Information sharing

Book editors

August 22, 2001

CENTER FOR THE STUDY
OF LANGUAGE
AND INFORMATION
Contents

1 Towards the Generation of Document-Deictic
   References 1
Towards the Generation of Document-Deictic References

Abstract. In this paper, the notion of document deixis is introduced as a cross-referencing device distinct from anaphora, and a number of problems for the computational generation of document deixis are presented. A preliminary proposal for the generation of two types of document deixis is sketched and the impact on the architecture of a natural language generation system is discussed.

1.1 Introduction

Consider a written document dealing with a certain domain of discourse. Typically, most referring expressions in the document will refer to entities in this domain. The noun phrases in medical patient information leaflets, for example, talk about the medicine and the patient, as well as about such more abstract entities as actions (e.g., taking a pill), side effects (e.g., headache), and remedies. But, in addition to this, the leaflets contain expressions referring to entities of a very different kind: sections, paragraphs, pictures, etc., which do not ‘live’ in the domain of discourse. Instead, they are parts of the document itself. Examples include the following (cf. ABPI 1997).

1a. This booklet gives you information about (name of medicine)
1b. Read section F before using your medicine.
1c. The side effects mentioned above should disappear after a few days.
1d. Stick the patch onto your skin as shown in the picture.

Documents may of course refer to other documents. For example, a scientific paper usually refers to a number of other papers. We will discuss the phenomenon of documents referring to their own parts. Moreover,
we will limit ourselves to *continuous* document parts, which are not interrupted by other parts of the document.\footnote{An example of a reference to a discontinuous document part might be the NP ‘the footnotes in this paper’ which, depending on the style in which the footnotes are realised, may or may not be a continuous part of the document.}

The phenomenon under discussion has received little attention\footnote{This is true with the partial exception of work on multimedia generation, where the problem of generating *cross-media* referring expressions has been addressed (e.g., McKeown et al. 1992; Andre & Rist 1994). See section 1.4.3 for some relevant remarks.}, and no consensual terminology appears to be in usage.

Expressions like ‘the previous section’ are deictic, in the sense that they involve an implicit or explicit reference to the immediate situation of the reader (‘the section before the one you’re reading now’).\footnote{Note the difference with *anaphoric* expressions, which do not refer to parts of the document but to ordinary domain entities - albeit with help from their antecedents.} Consistent with this view, (Lyons 1977) defines the expressions which refer to, but are *not co-referential with* any preceding linguistic form, as ‘textual deixis’. ‘Impure’ textual deixis, according to Lyons, is the case where the reference is to abstract entities such as facts, and ideas, as in ‘This idea was first put forward by Aristotle’, where the subject refers to the idea expressed by some previous text, rather than the text itself. Impure textual deixis has also been dubbed ‘discourse deixis’ and this phenomenon has recently received a fair amount of attention (Levinson 1983; Webber 1991; Asher 1993; Eckert & Strube 1999). Many of the phenomena we are interested in could be subsumed under Lyons’ textual deixis, but we will avoid this term since it suggests an unwarranted limitation to references to text excluding references to pictures and other graphical components of documents. Therefore, we use the term *document deixis* (d deixis) to refer to references to document parts, paralleling the use of the term ‘discourse deixis’ (Webber 1991; Paraboni & van Deemter 1999).

The goal of this paper is twofold. First, we intend to discuss this little-known linguistic phenomenon, discuss how it differs from a such related phenomena as anaphora and discourse deixis, and introduce some of its main incarnations. Second, we will try to shed more light on the phenomenon by asking how document-deictic expressions may be produced by a natural language generation (N.L.G) programme of a particular kind. This N.L.G perspective will allow us to ask when it is proper to use a document-deictic expressions and what is its proper form in a given situation.

In section 1.2, we introduce the phenomenon of document deixis.
tion 1.3, we formulate assumptions about the type of natural language generation system under which we intend to generate DDX. Sections 1.3 and 1.4 explore how the semantic content and the surface realisation of DDX can be generated. Section 1.4.3 describes some of the consequences of DDX generation for the architecture of an NLG program. The final section (1.5) summarizes and draws some conclusions.

1.2 The phenomenon of Document Deixis

At the heart of the issue of document deixis lies an ontological distinction, between what we will call *domain entities* (e.g., pills, patients, side effects), which lead a life independently of the document, and *document entities* (e.g., sections, paragraphs, lists, sentences), which are only created by the author during the writing process. Accordingly, one can distinguish between Noun Phrases referring to document entities (NP_{dom}) and Noun Phrases referring to domain entities (NP_{doc}). Interestingly, however, both types of NPs can contain NPs of the other type as a constituent. For example, an NP_{dom} can contain an NP_{doc} (e.g., 'the side effects described in section 4'); the reverse is also possible (e.g., 'the section that warns against side effects'). In principle, such embeddings can get very complex, e.g.

[The advice given in [the section that warns against [the side effect described in [section 5]]]].

In fact, such embeddings can be arbitrarily deep:

\[ \cdots \text{NP}_{dom}[\cdots \text{NP}_{doc}[\cdots \text{NP}_{dom}[\cdots \text{NP}_{doc}[\cdots \text{NP}_{dom}[\cdots \cdots \cdots ] \cdots ] \cdots ] \cdots ] \cdots ] \cdots \]

(The innermost NP can be an NP_{dom} or an NP_{doc}, and the same is true for the outermost NP.) Our study of existing corpora, however, suggests that most of these embeddings are extremely infrequent.\(^4\) Henceforth, we will limit attention to unembedded DDX and embeddings of the form NP_{dom}[..NP_{doc}[..]]. Let us look at these two types in turn.

First, the unembedded kind. This most pure form of DDX is frequent in the introductory parts of scientific papers, for example, where they announce which issues are treated in which section. For instance,

2. In the previous section we have introduced the phenomenon of document deixis

Sentences containing such DDX tend to be semantically redundant, making explicit something which is true of the document itself. As a result, they are one of the most common instantiations of self-reference (e.g.,

\(^4\)This is true even if we discard uninformative descriptions like ‘The advice given in the section that gives advice’.

