ivandre@terra.com.br, Kees van.Deemter@itri.brighton.ac.uk

Abstract

This paper argues that algorithms for the generation of referring expressions should aim to make it easy for hearers and readers to *find the referent* of the expressions that are generated. To illustrate this claim, an algorithm is described for the generation of expressions that refer across a hierarchically ordered domain, and which takes search effort into account by adding logically redundant information. To support the ideas underlying the algorithm, a psycholinguistic experiment is presented that confirms readers' preference for the generated, logically redundant expressions over non-redundant alternatives.

1 Introduction

Common sense suggests that unnecessarily complex utterences are best avoided by speakers and writers who want to get their message across and, broadly speaking, this idea is confirmed by empirical research (Clark 1992, Cremers 1996). The present paper will discuss the complexity of *referring* expressions, and the implications that arise for Natural Language Generation (NLG) if unnecessary complexity of referring expressions is to be avoided. We focus on the complexity of *search*, bypassing complexity of *interpretation* (i.e., determining the meaning or logical form of the referring expression). Search is understood here as the effort needed to 'find' the referent of an expression once the meaning of the expression has been determined. Note that complexity of search and interpretation are often inversely related. The expression (b), for example, is longer and more difficult to *interpret* than (a), but the additional material in (b) makes *search* easier once interpretation is successfully completed. Both (a) and (b) determine their referent uniquely.

a. *68 Lincoln Street, Brighton*
b. *68 Lincoln Street, in the middle of Brighton's Hannover area*

We will explore how an NLG program can generate the kind of redundancy exemplified in (b), so as to simplify the search for a referent.[1] We focus on a domain whose inhabitants are parts of a document rather than of the 'real' world. (See the Conclusion section for some remarks about other types of domains.) Examples of the expressions generated are

c. *picture 1*
d. *picture 1, in section 2*

We focus on the first time an entity is mentioned, disregarding anaphoric references (e.g., 'it').

2 Generating references that are easy to resolve

Generation of Referring Expressions (GRE) is a key task of NLG systems (e.g., Reiter and Dale 2000, section 5.4). The task of a GRE algorithm is to find combinations of properties that allow the generator to refer uniquely to an entity, called the *target* of the algorithm, and to express these properties in a linguistic description. We will focus on the first part of the problem, which involves determining the semantic content of a description, paying no attention

[1] Compare Jordan 2000, Jordan (forthcoming), where other factors triggering logically redundant material in referring expressions are explored.

to linguistic realization (e.g., Malouf 2000). When there is no risk of confusion, we will use the term 'description' loosely to refer to either the combination of properties or it linguistic realization.

GRE algorithms take as their input a knowledge base that is shared between writer and reader, generating unique descriptions of entities whenever the knowledge base allows it (van Deemter 2002). These algorithms are designed in such a way that *generation* is made easy (i.e., quick). The implications for the reader, for example in terms of the difficulty of *finding the referent*, are never taken into account. Note that some algorithms *do* generate descriptions that are optimally brief (Dale 1989), while others *approximate* optimal brevity (Dale and Reiter 1995), and this can be argued to make interpretation easier. As we have seen, however, a short description can sometimes make search difficult.

To illustrate, the Incremental Algorithm works roughly as follows. (See Dale and Reiter 1995 for details). Attributes represented in the shared Knowledge base are ordered in a linear preference ordering. The Attributes in the ordering are considered one by one, to see if any of their Values contributes something to the description, typically by removing 'distractors' (i.e., objects other than the referent); if an Attribute (e.g., COLOUR) can contribute something then a suitable Value (e.g., RED) for this Attribute is selected as a part of the description. This is repeated 'incrementally' until the logical conjunction of all selected Attribute/Value combinations result in a unique identification of the referent. There is no backtracking, and this is what keep the complexity of the algorithm linear.

While relying on the shared Knowledge Base, GRE algorithms such as the Incremental Algorithm fail to account for the time or effort that it takes a reader to discover which properties (i.e., which Attribute/Value combinations) are true of a given object. Once a description has been interpreted, at least two things can make it difficult to find its referent: the 'opacity' of the properties used in the description, and the size and structure of the search space.

