Using aggregation for selecting content when generating referring expressions

John A. Bateman
Sprach- und Literaturwissenschaften
University of Bremen
Bremen, Germany

e-mail: bateman@uni-bremen.de

Abstract

Previous algorithms for the generation of referring expressions have been developed specifically for this purpose. Here we introduce an alternative approach based on a fully generic aggregation method also motivated for other generation tasks. We argue that the alternative contributes to a more integrated and uniform approach to content determination in the context of complete noun phrase generation.

1 Introduction

When generating referring expressions (RE), it is generally considered necessary to provide sufficient information so that the reader/hearer is able to identify the intended referent. A number of broadly related referring expression algorithms have been developed over the past decade based on the natural metaphor of 'ruling out distractors' (Reiter, 1990; Dale and Haddock, 1991; Dale, 1992; Dale and Reiter, 1995; Horacek, 1995). These special purpose algorithms constitute the 'standard' approach to determining content for RE-generation at this time; they have been developed solely for this purpose and have evolved to meet some specialized problems. In particular, it was found early on that the most ambitious RE goal—that of always providing the maximally concise referring expression necessary for the context ('full brevity')—is NP-hard; subsequent work on RE-generation has therefore attempted to steer a course between computational tractability and coverage. One common feature of the favored algorithmic simplifications is their incrementality: potential descriptions are successively refined (usually non-destructively) to produce the final RE, which therefore may or may not be minimal. This is also often motivated on grounds of psychological plausibility.

In this paper, we introduce a completely different metaphor for determining RE-content that may be considered in contrast to, or in combination with, previous approaches. The main difference lies in an orientation to the organization of a data set as a whole rather than to individual components as revealed during incremental search. Certain opportunities for concise expression that may otherwise be missed are then effectively isolated. The approach applies results from the previously unrelated generation task of 'aggregation', which is concerned with the grouping together of structurally related information.

2 The aggregation-based metaphor

Aggregation in generation has hitherto generally consisted of lists of more or less *ad hoc*, or case-specific rules that group together particular pre-specified configurations (cf. Dalianis and Hovy (1996) and Shaw (1998)); however Bateman et al. (1998) provide a more rigorous and generic foundation for aggregation by applying results from data-summarization originally developed for multimedia information presentation (Kamps, 1997). Bateman *et al.* set out a general purpose method for constructing **aggregation lattices** which succinctly represent *all possible structural aggregations* for any given data set.[1] The application of the aggregation-based metaphor to RE-content determination is motivated by the observation that if something is a 'potential distractor' for some intended referent, then it is equally, under appropriate conditions, *a candidate for aggregation together with the intended referent.* That

[1] 'Structural' aggregation refers to opportunities for grouping inherent in the *structure* of the data and ignoring additional opportunities for grouping that might be found by modifying the data inferentially.

is, what makes something a distractor is precisely the same as that which makes it a potential co-member of some single grouping created by structural aggregation. To see this, consider the following simple example discussed by Dale and Reiter (1995) consisting of three objects with various properties (re-represented here in a simple association list format):[2]

```
(o1 (type dog)(size small)(color black))
(o2 (type dog)(size large)(color white))
(o3 (type cat)(size small)(color black))
```

To successfully refer to the first object o1, sufficient information must be given so as to 'rule out' the possible distractors: therefore, type alone is not sufficient, since this fails to rule out o2, nor is any combination of size or color sufficient, since these fail to rule out o3. Successful RE's are 'the small dog' or 'the black dog' and not 'the small one', 'the dog', or 'the black one'.

Considering the data set from the aggregation perspective, we ask instead how to refer most succinctly to *all* of the objects o1, o2, o3. There are two basic alternatives, indicated by bracketing in the following:[3]

1. (A (small black and a large white) dog) and (a small black cat).
2. (A small black (dog and cat)) and (a large white dog).

The former groups together o1 and o2 on the basis of their shared type, while the latter groups together o1 and o3 on the basis of their shared size and color properties. Significantly, these are just the possible sources of distraction that Dale and Reiter discuss.