Barwise & Etchemendy 1987; Hofstadter 1985). Self-reference, as is well known, can cause a sentence to be self-affirming (‘This sentence contains five words’), self-defeating (‘This sentence contains exactly five words’), or paradoxical (‘This sentence is false’). Each of these complications can arise from document deixis: If a text contains meta-level DDX then this causes the text as a whole to become self-referential, leading to the logical peculiarities just mentioned. For example, consider a document D containing the example sentence (2). There are two possibilities: either the indicated section describes the side effects or it does not. In the first case, the truth of (2) can be inferred by reading the remainder of the document (in particular the next section); hence, (2) becomes analytical; in the second case, (2) makes the document self-defeating.

DDX that are used in this way will be called meta-level DDX and we will distinguish them from another type, which we will call object-level DDX. Object-level DDX are usually part of a larger constituent (e.g. ‘the pills described under the header ‘Your medicine’’) which refers to a domain entity. The domain entity in question may be concrete (e.g., a medicine, as in 3a) or abstract (e.g., a problem, as in 3b), in which case we are dealing with discourse deixis (Webber 1991).

3a. The pills described under the header ‘Your medicine’ are best swallowed with water.

3b. The problems described in section 5 need to be discussed with your doctor.

Since object-level DDX have as their primary purpose to refer to a domain entity, via the reference to a document entity, they are typically part of a statement that adds to the logical content of the document. Having said this, object-level DDX can, just like any other type of referring expressions, contain logically superfluous material (e.g. (Dale & Reiter 1995), section 2.4); in particular, the DDX itself can be superfluous (i.e., not necessary for the identification of the intended referent). Consider, for example, the following sentence, written in a context where the deictic part of the description (i.e., ‘in section 5’) is non-restrictive because we assume that side effects are mentioned in one section only.

4. Consider (...) the side effects mentioned in section 5.

On the one hand, the deictic description ‘section 5’ is instrumental in the reference to a particular class of side effects (which may be most easily identified by looking them up in section 5); on the other hand, the deictic description might also be motivated by the need to point the reader to some information, also contained in section 5, about these side effects.
Towards the Generation of Document-Deictic References

Similar complications arise in relation to meta-level DDX. The core function of a meta-level DDX is not to add to the logical content of the document (i.e., to make claims about the domain) but to increase its readability, for example by pointing the reader to a part of the document that the author deems instructive for some reason. Having said this, meta-level DDX can add logical content. A slight variant of sentence (1a), for example, uses a meta-level DDX to make the novel point that the information in the leaflet is important.\(^5\)

1a. This booklet gives you information about (name of medicine)
1a'. This booklet gives you important information about
(name of medicine)

To sum up the main issues arising from this discussion, object-level DDX are primarily used as a part of a statement that makes a document more informative (but nonrestrictive properties may serve to make the statement more accessible); meta-level DDX, by contrast, are primarily employed to make a document more accessible (but they sometimes add information about the domain as well). These complications notwithstanding, it will prove to be useful to distinguish these two types of DDX.

1.3 Generating Document Deixis

We now turn to questions of natural language generation. Firstly, we will introduce a class of NLG systems, one of whose tasks it is to find out under what circumstances a document-deictic expression is called for, as well as to determine its proper form (section 1.3.1). Since document-deictic references are referring expressions, it will be useful to make some preliminary remarks about existing algorithms for the generation of referring expressions (section 1.3.2) before we turn to the generation of document-deictic reference in section 1.4.

1.3.1 Natural Language Generation for Document Authoring

NLG systems may serve different purposes, and the nature of the texts that they produce can vary greatly. In this paper, we will be focusing primarily on a class of NLG systems whose purpose is the authoring of complex documents. Systems of this kind have a user who is trying to create a document, assisted by the system. The user, or author, specifies the content (and sometimes, but only at a high level of abstraction, the

---

\(^5\)In some cases, a meta-level DDX may even be employed to clarify reference. For example, in “The red 30mg pills are best swallowed with water. (See the text under the header ‘Your medicine’ to be sure you’re taking the right pills),” the text in brackets involves a meta-level DDX whose purpose is to add information about the reference of the NP ‘The red 30mg pills’.
form) of the document, whereupon it is the responsibility of the system
to convert this specification into a coherent document in a given target
language. Document authoring systems try to make the creation of doc-
ments — possibly in a number of languages at the same time — easier
by taking away from the author much of the responsibility over the lin-
guistic form of the document. In its most extreme form, the user of this
type of system is no longer a writer, but just a specifier of content.

In actual practice, the degree to which the author is still involved in
decisions about the form of a document can vary. Document-authoring
software can, for example, allow an author to ask for a text without
semicolons, to avoid certain types of overly complex texts (Bouayad-
Agha et al. 2000). Even these systems, however, take away low-level de-
tails such as lexical choice, and the construction of referring expressions,
from the author. For example, the author may specify that a certain per-
son a has a certain property F. Whether a is referred to using a proper
name, a pronoun, or some lengthy description — possibly involving an
object-level $\text{dx}$ — is up to the system, and analogously for the way in
which the property $F$ is designated. Before we turn to the question how
$\text{dx}$ can be generated by a system of this kind, let us briefly sketch how
‘ordinary’ referring expressions are generated.

