Generating References to Parts of Recursively Structured Objects

Helmut Horacek
Universität des Saarlandes
FB 6.2 Informatik
horacek@cs.uni-sb.de

Abstract

Algorithms that generate expressions to identify a referent are mostly tailored towards objects which are in some sense conceived as holistic entities, describing them in terms of their properties and relations to other objects. This approach may prove not fully adequate when referring to components of structured objects, specifically for abstract objects in formal domains, where scope and relative positions are essential features. In this paper, we adapt the standard Dale and Reiter algorithm to specifics of such references as observed in a corpus about mathematical proofs. Extensions incorporated include an incremental specialization of property values for metonymic references, local and global positions reflecting group formations and implicature-based scope preferences to justify unique identification of the intended referent. The approach is primarily relevant for domains where abstract formal objects are prominent, but some of its features are also useful to extend the expressive repertoire of reference generation algorithms in other domains.

1 Introduction

Over the past two decades, a number of algorithms for generating referring expressions have been proposed. Almost all of these algorithms conceive objects in some sense as holistic entities, describing them in terms of their properties and relations to other objects, but not treating components of an object as objects in their own rights. This approach may yield inadequate results for references to components of recursively structured objects. Consider, for instance, a Rubic's cube where one side is currently visible, and reference is intended to a square consisting of the visible squares of four white subcubes, which are the only white elements on the visible side. The best way to refer to this composed structure is the concise "the white square", which exploits a preference for maximum scope objects, typical for such recursive structures. However, most reference generation algorithms would attempt to disambiguate the intended referent from its four components, producing an unnecessarily long expression, such as "the big white square" or "the white square which is composed of four squares". These expressions are not really bad, especially the first one, but things might turn out really awkward for more complex structural compositions, where the maximum scope preference often allows the identification in a surprisingly concise form.

In this paper, we address this problem by examining referring expressions produced by humans in domains with recursively structured objects playing a prominent role. Specifically, we have studied referring expressions in a corpus of simulated human-computer dialogs about tutoring mathematical problem-solving (Wolska et al. 2004, with recent additions in this paper). We express the criteria and preferences observed in a way compatible with the incremental reference generation algorithm of Dale and Reiter (1995), and we extend their algorithm by adapting the property selection and discrimination testing criteria accordingly.

This paper is organized as follows. First, we motivate our approach. Then we describe our corpus and the relevant phenomena observed in it. Next, we present extensions to the incremental algorithm that allow the generation of this kind of referring expressions. Finally, we illustrate how some examples from the corpus are handled and discuss our achievements.

2 Previous Work

Within this paper, we adopt Dale's terminology (1988). A *referential description* (Donellan 1966) serves the purpose of letting the hearer or reader identify a particular object or set of objects in a situation. Referring expressions to be generated are required to be *distinguishing descriptions*, that is, descriptions of the entities being referred to, but not to any other object in the *context set.* A context set is defined as the set of the entities the addressee is currently assumed to be attending to – this is similar to the concept of focus spaces of the discourse focus stack in Grosz' & Sidner's (1986) theory of discourse structure. Moreover, the *contrast set* (the set of *potential distractors* (McDonald 1981)) is defined to entail all elements of the *context set* except the *intended referents*.

Generating referring expressions is pursued since the eighties (e.g., (Appelt 1985), among several others). Subsequent years were characterized by a debate about computational efficiency versus minimality of the elements appearing in the resulting referring expression (Dale 1988, Reiter 1990, and several others). In the mid-nineties, this debate seemed to be settled in favor of the incremental approach (Dale and Reiter 1995) – motivated by results of psychological experiments (e.g., Levelt 1989), certain non-minimal expressions are tolerated in favor of adopting the fast strategy of incrementally selecting ambiguity-reducing attributes from a domain-dependent preference list. Complementary activities include the generation of vague descriptions (van Deemter, 2000) and extensions to multimodal expressions (Van der Sluis 2005). Recently, algorithms have also been developed to the identification of sets of objects rather than individuals (Bateman 1999, Stone 2000, Krahmer, v. Erk, and Verweg 2001), and the repertoire of descriptions has been extended to boolean combinations of attributes, including negations (van Deemter 2002). To avoid the generation of redundant descriptions what incremental approaches typically do, Gardent (2002) and Horacek (2003) proposed exhaustive resp. best-first searches.