The set of possible aggregations can be determined from an aggregation lattice corresponding to the data set. We construct the lattice using methods developed in Formal Concept Analysis (FCA) (Wille, 1982). For the example at hand, the aggregation lattice is built up as follows. The set of objects is considered as a relation table where the columns represent the object attributes and their values, and the rows represent the individual objects. Since the attributes (e.g., 'color', 'size', etc.) can take multiple values (e.g., 'large', 'small'), this representation of the data is called a **multivalued context**. This is then converted into a **one-valued context** by comparing all rows of the table pairwise and, for each attribute (i.e., each column in the table) entering one distinguished value (e.g., T or 1) if the corresponding values of the attributes compared are identical, and another distinguished value (nil or 0) if they are not. The one-valued context for the objects o1–o3 is thus:

object pairs	*type*	*size*	*color*
o1-o2	1	0	0
o1-o3	0	1	1
o2-o3	0	0	0

This indicates that objects o1 and o2 have equal values for their type attribute but otherwise not, while o1 and o3 have equal values for both their size and color attributes but not for their type attributes. The one-valued context readily supports the derivation of **formal concepts**. A formal concept is defined in FCA as an extension-intension pair (A, B), where the extension is a subset A of the set of objects and the intension is a subset B of the set of attributes. For any given concept, each element of the extension must accept all attributes of the intension. Visually, this corresponds to permuting any rows and columns of the one-valued context and noting all the maximally 'filled' (i.e., containing 1's or T's) rectangles. A 'subconcept' relation, '$<_{FCA}$', is defined over the set of formal concepts thus:

$$(A, B) <_{FCA} (A^*, B^*) \quad \text{iff} \quad A \subseteq A^* \Leftrightarrow B^* \subseteq B$$

The main theorem of FCA then shows that $<_{FCA}$ induces a complete lattice structure over the set of formal concepts. The resulting lattice for the present example is shown in Figure 1. Each node is shown labeled with two pieces of information: the intension and the extension. The intensions consist simply of the sets of properties involved. The representations of the extensions emphasize the *function* of the nodes in the lattice—i.e., that the indicated objects (e.g., o1 and o2 for the leftmost node) are *equal with respect to all the attributes contained in the intension* (e.g., type for the leftmost node).

[2]This style of presentation is not particularly perspicuous but space precludes providing intelligible graphics, especially for the more complex situations used as examples below. In case of difficulties, we recommend quickly sketching the portrayed situation as a memory aid.

[3]The exact rendering of these variants in English or any other language is not at issue here.

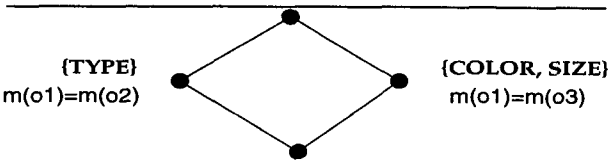

Figure 1: Simple aggregation lattice

This lattice may be construed as an *aggregation* lattice because the functional redundancies that are captured are precisely those redundances that indicate opportunities for structurally-induced aggregation. The leftmost node shows that the attribute `type` may be aggregated if we describe `o1` together with `o2`, and the right-most node shows that {`color`, `size`} may be aggregated when describing `o1` and `o3`.

Now, given the equivalence between aggregation possibilities and 'distractors', we can also use the lattice to drive RE-content determination. Assume again that we wish to refer to object `o1`. In essence, a combination of attributes must be selected that is *not* subject to aggregation; any combination susceptible to aggregation will necessarily 'confuse' the objects for which the aggregation holds when only one of the objects, or **co-aggregates**, is mentioned. For example, the rightmost node shows that an RE with the content `size&color(o1)`, e.g., 'the small black thing', confuses `o1` and `o3`. To select attributes that are appropriate, we first examine the minimal nodes of the lattice to see if any of these do not 'impinge' (i.e., have no aggregation consequences: we make this more precise below) on the intended referent. In this case, however, all these nodes do mention `o1` and so no strong preference for the RE-content is delivered by the data set itself. This appears to us to be the correct characterization of the reference situation: precisely which attributes are selected should now be determined by factors not attributable to 'distraction' but rather by more general communicative goals involving discourse and the requirements of the particular language. The resulting attribute combinations are then checked against the aggregation lattice for their referential effectiveness in a manner reminiscent of the incremental approach of previous algorithms. Selection of `type` is not sufficient but the addition of either `color` or `size` is (`type&color` $= \perp$ and `type&size`$=\perp$).