1.3.2 Referring Expressions Generation Algorithm

Generation of referring expressions is a key component of NLG programs.
Two types of algorithms are used in this area, one of which deals with
NPs that stand in an anaphoric relationship with an earlier-generated
NP and where the main problem is to determine what type of NP is most
appropriate: a description, a proper name, a pronoun, or a demonstrative
for example (Passonneau 1998; McCoy & Strube 1999). The other type of
algorithm usually disregards linguistic context (see Krahnke & Theune
(1999) for an exception) and focuses, primarily, on the problem of finding
a set $L$ of properties which pick out a given unique individual from
among a set of distractors. The state of play in this area is discussed in
Dale & Reiter (1995). We will briefly summarize their algorithm.\footnote{The algorithm in Dale & Reiter (1995) is limited to descriptions of individuals. Generalizations covering reference to sets are starting to be forthcoming (Stone 1999; van Deen to appear) but the present paper focuses on individuals.}

The properties which form the basis of the algorithm are formalized as
pairs of the form \langle Attribute, Value \rangle. Attributes are ordered in a list $P$

reflect their degree of 'preference' (see below). Suppose $r$ is the individual
to be singled out, and $C$ is the set of elements from which $r$ is to be
Towards the Generation of Document-Dctic References

selected. Simplifying considerably, what happens is the following: the algorithm iterates through the list of attributes \( A \); for each attribute, it checks whether it has a value that rules out at least one member of \( C \) that has not already been ruled out; if so, the attribute is added to \( L \), with a suitable value. Members that are ruled out are removed from \( C \). The process of expanding \( L \) and contracting \( C \) continues until \( C = \{r\} \). The properties in \( L \) can be used by a linguistic realisation module to produce NPs such as ‘the white pill’, ‘the white pills for your heart condition’, etc. The algorithm is incremental — and is sometimes referred to as ‘the incremental algorithm’ — because properties are only added to \( L \); there is no backtracking. As a result, \( L \) may contain semantically redundant properties. According to Dale and Reiter, this has two advantages: it keeps the algorithm polynomial; and if the right preference order of attributes is chosen (reflected by the order of the attributes in \( P \)), the algorithm produces descriptions that are not unlike those uttered by humans.\(^7\)

It is worth elaborating on the function \texttt{FindBestValue} that chooses a value once an attribute has been selected. This function operates on a tree of values, where a daughter is more specific than (i.e., extensionally subsumed by) its parent. The function starts by considering the\(^8\) ‘Basic-level’ value of the attribute which, for simplicity, we can assume to be the most general value of the attribute (i.e., a value whose extension is a superset of all other values of the attribute)\(^9\) and moving to more specific values only if that value removes more distractors. This function balances a preference for more generic values with a preference for values that remove more distractors: the number of distractors is crucial, but genericity decides in case of a tie. The following schema summarizes the algorithm. (\( A \) is the list of attributes, \([V_i]\) is the denotation of \( V_i \))

\[
\begin{align*}
L &:= \emptyset \quad \{ \text{\( L \) is initialized to the empty set} \} \nonumber \\
\text{For each } A_i &\in A \text{ do } \nonumber \\
V_i &:= \text{FindBestValue}(r, A_i, \text{Basic Level}(A_i)) \\
\text{If } r e \{V_i\} &\subseteq [V_i] \{ \text{\( V_i \) would remove distractors but not } r \} \nonumber \\
\text{then do } &\quad \nonumber \\
L &:= L \cup \{A_i, V_i\} \quad \{ \text{Property of } V_i \text{ is added to } L \} 
\end{align*}
\]

\(^7\)See Krahmer & Theune [1999], however, for criticism.
\(^8\)The existence of a unique most general value is an assumption that is unproblematic from the point of view of the present paper (since the document as a whole plays this role (see section 1.4.2)).
\(^9\)In the work by Dale and Reiter, the basic level value is allowed to be a different one for different speakers, but this nuance is irrelevant for present purposes.
\[ C := C \cap [[V]] \] {} Elements outside \( V \) are removed from \( C \) 
If \( C = \{ r \} \) then Return \( L \) \{ Success \} 
Return \( Failure \) \{ All attributes have been tested, and still \( C \neq \{ r \} \) \}

It is important to be aware that success of an algorithm for the generation of referring expressions cannot always be guaranteed because there simply may not be enough properties in the domain model to characterize a given entity. But even if it is possible to characterize an object, there is no absolute guarantee that Dale and Reiter’s algorithm will find a characterization. A number of limitations of the algorithm are discussed in van Deenter (to appear), and remedies are proposed. A limitation that will be of specific relevance when DDX are generated concerns the situation that arises when a given attribute can have two values, say \( V_1 \) and \( V_2 \), which overlap in the sense that \( V_1 \not\subset V_2 \) and \( V_2 \not\subset V_1 \). The problems arising from these situations will be discussed in the section 1.4.1.

1.4 Generating DDX in a Document Authoring System

Suppose our document authoring system is of the kind where it is up to the system to decide whether a DDX is called for and, if so, which document entity it should refer to, and what form it should take. We will analyse the different tasks that face a system of this kind. Section 1.4.1 will focus on the first two questions, while section 1.4.2 addresses the third question, involving surface realisation. Section 1.4.3 describes architectural issues. Throughout the discussion, the distinction between object-level and meta-level DDX will be crucial.

1.4.1 Content Determination

With the term content determination we mean the process of determining, at a given point during the generation of a text, whether a DDX is called for and, if so, what document entity it should refer to. Whether the generated DDX is of object-level or meta-level, an important source of information is what we will call the document description base, which contains a record of the information conveyed by each relevant document entity.

The document description base

Documents contain document parts (e.g., sections, subsections, paragraphs, lists and pictures) each of which serves to express particular
Towards the Generation of Document-Dctic References / 9

chunks of information. When the document has been produced by an
NLG system, it is possible to find out what information is expressed by
each document part. In particular, it is not necessary to parse and in-
terpret the document, since the pairing between document parts and
domain entities is retrievable from the generation process. The data
structure in which this pairing is stored is called the document descrip-
tion base. In this section, we will sketch what type of information the
document description base might contain and how the document de-
scription base may be constructed.

The document description base records, for referable document entities,
what they are `about'. Aboutness, however, is not a trivial concept. For
example, if a section \( s \) makes a passing reference to a topic \( t \) that is
treated much more fully in section \( s' \), then the link between \( s \) and \( t \) is
less relevant than that between \( s' \) and \( t \). If, for example, an object-level
DDX is employed as a means of referring to a domain entity (`the side
effects discussed in section 5'), then this is unlikely to help the reader
much unless the section in question is associated strongly with the do-
main entity (i.e., if section 5 if the main repository of information about
the side effects in question). We will simplify by not distinguishing the
different ways in which a document entity may refer to a domain entity;
for example, it will not matter whether the document entity `discusses',
`introduces', `enumerates', or even `denies the existence of' the domain
entity: each of these cases is modelled through what we call the describe
relation.