All these procedures more or less share the design of the knowledge base which bears influence on the descriptor selection. Objects are conceived as atomic entities, which can be described in terms of sets of attributes and relations to other objects. In such a setting, a structured object can be represented, among others, by a set of relations to its components, which are themselves conceived as objects. An exception to this method is the work by Paraboni and van Deemter (2002) who use hierarchical object representations to refer to parts of a book (figures, sections., etc.). Reference to such a component is made identifiable by iteratively adding a description of embedding structures until obtaining uniqueness. There are, however, no approaches addressing identification of objects or their components when the structures in these objects are of a *recursive* nature. Objects of this kind are mostly abstract ones, such as formulas, but also some sorts of geometric objects. Typical applications where such objects are prominent include scientific-technical documentation and tutoring systems. As we will see in the next section, naturally observed references to such objects have a number of particularities which are not addressed by existing generation algorithms.

3 A Corpus with References to Formulas

In this paper, we analyze some phenomena in the context of references to mathematical formulas and their components, as observed in a corpus on simulated man-machine tutoring dialogs (Wolska et al., 2004). These dialogs constitute the result of Wizard-of-Oz experiments in teaching students mathematical theorem proving in naive set theory resp. mathematical relations. In these experiments, a human wizard took the role of the tutor, with constraints on tutoring strategy and on use of natural language, although the constraints on natural language use were relaxed to encourage natural behavior on behalf of the student.

In the corpus obtained this way, a number of quite particular expressions referring to components of recursively structured objects – the formulas – showed up. Consequently, it is our goal to automate the production of these kinds of referring expressions in a more elaborate version of the simulated tutoring system, with full-fledged natural language generation.

Representative examples originating from our corpus appear in Figure 1. Each example consists of two parts: 1. a student utterance, mostly a formula, labeled by (#a), which is the context for interpreting subsequent referring expressions, the intended referent appearing in

1. Reference to the typographic order

(1a) $(R\circ S)^{-1} = \mathbf{\{(x,y) \mid (y,x) \in R\circ S\} = \{(x,y) \mid \exists z\,(z \in M \wedge (x,z) \in R^{-1} \wedge (z,y) \in S^{-1})\}} = R^{-1}\circ S^{-1}$

(1b) Das geht ein wenig schnell. Woher nehmen Sie *die zweite Gleichheit*?
(That was a little too fast. How did you find *the second equality*?)

(2a) Nach 9 $\Rightarrow ((y,z) \in R \wedge (z,\mathbf{y}) \in S)$

(2b) Fast korrekt. *Das zweite Vorkommen von y* muß durch x ersetzt werden.
Almost correct. *The second occurrence of y* must be replaced by x.

(3a) $(R \cup S)\circ T$ ist dann $\{(x,y) \mid \exists z\,(z \in M \wedge ((x,y) \in R \vee (x,y) \in S) \wedge \mathbf{(y,z) \in T)\}}$

(3b) Nicht korrekt. Vermutlich liegt der Fehler *nach der letzten 'und'-Verknüpfung*
Not correct. The mistake is probably located *after the last 'and'-operation*

2. Reference by exploiting default scope and metonymic relations

(4a) $(R\circ S)^{-1} = \mathbf{\{(x,y) \mid \exists z\,(z \in M \wedge (y,z) \in R^{-1} \wedge (z,x) \in S^{-1})\}} \supseteq S^{-1}\circ R^{-1}$

(4b) Nein, das ist nicht richtig! Vergleichen Sie *den zweiten Term* mit Ihrer vorhergehenden Aussage!
No, this is not correct! Compare *the second term* with your previous assertion!

(5a) $\{(x,y) \mid (y,x) \in (R\circ S)\} = \mathbf{\{(x,y) \mid (x,y) \in \{(a,b) \mid \exists z\,(z \in M) \wedge (a,z) \in R \wedge (z,b) \in S\}\}}$

(5b) Das stimmt so nicht. Die *rechte Seite* wäre identisch mit R∘S.
This is not correct. The *right side* would be identical to R∘S.