To exemplify the first factor, we may refer to someone as 'the person wearing green underwear', but the colour of someone's underwear is not always easy to assess. In the Incremental Algorithm, this can be tackled easily, by taking the opacity of an Attribute into account in the preference ordering (see above) of Attributes: the algorithm could give preference to COLOUR OF ONE'S COAT over COLOUR OF ONE'S UNDERWEAR. As a result, the former Attribute would be considered (and possibly added to the description) before the latter. As a consequence, COLOUR OF ONE'S UNDERWEAR would only be considered if the referent cannot be identified uniquely without using a combination of more preferred Attributes, including COLOUR OF ONE'S COAT.

In what follows, we will focus on the second factor: the size and structure of the search space.

3 Generating document-deictic expressions

Document-deictic references (DDXs, for short) are expressions in a document which refer to a part of the same document. The term document deixis parallels Webber (1991)'s discourse deixis : 'deixis' because the referent of a DDX depends on its spatial co-ordinates (Lyons 1977); 'document' because its referent lives in a document rather than a discourse (Paraboni 2000). Examples of DDXs are the noun phrases (c) and (d) of section 1. Such expressions point the reader's direction to a part of the document that they are not presently reading. Sometimes they also serve as an indirect way of characterizing an entity in the world, as in 'The objects mentioned in section 2.3'. DDXs take the document itself as their domain, which is hierarchically ordered (in sections, subsections, etc.), allowing us to picture it as a tree.

Documents and the hierarchical relations that hold them together can be modelled in different ways. For present purposes, we stick with a simple Attribute/Value model.[2] One Attribute, TYPE, says

[2]At first sight, a Dale and Haddock-style modeling based on relational properties looks promising for the modeling of a hierarchy, but appearances are deceptive. Consider, for example, a description like 'the vase on the table', which can be said when there are many tables, as long as only one of them supports a vase (Dale and Haddock 1991). By comparison, if a document contains only two pictures, one of which occurs in the only Appendix, while the other occurs in one of the many

what kind of entity it is (picture, table, section, subsection, part, etc.). Given its TYPE, the identity of the referent *within its parent node* can usually be determined by means of one other Attribute. For example, a picture might be identified by the property TYPE: PICTURE and the property PICTURENUMBER: 3, the combination of which may be realized as 'picture (iii)'. Often, of course, this will not be enough to identify the referent *within the document as a whole*, necessitating addition information, as in 'picture (iii) *in/of section 2*'. Such additions will play a crucial role in what follows. Note that we will focus on 'hierarchical' properties of Document Entities. For simplicity, (and also because they cannot always be predicted by the author) page numbers are not taken into account here.[3]

The search for the referent Ref(d) of a document-deictic reference d is, of course, sensitive the location of d, but sometimes the words in d cause the 'starting point' for the search to be somewhere else, for example when d contains a phrase of the form 'in section ...'. More precisely, let d be of the form 'A in/of B', then B can either be empty, in which case d equals A, or B can itself be a DDX. We will assume that B provides the starting point s for search, unless B is empty, in which case s is d's parent. We assume that search is structured as follows:

> **Ancestral Search:** First, search for Ref(d) in the subtree dominated by the starting point s. If Ref(d) is not found there then search for Ref(d) in the subtree dominated by the parent of s, which is called s'. If Ref(d) is not found there then move up to the parent s'' of s',.., etc., until the root is reached. If, at this point, Ref(d) is still not found, search fails.

Note that this leaves the details of the search process unspecified. In particular, Ancestral Search leaves it open how the search *within* a given subtree (i.e., the subtree dominated by s or s', etc.) is carried out. Yet, it has various nontrivial implications. In particular, it predicts that more elaborate descriptions do sometimes, but not always, simplify search.