The reference situation is quite different when we wish to refer to either `o2` or `o3`. For both of these cases there exists a non-impinging node (the right and leftmost nodes respectively). This establishes immediate attribute preferences based on the organizational properties of the data. Content-determination for `o2` should include at least size or color ('the white thing', 'the large thing') and for `o3` at least type ('the cat'). These RE's are minimal.

3 Examples of aggregation-driven RE-content determination

In this section, we briefly summarize some more significant examples of RE-content determination using aggregation. Length limitations will require some shortcuts to be taken in the discussion and we will not follow up all of the alternative RE's that can be motivated.

3.1 Minimal descriptions

Dale and Reiter (1995) consider a number of variant algorithms that deviate from full brevity in order to achieve more attractive computational behavior. The first variant they consider relies on a 'Greedy Heuristic' (Dale, 1989; Johnson, 1974); they illustrate that this algorithm sacrifices minimality by constructing an RE for object `o1` in the context of the following properties concerning a set of seven cups of varying size (large, small), color (red, green, blue) and material (paper, plastic):

```
(o1 (size large)(color red)(material plastic))
(o2 (size small)(color red)(material plastic))
(o3 (size small)(color red)(material paper))
(o4 (size medium)(color red)(material paper))
(o5 (size large)(color green)(material paper))
(o6 (size large)(color blue)(material paper))
(o7 (size large)(color blue)(material plastic))
```

The greedy algorithm produces 'the large red plastic cup' although the true minimum description is 'the large red cup'.

The aggregation-based approach to the same data set provides an interesting contrast in result. The aggregation lattice for the data is given in Figure 2. The lattice is constructed as before: first by converting the multivalued context of the original data set to a one-valued context and then by imposing the subconcept

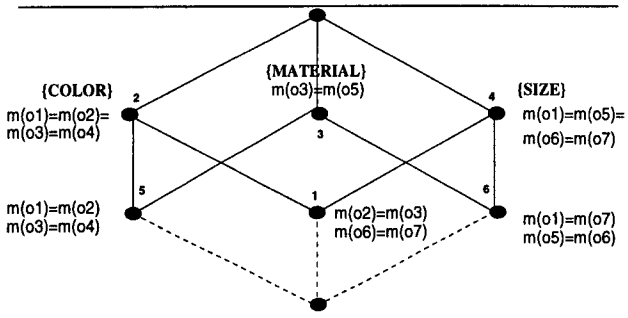

Figure 2: Aggregation lattice for the 'seven cups' example

relation over the complete set of formal concepts. The nodes of the lattice are also labeled as before, although we rely here on the formal properties of the lattice to avoid redundant labeling. For example, the two sets of attribute equalities given for node 1 (one relating o2 and o3, the other relating o6 and o7) apply to *both* color (inherited from node 2) and size (inherited from node 4); we do not, therefore, repeat the labeling of properties for node 1. Similarly, and due to the bidirectionality inherent in the subconcept definition, the attribute equalities of node 1 are also 'inherited' upwards both to node 2 and to node 4. The attribute equalities of node 4 therefore include contributions from both node 1 and node 6. We will generally indicate in the labeling only the *additional* information arising from the structure of the lattice, and even then only when it is relevant to the discussion. So for node 4 we indicate that o1, o5, o6 and o7 now form a single attribute equality set made up of three contributions: one from node 1 (o6 and o7) and two from node 6. Their combination in a single set is only possible at node 4 because node 4 is a superconcept of *both* node 1 and node 6. The other attribute equality set for node 1 (o2 and o3) does not add further information at node 4 and so is left implicit in node 4's labeling. The labeling or non-labeling of redundant information has of course no formal consequences for the information contained in the lattice.

To determine RE-content appropriate for referring to object o1, we again look for *minimal* (i.e., nearest the bottom) concepts, or aggregation sets, that do not 'impinge' on o1. The only node satisfying this requirement is node 1. This tells us that the set of possible co-aggregates for o1 with respect to the properties {size & color} is empty, which is equivalent to stating that there are no objects in the data set which might be confused with o1 if size&color(o1) forms the RE-content. Thus, 'the large red cup' may be directly selected, and this is precisely the true minimal RE for this data set.