(6a) $\{(x,y) \mid \exists z\,(z \in M) \wedge ((x,z) \in R \vee (x,z) \in S) \wedge (z,y) \in S\} =$
$\{(x,y) \mid \exists z\,(z \in M) \wedge (z,y) \in S \wedge ((x,z) \in R \vee (x,z) \in S)\} \Leftrightarrow \mathbf{((y,z) \in S \wedge (z,y) \in S)}$

(6b) Auf der *rechten Seite* ist z nicht spezifiziert
On the *right side*, z is not specified

(7a) $\mathbf{\{(x,y) \mid \exists z\,(z \in M) \wedge ((x,z) \in R \vee (x,z) \in S) \wedge (z,y) \in S\}} = \{(x,y) \mid \exists z\,(z \in M) \wedge$
$(z,y) \in S \wedge ((x,z) \in R \vee (x,z) \in S)\} \Leftrightarrow \mathbf{\exists z\,(z \in M \wedge ((y,z) \in S \wedge (z,y) \in S))}$

(7b) Diese Aussagen scheinen nicht gleichwertig zu sein. Ein z, das die Bedingung der *rechten Aussage* erfüllt, muß nicht die Bedingung der *linken Menge* erfüllen.
These assertions do not seem to be of equal range. A z which fulfills the condition of the *right assertion* does not necessarily fulfill the condition of the *left set*.

3. Reference by exploiting default scope for building groups of objects

(8a) $K((A \cup B) \cap (C \cup D)) = \mathbf{K(A \cup B) \cup K(C \cup D)}$

(8b) De Morgan Regel 2 auf *beide Komplemente* angewendet.
De Morgan Rule 2, applied to *both complements*.

(9a) $\mathbf{(T^{-1}\circ S^{-1})^{-1} \cup (T^{-1}\circ R^{-1})^{-1}} = \{(x,y) \mid (y,x) \in (T^{-1}\circ S^{-1}) \wedge (y,x) \in (T^{-1}\circ R^{-1})\}$

(9b) Dies würde dem Schnitt *der beiden Mengen* entsprechen.
This would correspond to the intersection of *both sets*.

4. Reference to regions by expressions involving vagueness

(10a) Also ist $(R \cup S)\circ T = \{(x,z) \mid \exists v\,(((x,v) \in R \vee (x,v) \in S) \wedge \mathbf{(z,v) \in T)\}}$

(10b) Fast richtig. *Am Ende der Formel* ist ein Fehler.
Almost correct. At *the end of the formula*, there is a mistake.

(11a) Wegen der Formel für die Komposition folgt $(R \cup T)\circ(S \cup T) =$
$\{(x,z) \mid \exists z\,((x,z) \in R \wedge (z,y) \in T) \vee \mathbf{\exists z\,((x,z) \in R \wedge (z,y) \in T)\}}$

(11b) Fast richtig. In *der zweiten Hälfte der Formel* ist ein Fehler.
Almost correct. In *the second half of the formula*, there is a mistake.

Figure 1: References to components of mathematical objects in dialog fragments of our corpus

bold, and 2. a tutor response labeled by (#b), with at least one referring expression, in italics. Texts are given in the original German version, accompanied by a translation into English.

The examples are partitioned into four categories. The first one, (examples 1 to 3), illustrate references by the typographical position, from left to right. Items referred to in this manner are qualified by their formal category. (1) refers to an equality – two terms joined by an equal sign – in a sequence of three equalities. (2) refers to an instance of a variable, y, which must be further qualified by its position to distinguish it from another occurrence. (3) refers to the last occurrence of the *and* operator. Distinct surface forms are used for objects referred to by category ("second equality") resp. by name ("second *occurrence* of y").