First of all, we represent the relations between document entities and the
domain entities they realise in the form of `describe' relations. For ex-
ample, if a section \( s \) describes the side effects (sf1) of a medicine, we
represent this relation as `describe\( s, sf1 \)'. Consider, for example, the
following paragraph,

\[
\begin{align*}
\text{Estraderm MX patches contain a substance called oestradiol.}
\text{Oestradiol is a form of oestrogen and belongs to a group of medicines known as Hormone Replacement Therapy (HRT).}
\text{(ABPI 1997)}
\end{align*}
\]

Suppose this paragraph is generated from a structure containing the
following chunks of information:

1. Estraderm MX patches contain Oestradiol
2. Oestradiol is an oestrogen
3. Oestradiol is a Hormone Replacement Therapy
This suggests that, at a semantic level, Oestradiol has been described three times whereas each of Estraderm MX patches, oestrogen and Hormone Replacement Therapy has been described only once. This information may be simplified using a variety of methods. One extremely simple quantitative method, for example, might discard all objects except the one most often referred to in each given paragraph:

\[
\text{Document Description Base:} \\
\text{describe(s, Oestradiol)}
\]

In addition to quantitative methods of this kind, which are familiar from work on text summarization, a more direct approach is possible in those systems whose input draws a distinction between central and peripheral information. For example, if the input to the generator uses rhetorical structures (Mann & Thompson 1987), which organize texts in hierarchical structures most nodes of which have one nucleus and one satellite, then this can be taken into account, for example, by allowing the document description base to limit itself to domain entities described by the most nuclear constituent of any given paragraph. Applied to the example, it seems clear that the first sentence corresponds with the most nuclear information expressed in the paragraph, and hence this method suggests that both Estraderm and Oestradiol deserve a place in the document description base:

\[
\text{Document Description Base:} \\
\text{describe(s, Oestradiol)} \\
\text{describe(s, Estraderm)}
\]

Let us assume that a document description base can be built up during generation and move on to the question of how it can be exploited during content determination. Both main types of DDX will be discussed but the main focus of our proposal will involve object-level DDX.

**Meta-level Document Deixis**

A recent paper sketches the outlines of a system for the authoring of newspapers, focusing on what happens when articles ‘go over the page’ (Bateman et al. 2000). In newspapers, this does not always mean that the remainder of the paper follows on the next page, which is why a highly specific document-deictic reference is called for to facilitate reading, which says ‘Continued on page x’ or something equivalent to it.\(^{10}\) The authors analyse this as a mismatch between rhetorical structure

\(^{10}\) Note that pointers of this kind can even be disambiguating if there happen to be several pages that can be read as the continuation of the original page.
Towards the Generation of Document-Dctic References / 11

(i.e., the structure of the document in terms of rhetorical relations cf. Mann & Thompson 1987) and layout structure, and sketch how such mismatches give rise to 'navigational' pointers.

From our current perspective, the following observations may be made. Firstly, the navigational pointers in question are meta-level DDX aimed at making the newspaper more accessible without changing its truth-conditional content. Secondly, the pointers in question do not require the generator to take into account what the remainder of the article is about: it suffices to know where it is located. But thirdly, Bateman et al. argue that their application allows them to make an important simplification, namely to assume that the location referent of the DDX is always given. Some newspapers, for example, invariably put the remainder of any front-page article on the back page, and this makes the generation of DDX considerably easier since it allows them to disregard the architectural problems that are a consequence of the unpredictability of physical realisation (section 1.4.3).

Unfortunately, meta-level DDX are not always so easily finessed. They appear to be motivated by a wide variety of reasons, of which clashes between rhetorical and layout structure are only one. Moreover, the references often do require the generator to know what the document entity in question is 'about'. The point may be illustrated by comparing

5a. The final section contains a discussion of our experimental results.
5b. Section 5 compares our results with those of Simon.

In many genres, where experimental results are always discussed in the final section, (5a) may be generated using canned text. This, however, is different for (5b), in the absence of similar conventions concerning the place where specific papers are discussed. The generation of (5b) would require the generator to know which section has described architectural consequences, which is where the document description base comes in.

Of particular relevance in the domain of patient information leaflets is the case where meta-level DDX are generated because semantically or rhetorically related items of information are kept separate in the document. One type of case is especially frequent in instructional domains, of which patient information leaflets are an example. To illustrate, suppose a part of the document expresses the information that an effect (‘Goal’) may be achieved via some procedure (‘Method’):

Goal: Clean your inhaler
Method: Step 1: Take the inhaler apart  
   Step 2: Rinse its components in cold water  
   Step 3: Let the components dry in the air  
   Step 4: Reassemble the inhaler

Suppose, furthermore, that this relation between Goal and Method is expressed in section \( x \), while some other section \( y \) discusses the Goal without specifying the Method. (For example, section \( y \) explains that the inhaler must always be cleaned before use, and so on.) Then it is highly natural for this instruction to be accompanied by a reference to section \( x \) (‘see section \( x' \)'), since it may be instructive to read section \( x \).

Similar triggers for meta-level DDX include the existence of a gap in the user’s vocabulary (Feiner & McKeown 1990). For example, if the generator makes use of a term which is unknown to the reader, and which is discussed in another part of the document defined by a document entity, a reference to this other document part may be added:

‘(...) reset your insulin pen to * (see F)”

where the referred section ‘F’ describes how to reset an insulin pen in this way. References of the kinds discussed here are especially useful if \( x \) and \( y \) are not on the same page and

(a) \( y \) occurs in a section of the document that is likely to be skipped, or  
(b) the document does not have a fixed left-to-right reading order, or  
(c) \( x \) precedes \( y \) by a large number of pages.