3.2 Relational descriptions: restricting recursion

One early extension of the original RE-algorithms was the treatment of data sets involving relations (Dale and Haddock, 1991). Subsequently, Horacek (1995) has argued that the extension proposed possesses several deficits involving both the extent of coverage and its behavior. In particular, Horacek notes that "it is not always necessary that each entity directly or indirectly related to the intended referent and included in the description be identified uniquely" (p49). Partially to handle such situations, Horacek provides a further related algorithm that is intended to improve on the original and which he illustrates in action with reference to a rather more complex situation involving two tables with a variety of cups and bottles on them. One table (t_1) has two bottles and a cup on it, another (t_2) has only a cup. Information is also given concerning the relative positions of the cups and bottles.

The situation that Horacek identifies as problematic occurs when the reference task is to refer to the table t_1 and the the RE-algorithm has decided to include the bottles that are on this table as part of its description. This is an appropriate decision since the presence of these bottles is the one distinguishing feature of the selected table. But it is sufficient for the identification of t_1 for bottles to be mentioned at all: there is no need for either or both of the bottles to be distinguished more specifically. An RE-algorithm should therefore avoid attempting this additional, unnecessary reference task.

To form an aggregation lattice for this fact set, we extend our data representation to deal with relations as well as attributes. This is limited to 'reifying' the relations and labeling them with 'instance variables' as commonly done in input expressions for generation systems (Kasper, 1989). For convenience, we also at this point fold in the type information di-

```
(g7 (pred on)(arg1 b1)(arg1type bottle)(arg2 t1)(arg2type table))
(g8 (pred on)(arg1 b2)(arg1type bottle)(arg2 t1)(arg2type table))
(g9 (pred on)(arg1 c1)(arg1type cup)(arg2 t1)(arg2type table))
(g10 (pred on)(arg1 c2)(arg1type cup)(arg2 t2)(arg2type table))
(g11 (pred left-of)(arg1 b1)(arg1type bottle)(arg2 c1)(arg2type cup))
(g12 (pred left-of)(arg1 c1)(arg1type cup)(arg2 b2)(arg2type bottle))
```

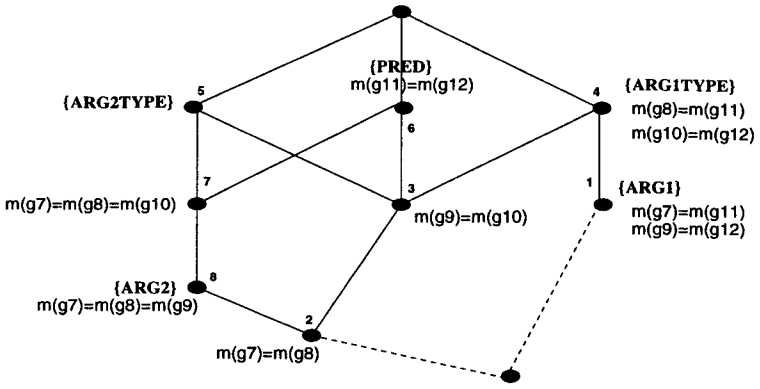

Figure 3: Aggregation lattice for example from Horacek (1995)

rectly as would be normal for a typed semantic representation. This gives the set of facts g7–g12 shown at the top of Figure 3.[4] Once the data set is in this form, aggregation lattice construction may proceed as described above; the result is also shown in Figure 3. This lattice reflects the more complex reference situation represented by the data set and its possible aggregations: for example, node 7 shows that the facts {g7, g8, g9, g10} may be aggregated with respect to both arg2type ('table': node 5) and pred ('on': node 6). Node 3, in contrast, shows that the two distinct sets {g9, g10} and {g7, g8} (again inherited upwards from node 2) may both individually (but not collectively) also be aggregated with pred, arg2type, and *additionally* with arg1type ('cup': node 4).