The second category, the only one specific to recursively structured objects, comprises references which look similar to the previous ones, but they do not reflect the *typographical* position but *structural* embeddings. Objects referred to by this kind of expressions are found on the top level of the embedding object or close to it. In most cases, references to the embedding level where the intended referent is to be found are left unexpressed, which carries the implicit meaning that the referent appears at the top most level in which the referred category can be found. In (4), for example, the entire formula contains many terms as its components, in various levels of embedding, so that orientation on typographic positions is not clear. However, on top level of the inequation chain, there are only three terms and the order among these is perfectly clear. (5) and (6) illustrate the role of incompleteness – only "right side" is mentioned, leaving the object whose right side is meant implicit. Consequently, this must be the right side of the whole formula. The last example in this category, (7) shows the reference to different levels of embedding in one sentence. While "right assertion" refers to the expression on the right side of the equivalence on top level, "left set" refers to the left of the two sets in the equation on the left side of that equivalence.

The third category, which features the reference to sets of objects, shows the interpretation of the embedding level in which the intended referent is to be found on the basis of number constraints. In precise terms, this is an instance of implicature (Grice 1975): if the number of objects that are on top level of the embedding object and satisfy the description, exceeds the cardinality specified, identification of the intended referents is transferred to one of the embedded substructures. In (8), three subexpressions satisfy the metonymic description "complement", but the expression refers only to two. Consequently, the intended referents must be found in one of the substructures where a precise cardinality match is present – here, the right side of the equation. Due to the implicature, expressing this additional qualification is not required. An additional complication arises in the context of interference across referring expressions in one sentence. In (9), "both sets" would be resolved to the two sides of the equation, without the context of the whole sentence. However, since "this" refers to the result of the preceeding assertion, that is, the right side of the equation, this part is in some sense excluded from the context for resolving the next referring expressions. Hence, the left side of the equation yields the two sets on top level as interpretation.

The fourth category comprises examples of references which are in some sense associated with vagueness. In references to formulas, we consider the *end* (example (10)) – which means the region towards the end, as a vague expression, but also the *second half* (example (11)), since it is not entirely clear whether this expression must be interpreted structurally or typographically, and a precise interpretation of "half" in the typographical sense is pointless.

In the following, we present methods for the automated generation of referring expressions of the kind illustrated in Figure 1 – concise ones. We address the following phenomena:

- Implicit scope interpretation
- Incomplete or metonymic expressions
- Implicatures of category and cardinality

We do, however, restrict our task to the generation of single referring expressions with precise references. Hence, we do not address vagueness issues, since the meaning of expressions as occurring in (10) and (11) is not fully clear. Moreover, we do not accommodate the context due to previously generated referring expressions as in (9), which we assume to be done by the embedding process.

3 Operationalization

In this section, we describe an operationalization of generating referring expressions of the kind discussed in the previous section. This operationalization is realized in terms of extensions to the algorithm by Dale and Reiter (1995). This algorithm assumes an environment with three interface functions: *BasicLevelValue*, accessing basic level categories of objects (Rosch 1978), *MoreSpecificValue* for accessing incrementally specialized attribute values according to a taxonomic hierarchy, and *UserKnows* for judging whether the user is familiar with the attribute value of an object. In a nutshell, *MakeReferringExpression* (Figure 2, including our extensions) iterates over the attributes *P* of an intended referent *r* (or a set of referents). In *FindBestValue*, a value is chosen that is known to the user and maximizes discrimination (*RulesOut*) – this value describes the intended referent and rules out at least one potential distractor in *C*. If existing, such values are iteratively collected in *L*, until *P* is empty or a distinguishing description is found. The value *V* of an attribute *A* is chosen within an embedded iteration, starting with the basic level value attributed to *r*, after which more specific values also attributed to *r* and assumed to be known to the user are tested for their discriminatory power. Finally, the least specific value that excludes the largest number of potential distractors and is known to the user is chosen.

The extensions to handle particularities for our concerns comprise several components:

- The knowledge representation of objects is enhanced by properties expressing positions in some context and by a meta-property about the use of descriptors – metonymic use of a descriptor when standing in relation to another one.
- The value selection for context-dependent descriptors requires special treatment; moreover, metonymic expressions are built in some sort of a two-step process.
- The discriminatory power in the subprocedure *RulesOut* is interpreted in local contexts for attributes expressing position.
- Termination criteria include a test whether a cardinality or position-based implicature establishes a unique preference.