Differences in style and genre exist, even within a given application domain. In some patient information leaflets, for example, such references are produced whenever the unknown term is used, (sometimes several times within the same section), whereas others use them on first mention only; yet others do not use them at all. Given the function that meta-level DDX have – namely, to make the document more easily accessible – such diversity is to be expected, since different authors can entertain different assumptions about their readership.

**Object-level Document Deixis**

**Object-level DDX versus anaphora.** We have seen that there is more to object-level DDX than reference but if we limit ourselves to restrictive uses of document-deictic information then one can view object-level DDX as abbreviation devices not unlike discourse anaphora: they are em-
employed as a means of avoiding the repetition of a long description which has already been introduced in the discourse. For example, given the antecedent ‘a medicine for yellow fever type B’, a subsequent reference to the same domain entity can be realised via object-level DDX (e.g., ‘the medicine mentioned in chapter 1’).

Interestingly, object-level DDX as such appear to be immune from many of the known constraints on discourse anaphora. For example, if a list of pathological symptoms has occurred somewhere in the document but no name has been attached to them, recency constraints (Ariel 1990) prohibit the use of anaphora (e.g., ‘they’, ‘the symptoms’, or ‘these symptoms’) if these symptoms become relevant at a much later place in the document. In such a case, a DDX can be used as an indirect way of referring to the symptoms, via the document part that describes them: e.g., ‘the symptoms described in Section 1.1’. Note that the specific form of the DDX can be constrained by various factors. A DDX of the form ‘the symptoms mentioned above’, or ‘the just-mentioned symptoms’ for example, is only felicitous if its ‘antecedent’ has occurred fairly recently.

In what follows we will assume that the choice between an anaphoric reference and a full, Dale and Reiter-type description has already been made (McCoy & Strube 1999; Krahmer & Theune 1999) and explore how a full description involving document-deictic properties may be made.

**Object-level DDX as Dale & Reiter-type descriptions.** The semantic function of a restrictive object-level DDX does not differ from that of ‘ordinary’ referring expressions, namely to single out an object. One obvious approach is to assume that object-level document deictic expressions are constructed in the same way as ordinary referring expressions (see section 1.3.2). Document entities can be viewed as document deictic properties of some of the domain entities they describe, e.g., a domain entity realised as a picture may have, besides all its domain properties, the DDX property ‘being described by’ the picture, and this property can be used in referring expressions just like any domain property. Thus, we will assume there to be an attribute DescribedBy which can have any document entity as its value. The relation with the describe relation in the document description base is obvious:

The domain entity dom-ent has the Value doc-ent for the Attribute DescribedBy if the relation describes(doc-ent,dom-ent) holds in the document description base.

Continuing to assume that the parts of a document are sections ordered in a tree, the permitted values for a given document (arranged in the
form of a tree according to their place in the document) may, for example, be

```
  document
  /     \   
sec 1  sec 2
  /   \   /
sec 1.1 sec 1.2 sec 2.1 sec 2.2
```

('sec 1' is short for 'being described by section 1, and so on.) Properties like 'x is described by section 2.1' (where x is a domain entity) will be called document-related properties, as opposed to the ordinary, domain-related properties. To appreciate the analogy, suppose the document deals with medicines, for example, then the description 'the tablet mentioned in section 1.2' may distinguish one tablet from all others, analogous to an 'ordinary' description like 'the white tablet'. As a result, the improved generation algorithm could produce both ordinary descriptions and ones that combine ordinary descriptions with a property consisting of the attribute DescribedBy and one of its permitted values.

The problem of overlapping values. There is one problem, however: the tree of values just described instantiates the problem of overlapping values mentioned in section 1.3.2 and, consequently, the algorithm proposed by Dale and Reiter will sometimes fail to find a description even if one exists.\footnote{In fact, the problem of overlapping values first occurred to us when studying document entity-valued Attributes in the context of document deixis generation.} Transposing an example in van Deemter (to appear) into the present, DDX-related, setting, suppose sections 1 and 2 describe some of the same medicines, while a is the target object:

```
  DescribedBy: section 1 ({a, b, e}), section 2 ({a, c, d, f}),
  Colour: Brown ({a, b}), Yellow ({c, d})
```

The Value 'section 1' (being the Best Value of DescribedBy, since it removes more distractors than 'section 2') is chosen first, reducing the initial set C to {a, b, e}. Now, the algorithm is doomed to end in Failure, since the different Values of Colour are unable to remove the unwanted b without sacrificing a. None of this can be corrected, since the algorithm does not use backtracking. Note that a uniquely identifying description of a would have been possible if 'section 2' had been chosen instead of 'section 1', leading to a description like 'the brown medicine described in section 2'. In other words, the algorithm fails in a situation where Success was perfectly achievable.
A number of possible solutions to this problem have been proposed in van Deemter (to appear), of which the following seems most appropriate under the present circumstances: a limited kind of backtracking is added to the algorithm, which “remembers” where the algorithm has encountered overlapping Values and, when it results in Failure, goes back to the last-encountered situation where it has made a choice between overlapping Values; if this does not lead to Success, the algorithm tracks back to the previous choice situation, and so on until no more choice situations are left (Failure) or a distinguishing description has been reached. As a result of this modification, the incremental algorithm is guaranteed to find a distinguishing description (possibly involving document-related properties) if one exists.

The choice of basic-level Values. It is interesting to explore the consequences of this approach, which uses the algorithm proposed by Dale and Reiter more or less unamended. In particular, the approach predicts (through the FindBestValue attribute) that whenever a section and one of its subsections remove the same number of distractors, the larger section will be chosen, because ‘being described by section 1’ is more general than ‘being described by section 1.2’, for example. In many cases, the strategy of favouring less specific properties may be correct, but in the case of document-related properties it would be misguided. To see why, imagine that the object-level document deictic expression serves to identify a particular side effect of a medicine. It is unlikely that the reader keeps in mind exactly which sections discuss which side effects. It is much likelier that she will use the DDX to look up where in the leaflet the side effect is described. Clearly, this is easier when the most specific (sub)section is mentioned, rather than the least specific section.12

This means that an alternative approach is called for, which favours the most specific values of the ‘described by’ attribute13. This can be done in two different ways: one possibility is to adapt the algorithm to favour the most specific values of this particular attribute (while still favouring the least specific values of all other attributes). A more uniform account is

---

12A preference for more specific properties over less specific ones is also likely in other spatially-oriented descriptions such as pointing out a region on a map. Note that the issue of what property to use [i.e., which section or part of the map to refer to] differs from the question of how to refer to it [e.g., by saying ‘the town of San Jose in California’, rather than just ‘San Jose’].