We first consider the reference task described by Horacek, i.e., identifying the object t1. Now that we are dealing with relations, the objects to be referred to generally occur as *values* of 'attributes'—that is, as entries in the data table—rather than as entire rows. In order to construct an appropriate RE we need to find relations that describe the intended referent and which do not allow aggregation with other relations describing other conflicting referents. We also need to indicate explicitly that the RE-content should not avail itself of the literal instance variables: these are to remain internal to the lattice and to RE-construction so that individuals remain distinct. We therefore distinguish been 'public' and 'private' attributes: public attributes are available for driving linguistic expression, private attributes are not. If we were not to impose this distinction, then referring expressions such as 'the table t1' would be seen as appropriate and probably minimal descriptions![5] An aggregation set that *does not involve* a private attribute will be called a **public concept**.

The first step in constructing an RE is now to identify the relations/events in which the intended referent is involved—here {g7, g8, g9}—and to specify the positions (both private and public) that the referent holds in these. We call the set of potentially relevant relations, the **reference information source set** (RISS). In the present case, the same argument position is held by the intended referent t1 for all RISS-members, i.e., privately arg2 and publicly arg2type. Next, we proceed as before to

[4]Note that this is then isomorphic to a set of SPL specifications of the form `(g7 / on :arg1 (b1 / bottle) :arg2 (t1 / table))`, etc.

[5]Note that this might well be appropriate behavior in some context—in which case the variables would be declared public.

find a non-impinging, minimal aggregate set. However, we can now define 'non-impinging' more accurately. A non-impinging node is one for which there is *at least one public superconcept* fulfilling the following condition: *the required superconcept may not bring any* RISS-*non-member together as co-aggregate with any* RISS-*member drawn from the originating aggregation set with respect to the specified public attribute of the intended referent.*

By these definitions both the minimal nodes of the lattice are non-impinging. However, node 2 is more supportive of minimal RE's and we will only follow this path here; formal indications of minimality are given by the depth and number of paths leading from the node used for aggregation to the top of the aggregation lattice (since any resulting description then combines discriminatory power from each of its chains of superconcepts) and the number of additional facts that are taken over and above the original RISS-members. Node 2 is therefore the 'default' choice simply given a requirement of brevity, although the generation process is free to ignore this if other communicative goals so decide.

There are two public superconcepts for node 2: both of nodes 7 and 3 inherit `arg2type` from node 5 but do not themselves contain a private attribute. Of these only node 7 brings one of the originating RISS-members (i.e., `g7` and `g8` from node 2) into an aggregation set with a RISS non-member (`g10`). Node 2 is therefore non-impinging via node 3. The attributes that may be aggregated at node 2 are `arg2` (node 2 $<_{FCA}$ 8), `arg2type` (2 $<_{FCA}$ 5), `pred` (2 $<_{FCA}$ 6) and `arg1type` (2 $<_{FCA}$ 4). Since this includes `arg2`, the private position of the intended referent, we know that the data set does not support aggregation for `g7` and `g8` with respect to any other distracting value for `arg2`, and so `g7` and `g8`, both collectively and individually, are appropriate and sufficient RE's for `t1`. Rendering these in English would give us:

`g7` or `g8` 'the table with a bottle on it'
`g7` plus `g8` 'the table with some bottles on it'

The precise rendering of the bottles depends on other generator decisions; important here is only the fact that it is *known* that we do not need to uniquely identify which bottles are in question. More identifying information for `arg1` (the bottles `b1` and `b2`) would be necessary only if an aggregation with *other* `arg2`'s (e.g., other tables) were possible, but it is not, and so the type information is already sufficient to produce an RE with no unwanted aggregation possibilities. The aggregation-based approach will not, therefore, go on to consider further facts *unless there is an explicit communicative intention to do so.*

```
(g8' (pred on)(arg1 b2)(arg1type bottle)
   (arg2 t2)(arg2type table))
(g12' (pred left-of)(arg1 c2)(arg1type cup)
   (arg2 b2)(arg2type bottle))
```