$\text{Group}(x) ::=$
$\quad G \equiv \{y \mid \exists z\ (\forall y\ \text{dominates}(z,y))\} \wedge G \supseteq x$

$\text{T-group-items} :: =$
$\quad \{x \mid \exists y\ (\neg\exists z\ \text{dominates}(z,y) \wedge \forall x\ \text{dominates}(y,x))\}$

$\text{L1-items} :: =$
$\quad \{x \mid \exists y\ (y \in \text{T-group-items} \wedge \text{dominates}(y,x))\}$

$\text{Group-pref}(\mathit{Group},N,V) :: =$
$\quad |(r \cup C) \cap \mathit{Group}| = N \wedge$
$\quad \forall x \in ((r \cup C) \cap \mathit{Group})\text{: Position}(x,\mathit{Group},N) = V$

$\text{T-group-pref}(N,V) ::=$
$\quad \text{Group-pref}(\text{T-group-items},N,V)$

$\text{L1-group-pref}(x,N,V) ::= \neg\text{T-Group-pref}(N,V) \wedge$
$\quad \text{L1-items} \supseteq \text{Group}(x) \wedge \text{Group-pref}(x,N,V) \wedge$
$\quad (\forall y\ (\text{Group}(y) \wedge \text{L1-items} \supseteq \text{Group}(y))\text{:}$
$\qquad (x \neq y \rightarrow \neg\text{Group-pref}(y,N,V)))$

Figure 2: Definitions with group components

In order to precisely define the extensions, we introduce some predicates and formal definitions for them (Figure 2). Composition in recursively structured objects is built on *dominates(x,y)*, expressing that component *y* is part of component *x*; chained compositions of *dominates* are acyclic. On that basis, groups of items are built according to local contexts. A *Group* which some items *x* belong to is the set of items dominated by one same item, if existing. Otherwise, *Group* is empty. A special group is the set of items on top level, *T-group-items*, which are all dominated by the entire structure, the root item, which is not dominated by any item. These items also build a group. In contrast, *L1-items*, which comprise the items one level below the *T-group-items*, are not all in one group. Intersection with the *Group* predicate yields subsets, where each element in these sets is dominated by one and the same *T-group-item* (see the definition of *L1-group-pref*). A central definition is *Group-pref* (group preference), used for testing the effect of implicatures. It is defined for the set of relevant items to be used within the algorithm ($r \cup C$), that is, the intended referents and still existing distractors, in relation to a *Group*, in the context of cardinality *N* and position *V*, which apply to the set of items. For that group to be preferred, the relevant items falling into that group must match the given cardinality and the position description (see the definition of *Position* in the next paragraph). On that basis, *T-group-pref* expresses

```
                 MakeReferringExpression ($r$,$C$,$P$)
$L \leftarrow \{\}$, $Ctotal \leftarrow C$                                [1]
for each member $A_i$ of list $P$ do
 case $A_i$ of                                                            [2]
  cardinality: $V \leftarrow |r|$                                          [3]
  global-position: $V \leftarrow$ Position($r$,$Ctotal$,$|r|$)             [4]
  local-position: $V \leftarrow$ Position($r$,Group($r$),$|r|$)            [5]
  other: $V$ = FindBestValue($r$,$A_i$,BasicLevelValue($r$,$A_i$))
 end case
 if RulesOut($<A_i,V>$,$C$) $\neq$ nil then
  if metonymic($A_i$,$X$) and $<$type,$X> \in L$ for some $X$            [6]
     and RulesOut($<A_i,V>$,$Ctotal$) $\supseteq$                          [7]
          RulesOut($<$type,$X>$,$Ctotal$)                                 [8]
   then $L \leftarrow L \setminus \{<\text{type},X>\} \cup \{<\text{type},V>\}$   [9]
   else $L \leftarrow L \cup \{<A_i,V>\}$ end if
  $C \leftarrow C$ - RulesOut($<A_i,V>$,$C$)                               [10]
 end if
 if $C = \{\}$ or Preference-by-Implicature then                          [11]
  if $<$type,$X> \in L$ for some $X$
   then return $L$          (an *identifying* description)
   else return $L \cup \{<\text{type},\text{BasicLevelValue}(r,\text{type})>\}$
  end if end if
end for
return $L$                  (a *non-identifying* description)

                 FindBestValue ($r$,$A$,$initial\text{-}value$)
if UserKnows($r$,$<A,initial\text{-}value>$) = true
then $value \leftarrow initial\text{-}value$
else $value \leftarrow$ no-value end if
if ($spec\text{-}value \leftarrow$ MoreSpecificValue($r$,$A$,$value$)) $\neq$ nil $\wedge$
($new\text{-}value \leftarrow$ FindBestValue($r$,$A$,$spec\text{-}value$)) $\neq$ nil $\wedge$
(|RulesOut($<A,new\text{-}value>$,$C$)| >
 |RulesOut($<A,value>$,$C$)|)                                              [12]
then $value \leftarrow new\text{-}value$ end if
return $value$

                 RulesOut ($<A,V>$,$C$)                                    [13]
if $V$ = no-value then return nil
else case $A_i$ of                                                         [14]
 cardinality: return $C \cap \bigcup$ Group$(c)$ $c \in C$,
                where |Group$(c) \cap C$| < $V$                            [15]
 global-position: return $\{x : x \in C \wedge$ Position(x,$Ctotal$,$|r|$) $\neq V$   [16]
 local-position: return $\{x : x \in C \wedge$ Position(x,Group($x$),$|r|$) $\neq V$
 other: return $\{x : x \in C \wedge$ UserKnows($x$,$<A,V>$) = false$\}$
end case end if

                 Preference-by-Implicature                                 [17]
$V \leftarrow$ any, $N \leftarrow$ any
if $<$global-position,$V> \in L \vee <$local-position,$V> \in L \vee$
   $<$cardinality,$N> \in L$ then                                          [18]
 return (T-group-pref($|r|$,$V$) $\wedge$ T-group-items $\supseteq r$) $\vee$
      (L1-items $\supseteq r$ $\wedge$ L1-group-pref($r$,$|r|$,$V$))        [19]
else return false end if
```