13It appears that some document genres may use a preferred referent type [e.g., a given subsection level] instead of simply referring to the most specific subsection. Further corpus investigation will be required to test this hypothesis.
obtained, however, if the DescribedBy attribute is modelled as having the most specific subsections (i.e., the leaves of the document tree) as its only values, while the algorithm for selecting values is left unchanged:

\[
\text{DescribedBy:} \\
\quad \text{sec 1.1 (\{a, b\}), sec 1.2 (\{a, e\}),} \\
\quad \text{sec 2.1 (\{a\}), sec 2.2 (\{c, d, f\})}
\]

With this second modification in place, the algorithm will generate references that are easier to interpret than would otherwise have been the case.

**When to use document-related properties?** It will be convenient to think of this variant of the algorithm (let us call it D&R\textsubscript{d,e}) as separate from one that makes use of domain-related properties only (D&R). Whether to choose the first (and obtain an NP containing a DDX) or the latter (and obtain an ordinary description) is a choice that can be made in different ways, depending on application domain and document genre.

In some genres, the decision for the generation of object-level DDX is linked directly to the layout decisions made during document generation. Scientific papers on computational linguistics, for instance, make use of enumerated linguistic examples to facilitate reference (e.g., ‘see example 17’), and when this happens, object-level DDX appears to be the method of choice. This observation may be incorporated into the Dale & Reiter algorithm by assigning different priorities to document-related properties depending on layout: document-related Attributes that are made salient as a result of layout decisions (e.g., numbering the example sentences) receive higher priority than any other property. The algorithm itself can be used unamended.

In other cases, however, document-related properties are probably better viewed as a repair strategy: ordinary descriptions (i.e., those not using any document-related properties) are the method of choice. In these cases, document-related properties are used to avoid overly long, or otherwise awkward descriptions, reflecting a Gricean-style brevity maxim (Grice 1975). Suppose, for example, the generator has to refer to a particular side effect, discussed in the body of a leaflet (i.e., not marked by numbers, headers, or other layout devices). Suppose, furthermore, that the description produced by D&R – i.e., the description that does not use document-related properties – makes use of the following (rather long) list of properties:
Towards the Generation of Document-Dilictic References / 17

\[ L = \{ \text{associated with medicine 1, associated with medicine 2, long-term} \} \]

whereas the side effect in question is also the only one to be described in, say, section 1. In a case like this, the sheer length of \( L \) might be sufficient reason to use the document-related leading to a description like ‘The side effects described in section 1’, rather than ‘The long-term side effects that are associated with both medicine 1 and medicine 2’. This suggests an algorithm of the following form. (The function ‘Output’ delivers the output of an algorithm; ‘Length’ counts the number of properties in a list.)

\[
L := \text{Output(D&R)}
\]

If \( \text{Length}(L) > x \) Then

\[
L' := \text{Output(D&R}_{d,\mu})
\]

If \( \text{Length}(L') \leq x \) Then \( L := L' \)

The precise way in which ‘Length’ measures the length of a description, and choice of an appropriate value for \( x \), have yet to be determined. We conclude this section by listing some remaining problems which have to be addressed:

1. The choice for a particular referent may be influenced by the distance between the DBX and its referent, or by a particular preference order among referent types.

2. There are, of course, issues of aggregation to be taken into account. For example, if a particular type of domain entity is described through all - or even most - subsections of a document section, we may either list the subsections (e.g., ‘the symptoms described in section 3.1 and 3.2’) or only the main section as a whole (e.g., ‘the symptoms described in section 3’).

3. The algorithms described so far work as if document-related properties were always completely precise. A more sophisticated version of the algorithm would take vague references like ‘above’, ‘below’, ‘earlier’, etc. into account.

1.4.2 Surface realisation

In this section, we turn to issues of surface realisation, focussing on NPs referring to document entities.\(^{14}\) As in the case of ordinary refer-

\(^{14}\)Note that object-level non-involving referring expressions at two different levels: if the non has the syntactic structure \([\ldots \{& N P_1 \ldots \} \& N P_2 \ldots \] \( N P_1 \) then both \( N P_1 \) and \( N P_2 \) are referring expressions whose generation gives rise to similar (but distinct) questions. Generation of \( N P_1 \) was the topic of section 1.4.1; generation of \( N P_2 \) is presently under discussion. For a discussion of issues of surface realisation of ordinary NPs referring to domain entities, see Shaw & Hatzivassiloglou (1999).
ring expressions, linguistic realisation of DIX poses nontrivial problems. Consider a DIX Re that refers to the document entity Rt. The surface realisation of Re can vary between, for example,

'this section',
'section 2',
'the section printed in boldface',
'the section starting on page 5', etc.

We have seen that realisation is, to some extent, genre-dependent. For example, in some genres, pictures are most often referred to by their spatial position in the page (e.g., 'the left-most picture'), but in scientific papers they tend to be referred to via numbers (e.g., 'picture 3.2'). In the discussion that follows, we will concentrate on references to document entities that are constituents of a hierarchically-organised document such as sections, subsections, pictures, items, etc. References to parts of the document that lie outside such a hierarchy (e.g., pages), although very relevant in some genres (magazines, newspapers, etc.), are virtually non-existent in the corpora we have studied (i.e., in patient information leaflets and computational linguistics papers) as, of course, they are in most web-based documents. The following is an example of a hierarchical document (doc) made up of sections (s1 and s2) and enumerated items (A, B, and A).

```
  doc
     / \
    s1  s2
     / \ 1
    A   B A
```

DIX is a special case of reference, so one might expect to see another application of a Dale and Reiter-style algorithm, perhaps with appropriate modifications (cf. sections 1.3.2 and 1.4.1), applied to a domain of discourse whose elements are document entities and whose properties are document-related. There are reasons, however, to use a different approach.