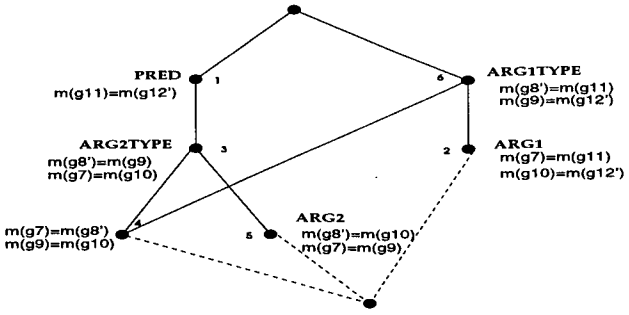

Figure 4: Aggregation lattice for modified example situation from Horacek

3.3 Relational descriptions: when further information is necessary

In this final example we show that the behavior above does not preclude information being added when it is in fact necessary. We show this by adapting Horacek's set of facts slightly to create a different aggregation lattice; we move one of the bottles (`b2`) over to the other table `t2`, placing it to the right of the cup. We show the modified facts and the new aggregation lattice in Figure 4. Here a few concepts have moved in response to the revised reference situation: for example, `arg2type` (node 3) is now a direct subconcept of `pred` indicating that in the revised data set there is a functional relationship between the two attributes: all co-aggregates with respect to `arg2type` are necessarily also co-aggregates with respect to `pred`. In the previous example this did not hold because there were also facts with shared `pred` and non-shared `arg2type` (facts `g11` and `g12`: node 6).

We will again attempt to refer to the table t1 to compare the results with those of the previous subsection. To begin, we have a RISS of {g7, g9} with the intended referent in arg2 (private) and arg2type (public) as before. We then look for non-impinging, most-specific nodes. Here, nodes 4 and 5 are both impinging. Node 4 is impinging in its own right since it sanctions aggregation of both the RISS-members it mentions with non-members with respect to arg2type (node 3) and arg1type (node 6); this deficit is then inherited upwards. Node 5 is impinging by virtue of its first and only available public superconcept, node 3, which sanctions as co-aggregates {g7, g8′, g9, g10} with respect to arg2type. Neither node 4 nor node 5 can therefore support appropriate RE's. Only node 2 is non-impinging, since it does not sanction aggregation involving arg2type or arg2, and is the only available basis for an effective RE with the revised data set.

To construct the RE we take the RISS-member of node 2 (i.e., g7) and consider it and the aggregations it sanctions as candidate material. Node 2 indicates that g7 may be aggregated with g11 with respect to arg1type; such an aggregation is guaranteed not to invoke a false referent for arg1 because it is non-impinging. Moreover, we can infer that g7 alone is insufficient since nodes 3 and 4 indicate that g7 is a co-aggregate with facts with non-equal arg1 values (e.g., g8′), and so aggregation is in fact necessary. The RE then combines:

```
(g7 (pred on)(arg1 b1)(arg1type bottle)
   (arg2 t1)(arg2type table))
(g11 (pred left-of)(arg1 b1)(arg1type bottle)
   (arg2 c1)(arg2type cup))
```

to produce 'the table on which a bottle is to the left of a cup'. This is the only RE that will identify the required table in this highly symmetrical context. No further information is sought because there are no further aggregations possible with respect to arg2 and so the reference is unique; it is also minimal.

4 Discussion and Conclusion

One important feature of the proposed approach is its open-nature with respect to the rest of the generation process. The mechanisms described attempt only to factor out one recurrent problem of generation, namely organizing instantial data to reveal the patterns of contrast and similarity. In this way, RE-generation is re-assimilated and seen in a somewhat more general light than previously.

In terms of the implementation and complexity of the approach, it is clear that it cuts the cake rather differently from previous algorithms/approaches. Some cases of efficient reference may be read-off directly from the lattice; others may require explicit construction and trial of RE-content more reminiscent of the previous algorithms. In fact, the aggregation lattice may in such cases be usefully considered in *combination* with those algorithms, providing an alternative method for checking the consistency of intermediate steps. Here one important difference between the current approach and previous attempts at maintaining consistency is the re-orientation from an incremental procedure to a more static 'overview' of the relationships present, thus providing a promising avenue for the exploration of referring strategies with a wider 'domain of locality'.