Figure 3: The algorithm in pseudo-code

preference for top-group items, when bound to Group, and *L1-group-pref* expresses preference for such a group with *x* one level below.

The knowledge representation of objects is enriched by some properties which are not intrinsic to an object itself. These properties comprise descriptors *cardinality*, *position*, and the meta-property *metonymic*. The predicate *metonymic(x,y)* expresses the acceptability of a metonymic reference of a descriptor *x* for a category *y* (e.g., an operator for a formula, in mathematical domains). The descriptor *cardinality* easily fits in the standard schema of the procedure. However, it only contributes to the discrimination from potential distractors in the context of effects of implicature. The most complex addition is the descriptor *position*, which expresses some sort of relative position of an object considered within the context of a set of comparable objects (e.g., first, second). There are two dimensions along which such descriptors are meaningful in the domain of mathematical formulas and in similar domains with recursively structured objects: (1) the typographical position within the entire object, referred to by the descriptor *global-position*, and (2) the position within the structural level where the object in question resides, referred to by the descriptor *local-position*. Moreover, that position also depends on the number of objects considered, if subgroups of objects are built prior to checking their position within the entire group (e.g,: the first *two* items). This information is encapsulated in the function *Position(x,y,n)*, where *x* denotes the object or set of objects whose position within group *y* is the value of that function, where subgroups of *n* objects are formed. In order to yield a proper result, *x* must be a subset of *y* and the position value within *y* must be the same for all elements of *x*. Otherwise, the value is undefined. For example, for a group *G*=<1,2,3,4, 5,6>, *Position({3},G,1)* = 3, *Position({3},G,2)* = 2, and *Position({2,3},G,2)* = undefined. In some sense, this handling of positions is a generalization of the ordering for vague descriptors in (van Deemter 2006). Also in accordance with van Deemter, we separate descriptor selection from surface form determination, yielding, for example, "left set" for {<type,set>, <local-position,first>}, the first part of an equation, and "second occurrence of x" for {<type,x>, <local-position,second>}.