Firstly, we will simplify by disregarding domain properties (e.g. 'the section describing side effects'): our document entities will be characterized by layout properties alone. Layout properties are of the form ⟨Attribute, Value⟩, where Attribute is a layout attribute (e.g., section number, section title, item number, pictures number, etc.) and Value an appropriate value. Such properties are often written by juxtaposing the Attribute and the Value (e.g., 'Part A', 'picture 5'). But, if only layout properties are used then many document entities cannot be identified
Towards the Generation of Document-Deductive References

uniquely. For example, if pictures are numbered per section, then the
document may contain several pictures with the same number. Conse-
quently, we need an approach that takes the hierarchical structure of the
document into account, allowing us to distinguish between, for example,
picture 5 of section 1 and picture 5 of section 2.

Secondly, and perhaps more interestingly, finding a distinguishing de-
scription is sometimes not enough. This issue at hand is akin to what
Dale and Reiter called Navigational information (Dale & Reiter 1995).
Navigational information is often disregarded, but in the present setting,
where we are dealing with domains that are potentially huge, it becomes
vital.\footnote{Compare our discussion of finding the ‘best value’ of a document-related property in section 1.4.1.} a felicitous \textit{dx} also has to provide information on how to locate the referent in the document taking into account the reader’s knowledge about the document structure. For this reason, our algorithm will sometimes make use of structural information (relating to the hierarchical structure of the document) even in cases where this is not needed for unique identification of a document entity. For example, suppose an item of the document has been assigned the label ‘B’. The \textit{Re} ‘item B’ on its own may be infelicitous if the distance between \textit{Re} and \textit{Rt} is large, even if there is only one item B in the entire document, because the description offers little help in finding the intended referent, which could be located literally anywhere in the document. Let us sketch how this idea is formalized.

We will assume that the generator ‘knows’, for each layout attribute what its ‘scope’ on the document is. For example, it can look up whether picture numbers are assigned at the level of the document itself, at one of its main sections, etc. If an attribute is assigned at a document entity $x$ then its values are uniquely distinguishing within $x$ (and possibly in a larger part of the document). In our example, the scope of the layout attribute ‘item label’, for document entity marked $Rt$, is the subtree rooted in $s1$, since items are assigned labels A, B, etc. \textit{per section}. The same is true for the sibling of $Rt$.

Let $d$ be a document tree containing a node $n$, and let $A$ be a
layout Attribute associated with $n$. Then the \textit{scope} of $A$ with
respect to $d$ and $n$ is that ancestor of $n$ where the values of
$A$ are assigned.

The notion of scope is utilised by the algorithm as follows (See Paraboni
(2000) for details):
1. Select the first document entity to be referred to and call it $R_t$.
2. Generate a description of $R_t$ making use of an attribute $A$ that distinguishes $R_t$ from each of its sibling nodes.
3. If the scope of $A$ includes $Re$ then stop.
4. Otherwise, select an ancestor node of $R_t$ (typically, its parent node), call this node $R_t$ and go back to step 2.

Thus, references are added to the resulting description in hierarchical order, from lowest to highest level. The crucial part of the algorithm is step 3, which states that the reference is complete when the scope subtree of the latest reference includes $Re$. Let us illustrate the procedure, once again assuming a document structure made up of sections $s1$ and $s2$ containing enumerated items $A$ and $B$ (as depicted above). Now a reference in $s2$ to item $B$ in the previous section has to include information about the section which contains the referent, even though “item $B$” is a distinguishing description within the entire document:

```
  doc
    / \
   s1  s2 (Re)  ===>  "item B of section 1"
    / \ |  
   A   B A (Rt)
```

One way to see why hierarchical information is required for the description of $R_t$ is that, although this is not necessary for the unique identification of $R_t$ itself, hierarchical information would be necessary to identify $R_t$'s sibling ('part A of section 1').

This concludes our sketch of the algorithm. Note that the account can be extended to many other kinds of attributes, such as spatial relations (e.g., “above”, “below” etc.) which cannot, however, be discussed here.

### 1.4.3 Architectural Issues

The issues discussed so far, although most crucial in an NLG context are also at least indirectly relevant for interpretation. If a text says 'item $B$ of the list in section 4.3.2', for example, we normally infer that 4.3.2 is not the present section.\(^1\) Something similar is true to the issues discussed in what follows: they are most directly relevant to NLG, but there are obvious ramifications for incremental interpretation.\(^2\)

---

\(^1\)For a discussion of the connection between Gricean maxims and the interpretation of referring expressions, see Dale & Reiter (1995).

\(^2\)Some of the issues discussed here are reminiscent of problems with *kalaphoric* (i.e., forward-anaphoric) references, see for example van Deemter (1990).
One potential issue in this area concerns meta-level DDX, which pose the question where they come from, given that they do not add to the truthconditional content of the text (section 1.2). If, for example, the input to the generator is based on Rhetorical Structure Theory, then the most obvious approach is to let Rhetorical Structure distinguish between a document with and without a given DDX. (In the first case, the content of the sentence containing the DDX might, for example, stand in an ELABORATE relation to that of some other text span.) But if this is the case, then the generator’s autonomous decision to generate a DDX involves changing its own input. The alternative is to make DDX a lower-level decision which does not affect Rhetorical Structure, in which case there appear to be no architectural consequences.