This re-orientation is also reflected in the differing computational complexity of the approaches: the run-time behavior of the previous algorithms is highly dependent on the final result (number of properties known true of the referent, number of attributes mentioned in the RE), whereas the run-time of the current approach is more closely tied to the data set as a whole, particularly to the number of facts (n_d) and the number of attributes (n_a). Test runs involving lattice construction for random data sets ranging from 10 to 120 objects, with a number of attributes ranging from 5 to 15 (each with 5-7 possible values) showed that a simple experimental algorithm constructed for uncovering the formal concepts constituting the aggregation lattices had a typical run-time approximately proportional to $n_a n_d^2$. Although worst-case behavior for both this and the lattice construction component is substantially slower, there are now efficient standard algorithms and implementations available that mitigate the problem even when manipulating quite sizeable data sets.[6] For the sizes of data

[6] A useful summary and collection of pointers to complexity results and efficient algorithms is given by Vogt

sets that occur when considering a RE, time-complexity is not likely to present a problem.

Nevertheless, for larger data sets the approach given here is undoubtedly considerably slower than the simplified algorithms reported both by Dale and Reiter and by Horacek. However, in contrast to those approaches, it relies only on generic, non-RE specific methods. The approach also, as suggested above, appears under certain conditions to effectively deliver maximally concise RE's; just what these conditions are and whether they can be systematically exploited remain for future research. Finally, since the use of aggregation lattices has been argued for other generation tasks (Bateman et al., 1998), some of the 'cost' of deployment may in fact turn out to be shared, making a direct comparison solely with the RE-task in any case inappropriate. Other generation constraints might then also naturally contribute to restricting the overall size of the data sets to be considered—perhaps even to within acceptable practical limits.

Acknowledgements

This paper was improved by the anonymous comments of reviewers for both the ACL and the European Natural Language Generation Workshop (1999). Remaining errors and obscurities are my own.

References

John Bateman, Thomas Kamps, Jörg Kleinz, and Klaus Reichenberger. 1998. Communicative goal-driven NL generation and data-driven graphics generation: an architectural synthesis for multimedia page generation. In *Proceedings of the 1998 International Workshop on Natural Language Generation*, pages 8–17. Niagara-on-the-Lake, Canada.

Robert Dale and Nicholas Haddock. 1991. Generating referring expressions involving relations. In *Proceedings of the 1991 Meeting of the European Chapter of the Association for Computational Linguistics*, pages 161–166, Berlin.

Robert Dale and Ehud Reiter. 1995. Computational interpretations of the Gricean maxims in the generation of referring expressions. *Cognitive Science*, 19:233–263.

Robert Dale. 1989. Cooking up referring expressions. In *Proceedings of the Twenty-Seventh Annual Meeting of the Association for Computational Linguistics*, Vancouver, British Columbia.

Robert Dale. 1992. *Generating referring expressions: constructing descriptions in a domain of objects and processes.* Bradford Books, MIT Press, Cambridge, Massachusetts.

Hercules Dalianis and Eduard Hovy. 1996. Aggregation in natural language generation. In Giovanni Adorni and Michael Zock, editors, *Trends in natural language generation: an artificial intelligence perspective*, pages 88–105. Springer-Verlag.

Helmut Horacek. 1995. More on generating referring expressions. In *Proceedings of the Fifth European Workshop on Natural Language Generation*, pages 43–58, Leiden, The Netherlands.

D. Johnson. 1974. Approximate algorithms for combinatorial problems. *Journal of Computer and Systems Sciences*, 9.

Thomas Kamps. 1997. *A constructive theory for diagram design and its algorithmic implementation.* Ph.D. thesis, Darmstadt University of Technology, Germany.

Robert T. Kasper. 1989. A flexible interface for linking applications to PENMAN's sentence generator. In *Proceedings of the DARPA Workshop on Speech and Natural Language.*

Ehud Reiter. 1990. Generating descriptions that exploit a user's domain knowledge. In R. Dale, C. Mellish, and M. Zock, editors, *Current Research in Natural Language Generation.* Academic Press, London.

James Shaw. 1998. Clause aggregation using linguistic knowledge. In *Proceedings of the 1998 International Workshop on Natural Language Generation*, pages 138–147. Niagara-on-the-Lake, Canada.

Frank Vogt. 1996. *Formale Begriffsanalyse mit C++. Datenstrukturen und Algorithmen.* Springer-Verlag.

R. Wille. 1982. Restructuring lattice theory: an approach based on hierarchies of concept. In I. Rival, editor, *Ordered Sets*, pages 445–470. Reidel, Dordecht/Boston.

(1996). Formal techniques for minimizing the size of the data set that is used for further processing are also given.