In order to process these enhanced representations adequately, we have incorporated appropriate modifications in the procedure *MakeReferringExpression* (labeled by [#] in Figure 3). First, the original set of potential distractors is stored for computations within a global context [1]. Then the value selection for the attribute currently considered is done [2], which is different from the usual call to *FindBestValue* for *cardinality* [3], *global-position* [4], and *local-position* [5]; the latter two are realized by the function *Position*, with appropriate instantiations for the group parameter. Next, the treatment for the inclusion of metonymic properties in the description is addressed. If the metonymic descriptor fits to the object category [6], and its discriminatory power [7] dominates that associated with the type descriptor [8], the descriptor values are conflated by overwriting the type value by that of the metonymic descriptor [9]. The two calls to *RulesOut* involved in the above test ([7] and [8]) are the only references to *Rules Out* where effects on the original, entire set of distractors are tested. Therefore, the parameter *C* is added in the definition of *RulesOut* [13] and in all other places where that procedure is called [10], [12]. Similarly to the inclusion of attribute-value pairs in the description, the exclusion tests in *RulesOut* are specific for non-intrinsic attributes [14]. For *cardinality*, those distractors are excluded which belong to a group where the number of still relevant distractors (those consistent with the partial description built so far) is below that cardinality [15]. Similarly, for testing position values, those distractors are picked for which the values returned by the function *Position,* in dependency of the relevant scope – the group the intended referent(s) belong to, are not consistent with value of the attribute considered (*global-position* resp. *local-position*) [16]. Finally, the termination criterion [11] is enhanced, by taking into account the effect of implicatures through cardinality and position descriptors, by the function *Preference-by-implicature* [17]. In this function, the values of *cardinality* and *global-position* or *local-position* are instantiated, provided they appear in the description *L* [18]. The return value is the result of a test whether there exists preference for the top-level, or for that level 1 group which contains the intended referents [19].

4 Examples

In this section, we illustrate how particularities of our application domain are modeled and how the procedure behaves in generating the referring expressions observed in our corpus. The ordered list of attributes, *P*, consists of <type, form, cardinality, global-order, local-order> for atomic items and of <type, operator, cardinality, local-order, dominated-by> for the composed expressions – dominated-by is the inverse of dominates. The meta-predicate *metonymic* is instantiated for pairs <variable, form>, <expression, local-order>, and <term, operator> for producing expressions such as "x" referring to variable x, "left side" referring to the left part of an assertion or equation, and "complement" referring to a term with complement as top level operator.

We show the generation of two examples.

1. example: "Left set" in (7) in Figure 1. It is generated by choosing "set" as the *type*, followed by unsuccessful attempts to pick an *operator* attribute (there is none defined for that set), and a *cardinality* (which yields no discrimination). Then "first" is chosen for *local-ordering*, yielding unique identification (the embedding is left implicit), and this value is expressed by "left" on the surface.

2. example: "both complements" in (8). It is generated by choosing "term" as the *type*, followed by "complement" as the *operator*, which overwrites "term" due to its specification as metonymic with respect to that category. Then "2" is chosen for *cardinality*, which yields unique identification since a subgroup preference for level one is present.

Altogether, the algorithm is able to generate the expressions occurring in our corpus, or quite similar ones, assisted by the application-specific tailored list *P*. Exceptions constitute reference to regions related to some formula component, such as (3) in Figure 1, effects of interference of scope across several referring expressions, such as (9), and expressions involving vague region descriptors, such as (10) and (11). While the last set of examples comprises more than referring expressions, the first two can be handled, but the generated expressions are typically a bit cumbersome, such as "the third term in the condition of the set" instead of "after the last 'and'-operation" in (3) and "both sets on the left side" instead of simply "both sets" in (9).

5 Conclusion and Discussion

In this paper, we have presented an approach to generating referring expressions that identify components of recursively structured objects. Known techniques are enhanced by measures building metonymic expressions, descriptors expressing positions relative to some subgroup of object components, and exploiting the effect of implicatures due to cardinality and position descriptors. Concise expressions can be generated, in accordance with those in our corpus.

While our elaborations are domain-specific to a certain extent, several parts of our method are also much broader applicable. Metonymic expressions are quite common, and we think that building them within the task of reference generation is superior to doing this in a process thereafter, because this enables an easy computation of the discrminatory power of both alternatives, the implicit and the explicit one. Another aspect of broader relevance concerns the effect of implicatures in connection with object subgroups. While the group building itself, which is based on compositions of the relation *dominates,* is specific to our environment, the techniques to establish preferences among groups and deriving identification from that pertain to other environments. For instance, when a subgroup of two items of some kind is visually identifiable in the context of a few other subgroups with different cardinalities, "the two X's" would lead to the identification of the subgroup in focus, through the effect of implicature, the group formation being based on local proximity. Thus, only the group formation schema needs to be changed.