A more general problem, which can arise in connection with both types of DDX, concerns the order in which the several subtasks of the generation process are to be carried out. For example, in the pipelined NLG architecture that is often applied in NLG (Reiter 1994), linguistic realisation has to wait until late in the generation process. Normally – unless a DDX has somehow been anticipated – this will mean that the document description base (cf. section 1.4.1) is not available when DDX are generated. More precisely, if a document entity has not been ‘created’ (or specified in sufficient detail) by the time the NLG system has to refer to it, the document description base cannot be produced and, therefore, neither meta-level nor object-level DDX can be generated. For example,

1. **Dependency between surface realisation and DDX:** the realisation of references to the text (e.g., ‘Read the section entitled “how to use your medicine”’) may not be possible if the referred text has not been realised yet. For example, if realisations operates from left to right, then forward-referencing DDX (where Rt occurs to the right of Re, cf. van Deemter (1990), become problematic.

2. **Dependency between layout and DDX:** the realisation of references making use of layout properties (e.g., ‘Read section B’) may not be possible if the referred layout property has not been generated yet.

3. **Dependency between page model and DDX:** since the text of the referring expression is part of the same document which contains the referent, the surface realisation of a DDX containing a page number may move the intended referent to the next page of the document, causing the DDX to point to the wrong place.

It is interesting to see how similar problems in the area of multimedia document generation have been tackled in McKeown et al. (1992).
Among other things, the COMET system generates descriptions like 'the old holding battery shown in a separate picture', where the problem is to let the Referring Expressions Generation module know that the battery happens to be shown in a separate picture, which a decision by the Realisation module. McKeown et al. tackled the problem by using logical forms as the basis of the different components of a multimedia generator, the subformulas of which are annotated with information about media choice. By using the logical forms in this way, they play the role of a BLACKBOARD onto which the multimedia generator can write what combination of media is employed to express certain things, and from which the generator can read information of the same type. In the case of COMET, media choice appears to be the only kind of annotation employed, but this idea may be generalized to include the fact that something is expressed in the second picture occurring in section 3, for example. A solution along these lines is compatible with the use of a document description base, but it would be a radical departure from pipelined architectures and a stark generalization of the work on COMET as well. The alternative would be a two-phase generator whose first phase generates a document containing preliminary DDX expressions; in a revision phase, the DDX are adjusted so as to correspond to the actual realisation of the document.18

We have seen that document-deictic expressions can be dependent on subtle layout features of the document. Accordingly, some layout features of the document seem to be specifically chosen to facilitate DDX. A case in point is the use of numbered examples in scientific papers, which allow the use of references like 'Example 1.2.1. shows that...' etc. But, of course, authors may change their minds. Ideally, therefore, a document authoring system allows authors to jump back and forth through a document, for example, adding layout features to facilitate document-deictic reference. Issues of this kind do not concern the architecture of an NLG system proper, but they are crucial to the architecture of a document authoring system (Power & Scott 1998).

1.5 Conclusion
Having introduced the phenomenon of document deixis and having discussed briefly how it differs from other types of context-sensitive referring expressions, such as anaphora and discourse deixis, we have explored how the two main kinds of document deictic expression, object-level and meta-level DDX, may be produced by a Natural Language Generation

---

18 Compare Inui et al. (1992) for a different motivation for a two-phase generation architecture.
program.

Although meta-level and object-level \( D\&X \) raise partly different issues for a generator, there are also important commonalities between them. Both, for example, can make use of what we have called a Document Description base, an information source which is generated as a side effect of the generated document (Section 2.1), and both pose some difficult problems for traditional NLG architectures, mainly with respect to the dependency between document content and layout (Section 4.4).

We have devoted most of our attention in this paper to object-level \( D\&X \), which make use of a combination of domain-related and document-related properties to characterize a domain entity. The fact that the document-related properties of which object-level \( D\&X \) make use (e.g., ‘being described in section 3.2’) refer to the inhabitants of a secondary, potentially huge, and spatially structured domain (namely the document itself) makes the generation of object-level \( D\&X \) an intriguing area for investigation of various aspects of reference.

Firstly, we have argued that the spatial nature of this secondary domain poses a challenge to traditional referring expressions generation algorithms such as presented in Dale & Reiter (1995) because it highlights the problems that arise when the Values of an Attribute overlap, and we have shown how this limitation may be removed.

Secondly, we have argued that the hierarchical structure of the domain (i.e., the document itself), tends to make the problem of finding a uniquely distinguishing description relatively easy, whereas the potentially great size of the domain tends to make navigational issues of paramount importance. This issues has come up in two different guises.

a. The usual assumption, that general properties are better than specific ones (modulo the number of distractors removed), is no longer always valid. For example, if an object was described in section 2 only once, namely in section 2.3.1, then it is more helpful to refer to it as being ‘described in section 2.3.1’ than as ‘described in section 2’.

b. It is often necessary to include additional properties into a description, over and above the ones that uniquely distinguish a document entity. For example, even when a document contains only one item labelled as ‘B’, it is normally necessary when referring to the item, to add in which part of the document (e.g., in which section) the item occurs.
Finally, we have discussed a number of ways in which the generation of document deixis poses an architectural challenge to existing NLG (and document authoring) systems. Some of these challenge the assumption that NLG systems follow a pipeline model (Reiter 1994); others challenge the assumption that generation of a text operates basically from left to right. Space has not allowed us to discuss these issues in great detail, and their resolution must partly be left to future research.

Efforts at implementing the ideas in this paper are underway. An algorithm for generating references to document parts (see section 1.4.2), for example – which is one of the key mechanisms common to the generation of both object-level and meta-level DBX – has been described in Paraboni (2000)\(^{10}\)

1.6 Acknowledgements

We would like to thank Richard Power and John Bateman for their comments and advice. 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.

References


Robert Dale and Ehud Reiter: 1995, Computational interpretations of the

\(^{10}\)The work in Paraboni (2000) is based on an analysis of the Patient Information Leaflets in ABPI (1997), a corpus that is used in a number of current NLG projects, including GNOME. (Generation of Nominal Expressions, funded by the UK EPSRC under grant reference GR/L51126.)
Towards the Generation of Document-Dctic References


Ehud Reiter: 1994, Has a consensus NL generation architecture appeared and is it psycholinguistically plausible? in: proceedings of the 7th Interna-
tional Natural Language Generation Workshop, pp.163-170, Maine U.S.A.