Example: Suppose D contains *only one* document entity whose TYPE is PICTURE and whose PICTURENUMBER is 1. Sections are identified (within their parent node) by the Attribute SECTIONNUMBER, parts by their PARTID.

```
                    D
     section1          section2
       /    \        /    |    \
partA    partB  partA partB partC
  |                           |
  d                     picture 1
```

Suppose the referring expression d occurs in part A of section 1. Then some of the ways in which we could uniquely identify the referent are:

d_1. *picture 1 in part C of section 2*
d_2. *picture 1 in section 2*
d_3. *picture 1 in part C*
d_4. *picture 1*

Each of these would require a certain amount of search effort. As the reader may verify, Ancestral Search predicts that the referring expression d_1 requires less search than the shorter expressions d_2 because section 2 includes part C. The referring expression d_4 fares worst of all: It is a perfectly unique description of its referent, and it does so in the briefest possible way, but Ancestral Search predicts that it is difficult to interpret: the starting point s is part A of section 1, where Ref(d) is not found; at the next step, (the rest of) section 1 is searched and, once again, Ref(d) is not found. Next, all (the rest of) D is searched and finally Ref(d) is found. By contrast, adapting the example, if the picture and d both lived in part A of section 1 then the description 'picture 1' would require the same amount of search as the description 'picture 1 in part A '.

Going back to the original example, note that a Reference like 'part C' (in d_3) is *qualitatively* worse than 'part C of section 2' in d_1, since it not only forces a reader to search through an unnecessarily large part of the document (i.e., the whole document instead of only section 2): by forcing the reader to 'travel' beyond a point (i.e., beyond section 1) where the referent *might* have been found, they can cause confusion.[4] Even if the reader eventually comes

sections; then 'the picture in the section' is not an acceptable description, whereas 'the picture in the Appendix' is.

[3]Page numbers offer a secondary ordering of the document, alongside its hierarchical structure; if they are taken into account then the generation algorithm has to be adapted. (Cf., section 5 on nonhierarchical domains.)

[4]The referent 'might have been found' in section 1 because the Attribute PARTID is defined there, as is witnessed by the

across part C, she has no simple way of knowing whether that document part is the intended referent (for perhaps part C of section 1 has been overlooked). In cases like this, we speak of a *lack of orientation* (LO). Evidence confirming that writers tend to avoid LO is discussed in section 4.

An even more problematic situation occurs when a DDX contains a 'dead end' (DE). Suppose the document contains only two pictures, located in parts B and C of section 2, respectively. Then for d (in part A of section 1) to refer to the one in part B as 'the picture in part B' would be logically sufficient; nevertheless, this description would be a recipe for disaster: Ancestral Search predicts that the reader will reach a point where she searches all of section 1, in which she finds a section whose PartId is B, exactly as required – and yet it is the wrong section, since the intended referent lives in part B *of section 2*.[5]

The following generation algorithm reduces the amount of search performed by an interpreter *if Ancestral Search holds*. The document part r is the referent. L is the list of properties delivered by the algorithm, to be turned into an English description by a language realization program.

```
Full Inclusion($r$):

• $L := \langle\rangle$  { Initialize $L$ as the empty list }

• Identify($r$)
```

The function 'Identify' is defined recursively:

```
Identify($r$):

• $L :=$ Type(r)  { Include $r$'s Type in $L$ }

• $L := L + P$, where $P$ identifies $r$ uniquely
  within Parent($r$)

• IF Parent($r$) $= D$
  OR Parent($r$) includes Parent($d$)
  THEN STOP
  ELSE Identify(Parent($r$))
```

two document parts A and B in that section.

[5]One can even question the *correctness* of 'the picture in part B', under these circumstances. (Cf., Dale and Haddock 1991; see also footnote 1).

'Includes' is taken to be a reflexive relation: 'a includes b' iff either a is an ancestor of b or $a = b$. Applied to our earlier example, in the reference to part C Full Inclusion first builds up the list $L = \langle$Type=Part, PartId=C$\rangle$, then expands it to $L = \langle$Type=Part, PartId=C, SectionNumber=2$\rangle$. Now $r =$section 2 has been identified within D, and since Parent$(r) = D$, we reach STOP. L can be realized as 'part C of section 2'.