References

Appelt, D. 1985. Planning English Referring Expressions. *Artificial Intelligence* 26, pp. 1-33.

Bateman, J. 1999. Using Aggregation for Selecting Content when Generating Referring Expressions. In Proc. of the *37th Annual Meeting of the Association for Computational Linguistics (ACL-99)*, pp. 127-134, University of Maryland.

Dale, R. 1988. Generating Referring Expressions in a Domain of Objects and Processes. PhD Thesis, Centre for Cognitive Science, Univ. of Edinburgh.

Dale, R., and Reiter, E. 1995. Computational Interpretations of the Gricean Maxims in the Generation of Referring Expressions. *Cognitive Science* 18, pp. 233-263.

Donellan, K. 1966. Reference and Definite Description. *Philosophical Review* 75, pp. 281-304.

Gardent, C. 2002. Generating Minimal Definite Descriptions. In Proc. of the *40th Annual Meeting of the Association for Computational Linguistics (ACL-2002)*, pp. 96-103, Philadelphia, Pennsylvania.

Grice, H. 1975. Logic and Conversation. In Syntax and Semantics: Vol. 3, Speech Acts, pp. 43-58, Academic Press.

Grosz, B., and Sidner, C. 1986. Attention, Intention, and the Structure of Discourse. *Computational Linguistics* 12, pp. 175-206.

Horacek, H. 2003. A Best-First Search Algorithm for Generating Referring Expressions. In Proc. of the *10th Conference of the European Chapter of the Association for Computational Linguistics (EACL-2003)*, Conference Companion (short paper), pp. 103-106, Budapest, Hungary.

Krahmer, E., v. Erk, S., and Verleg, A. 2001. A Meta-Algorithm for the Generation of Referring Expressions. In Proc. of the 8th European Workshop on Natural Language Generation (*EWNLG-2001)*, pp. 29-39, Toulouse, France.

Levelt, W. 1989. *Speaking: From Intention to Articulation*. MIT Press.

McDonald, D. 1981. Natural Language Generation as a Process of Decision Making under Constraints. PhD thesis, MIT.

Reiter, E. 1990. The Computational Complexity of Avoiding Conversational Implicatures. In Proc. of the *28th Annual Meeting of the Association for Computational Linguistics (ACL-90)*, pp. 97-104, Pittsburgh, Pennsylvania.

Rosch, E. 1978. Principles of Categorization. In E. Rosch and B. Llyod (eds.) *Cognition and Categorization*, pp. 27-48, Hillsdale, NJ: Lawrence Erlbaum.

Stone, M. 2000. On Identifying Sets. In Proc. of the *First International Conference on Natural Language Generation (INLG-2000),* pp. 116-123, Mitzpe Ramon, Israel.

van Deemter, K. 2000. Generating Vague Descriptions. In Proc. of the *First International Natural Language Generation Conference (INLG-2000),* pp. 179-185, Mitzpe Ramon, Israel.

Paraboni, I., and van Deemter, K. 2002. Generating Easy References: the Case of Document Deixis. In Proc. of the *Second International Natural Language Generation Conference (INLG-2002)*, pp. 113-119, Harriman, NY, USA.

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

van Deemter, K. 2006. Generating Referring Expressions that Involve Gradable Properties. *Computational Linguistics*, to appear.

van der Sluis, I. 2005. Multimodal Reference. Dissertation, Tilburg University.

Wolska, M., Vo, B., Tsovaltzi, D., Kruijff-Korbayová, I., Karagjosova, E., Horacek, H., Gabsdil, M., Fiedler, A., and Benzmüller, C. 2004. An Annotated Corpus of Tutorial Dialogs on Mathematical Theorem Proving. In Proc. of the *4th International Conference on Language Resources and Evaluation*, pp. 1007-1010, Lisbon, Portugal.