4 Experiment

We wanted to find out whether referring expressions that reduce the complexity of search are actually preferred. Dead ends (DE) and Lack of orientation (LO) can only occur in fairly complex documents. Instead of trying to find a large number of such documents, we opted for an experimental aproach. More specifically, we asked subjects to compare different document-deictic references and to choose the one they found more suitable for each situation in a given document. One group of DDXs was produced by the Full Inclusion (FI); the other group of DDXs contained no more information than what is minimally required to make the DDX uniquely distinguishing; the latter category included examples of DE and LO.

The experiment made use of a specially designed schematic document (see *Appendix*). The document contains a number of incomplete statements, to be completed by the subject. The following is an (out of context) example of such an incomplete statement:

The green star is shown in
( ) part C of section 2
( ) part C

For each statement of this type, two alternatives are offered, one 'minimally distinguishing' description and the other corresponding to the output of the FI algorithm. Both alternatives are unambiguous references to the same object, the only difference being that the minimally distinguishing alternative involves a situation of DE or LO, which is prevented in the FI alternative. Our research hypotheses were that in situations of DE, and also in situations of LO, the expression produced by the FI algorithm would be chosen more often than the minimally distinguishing alternative.

Design of experiment. The experiment made use of 15 subjects, all of them with practice in document authoring. The subjects were instructed to put themselves in the shoes of the authors and to complete the referring expressions taking the (document-deictic) context into account. The document itself was presented in a printed version (3 pages-long, with no page numbers). Each subject was asked 16 questions, four of which were potential LOs (depending on the expression chosen), four were potential DEs, four were cases in which DE and LO do not occur, and four were control questions (to break the monotony of the task and to disguise the purpose of the experiment).

Half of the questions, in each of the 4 groups, involved backward references (i.e., to document parts preceeding d), the other half involved forward references. Within the LO and control groups, two of the four questions involved references to pictures, while the other two involved references to sections; all the questions involving DEs involved references to pictures. (Using DE references to sections would have led to highly artificial structures.) The DDX d and the referent r were always located on different pages of the document; had the referring expression d and the referent r occured on the same page of a document then physical proximity might have obscured navigational issues, leading to a bias towards the shortest alternative. When d and r occur in document parts with similar layout properties (e.g., when the positions of both d and r in the document happen to be subsections labelled as "C" in different sections of the document) there could be a bias towards the most complete (i.e., the longest) description; to avoid such a cheap victory, d and r were always put in document parts whose layout properties are different from each other. The presentational order of alternatives (i.e., minimally distinguishing versus FI) was evenly distributed.

Results. Situations of DE (100%) were *always* avoided. In situations involving LO, the FI version was chosen in 93% of cases, which is highly significant (p = .002723). In both cases, the research hypothesis was confirmed. In the cases not involving DE or LO, there was no significant preference for or against logical redundancy.

Discussion. The experiment offers support for the idea that references should include logically redundant information to simplify search in those cases where this makes the task of finding the referent (for example, as specified by Ancestral Search) easier. Asking subjects to compare the adequacy of candidate referring expressions, however, is an admittedly indirect way of gauging their reading behaviour, which might say as much about their writing behaviour as about their reading behaviour. An interesting alternative would be to measure reading times and/or text comprehension, with 'type of referring expression' as the independent variable.

5 Conclusion

This paper has discussed generation strategies that reduce the search for a referent by adding logically redundant properties. The properties that are added contain information about the place of the referent within a hierarchically structured domain. Algorithms for generating logically redundant references along the lines described here have been implemented in PROLOG. The descriptions generated do not always *minimize* search: to do this, one would have to revise Full Inclusion to ensure that the DDX identifies the parent of the referent, even if it includes the Parent of the DDX. This would sometimes lead to lengthy descriptions which would be difficult to interpret while requiring only slightly less search.[6]

The cost of search is not determined by the 'logical' structure of the domain alone. A pilot corpus study on twelve books by different authors and publishers and covering different fields of knowledge suggests that short descriptions, *without* indication of the place of the referent within the hierarchical structure, are preferred when search is unproblematic (i.e., when LO and DE do not play a role). Suppose, for example, that pictures are enumerated throughout the document. Then the description 'picture 43' makes its referent easy to find even though the description does not wear locational information on its sleeves (Paraboni 2000). Note that, in this paper, issues like the average physical distance be-

[6]An example arises in our example tree if the DDX d lives in part A of section 2. In this case, a reference that minimizes search would say 'picture 1 in part C *of section 2*.

tween referring expression and referent have been left out of consideration, but they are likely to play a role. In the previous example, for instance, if there are pictures on virtually every page then this makes it easier to find the picture.

Other domains. Although we have focused on the generation of DDXs, we expect that the main ideas carry over to references in other domains and settings. A number of cases may be distinguished. Firstly **(1)**, if a domain is hierarchically organized and references are deictic, for example, (i.e., they involve something analogous to our point d, where the DDX originates) then it is so similar to the one studied here that we expect referring expressions to behave in exactly the same way. An example that comes to mind is that of a speaker who explains the location of a house somewhere else in the same town. Suppose, for example, that only one street in Brighton has numbers above 2000 then '2568 Lincoln Street' is much better than 'the house with street number 2568'. (Cf. examples (a) and (b) in section 1.) Secondly **(2)**, if the domain is hierarchical while references are *not* deictic then the starting point s for Ancestral Search (see section 3) defaults to the root of the document (unless, of course, the DDX indicates another starting point). But even if, thirdly **(3)**, the domain is *not* hierarchical, the goals of making references easy does apply, and domain structure can sometimes be exploited to achieve it. Imagine a long line of people, for example, one of which has on a grey coat. Then 'the man in the grey coat' is a unique description, but 'the man in the grey coat, about 30 meters from here' could make it far easier to find the referent. In future work we intend to address search in nonhierarchical domains, and the question how this affects the generation of referring expressions.

6 Acknowledgements

This work is part of an ongoing PhD project of the first author and it has been supported by the CNPq, the Brazilian Research Council.

7 References

Clark, Herbert. 1992. *Arenas of Language Use*. CSLI Publications, Stanford, Ca.

Cremers, Anita. 1996. *Reference to Objects; an empirically based study of task-oriented dialogues.* Ph.D. thesis, University of Eindhoven.

Dale, Robert. 1989. Cooking up referring expressions. In Procs. of 27th Annual meeting of the Association for Computational Linguistics (ACL-1989), p.68-75.

Dale, Robert and Nicholas Haddock. 1991. Generating Referring Expressions involving Relations. Procs. of EACL, Berlin, pages 161-166.

Dale, Robert and Ehud Reiter. 1995. Computational Interpretations of the Gricean Maxims in the Generation of Referring Expressions. *Cognitive Science* 18: 233-263.

Lyons, John. 1977. *Semantics*. Cambridge University Press.

Jordan, Pamela W. 2000. Can Nominal Expressions Achieve Multiple Goals?: An Empirical Study. Procs. of ACL-2000, Hong Kong.

Jordan, Pamela W. (forthcoming). Contextual Influences on Attribute Selection for Repeated Descriptions. To appear in Van Deemter and Kibble (Eds.) *Information Sharing: Reference and Presupposition in Language Generation and Interpretation*. CSLI Publications, Stanford, Autumn 2002.

Malouf, Rob. 2000. The order of prenominal adjectives in natural language generation. In procs. of ACL-2000.

Paraboni, Ivandré. 2000. An algorithm for generating document-deictic references. Procs. of workshop Coherence in Generated Multimedia, associated with First Int. Conf. on Natural Language Generation (INLG-2000), Mitzpe Ramon. pp.27-31.

Reiter, Ehud and Robert Dale. 2000. *Building Natural language Generation Systems.* Cambridge University Press, Cambridge, UK.

van Deemter, Kees. 2002. Generating Referring Expressions: Boolean Extensions of the Incremental Algorithm" *Computational Linguistics* **28** (1), pp.37-52. March 2002.

Webber, Bonnie. 1991. Structure and Ostension in the Interpretation of Discourse deixis. *Language and Cognitive Processes* 6(2) May 1991, pp. 107-135.