Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics (ACL), Philadelphia, July 2002, pp. 96-103.

Generating Minimal Definite Descriptions

Claire Gardent
CNRS, LORIA, Nancy
gardent@loria.fr

Abstract

The incremental algorithm introduced in (Dale and Reiter, 1995) for producing distinguishing descriptions does not always generate a minimal description. In this paper, I show that when generalised to sets of individuals and disjunctive properties, this approach might generate unnecessarily long and ambiguous and/or epistemically redundant descriptions. I then present an alternative, constraint-based algorithm and show that it builds on existing related algorithms in that (i) it produces minimal descriptions for sets of individuals using positive, negative and disjunctive properties, (ii) it straightforwardly generalises to n-ary relations and (iii) it is integrated with surface realisation.

1 Introduction

In English and in many other languages, a possible function of definite descriptions is to *identify* a set of referents[1]: by uttering an expression of the form *The N*, the speaker gives sufficient information to the hearer so that s/he can identify the set of the objects the speaker is referring to.

From the generation perspective, this means that, starting from the set of objects to be described and from the properties known to hold of these objects by both the speaker and the hearer, a definite description must be constructed which allows the user to unambiguously identify the objects being talked about.

While the task of constructing singular definite descriptions on the basis of positive properties has received much attention in the generation literature (Dale and Haddock, 1991; Dale and Reiter, 1995; Horacek, 1997; Krahmer et al., 2001), for a long time, a more general statement of the task at hand remained outstanding. Recently however, several papers made a step in that direction. (van Deemter, 2001) showed how to extend the basic Dale and Reiter Algorithm (Dale and Reiter, 1995) to generate plural definite descriptions using not just conjunctions of positive properties but also negative and disjunctive properties; (Stone, 1998) integrates the D&R algorithm into the surface realisation process and (Stone, 2000) extends it to deal with collective and distributive plural NPs.

Notably, in all three cases, the incremental structure of the D&R's algorithm is preserved: the algorithm increments a set of properties till this set uniquely identifies the **target set** i.e., the set of objects to be described. As (Garey and Johnson, 1979) shows, such an incremental algorithm while being polynomial (and this, together with certain psycholinguistic observations, was one of the primary motivation for privileging this incremental strategy) is not guaranteed to find the **minimal solution** i.e., the description which uniquely identifies the target set using the smallest number of atomic properties.

In this paper, I argue that this characteristic of the incremental algorithm while reasonably innocuous when generating singular definite descriptions using only conjunctions of positive properties, renders it

[1] The other well-known function of a definite is to *inform* the hearer of some specific attributes the referent of the NP has.

cognitively inappropriate when generalised to sets of individuals and disjunctive properties. I present an alternative approach which always produce the minimal description thereby avoiding the shortcomings of the incremental algorithm. I conclude by comparing the proposed approach with related proposals and giving pointers for further research.

2 The incremental approach

Dale and Reiter's incremental algorithm (cf. Figure 1) iterates through the properties of the **target entity** (the entity to be described) selecting a property, adding it to the description being built and computing the **distractor set** i.e., the set of elements for which the conjunction of properties selected so far holds. The algorithm succeeds (and returns the selected properties) when the distractor set is the singleton set containing the target entity. It fails if all properties of the target entity have been selected and the distractor set contains more than the target entity (i.e. there is no distinguishing description for the target).

This basic algorithm can be refined by ordering properties according to some fixed preferences and thereby selecting first e.g., some base level category in a taxonomy, second a size attribute third, a colour attribute etc.

$\mathcal{I}$: the domain;
P_e, the set of properties of e;
To generate the UID D_e, do:

1. Initialise: $C := \mathcal{I}$, $D_e := \emptyset$.
2. Check success:
   If $C = \{e\}$ **return** D_e
   elseif $P_e = \emptyset$ **then** fail
   else goto step 3.
3. Choose property $p_i \in P_e$ which picks out the smallest set $C_i = C \cap \{x \mid p_i(x)\}$.
4. Update: $D_e := D_e \cup \{p_i\}$, $C := C_i$, $P_e := P_e - \{p_i\}$. **goto** step 2.

Figure 1: The D&R incremental Algorithm.

(van Deemter, 2001) generalises the D&R algorithm first, to plural definite descriptions and second, to disjunctive and negative properties as indicated in Figure 2. That is, the algorithm starts with a **distractor set** C which initially is equal to the set of individuals present in the context. It then incrementally selects a property P that is true of the target set ($S \subseteq [[P]]$) but not of all elements in the distractor set ($C \not\subseteq [[P]]$). Each selected property is thus used to simultaneously increment the description being built and to eliminate some distractors. Success occurs when the distractor set equals the target set. The result is a distinguishing description (DD, a description that is true only of the target set) which is the conjunction of properties selected to reach that state.

$\mathcal{I}$: the domain;
$S \subseteq \mathcal{I}$, the set to be described;
P_S, the properties true of the set S ($\mathcal{P}_S^+ = \bigcap_{x \in S} \mathcal{P}_x^+$ with $\mathcal{P}_x^+$ the set of properties that are true of x);

To generate the distinguishing description D_S, do:

1. Initialise: $C := \mathcal{I}$, $D_S := \emptyset$.
2. Check success:
   If $C = S$ **return** D_S
   elseif $P_S = \emptyset$ **then** fail
   else goto step 3.
3. Choose property $p_i \in P_S$ s.t. $S \subseteq [[p_i]]$ and $C \not\subseteq [[p_i]]$
4. Update: $D_S := D_S \cup \{p_i\}$, $C := C \cap [[p_i]]$, $P_S := P_S - \{p_i\}$. **goto** step 2.

Figure 2: Extending D&R Algorithm to sets of individuals.

Phase 1: Perform the extended D&R algorithm using all literals i.e., properties in $P_{+/-}$; if this is successful then stop, otherwise go to phase 2.

Phase 2: Perform the extended D&R algorithm using all properties of the form $P \vee P'$ with $P, P' \in P_{+/-}$; if this is successful then stop, otherwise go to phase 3.

Figure 3: Extending D&R Algorithm to disjunctive properties

To generalise this algorithm to disjunctive and negative properties, van Deemter adds one more level of incrementality, an incrementality over the length of the properties being used (cf. Figure 3). First, literals are used i.e., atomic properties and their negation. If this fails, disjunctive properties of length two (i.e. with two literals) are used; then of length three etc.

3 Problems

We now show that this generalised algorithm might generate (i) epistemically redundant descriptions and (ii) unnecessarily long and ambiguous descriptions.

Epistemically redundant descriptions. Suppose the context is as illustrated in Figure 4 and the target set is $\{x_1, x_2\}$.

	pdt	secr	treasurer	board-member	member
x_1	•			•	•
x_2		•		•	•
x_3			•	•	•
x_4				•	•
x_5				•	•
x_6					•

Figure 4: Epistemically redundant descriptions
"The president and the secretary who are board members and not treasurers"

To build a distinguishing description for the target set $\{x_1, x_2\}$, the incremental algorithm will first look for a property P in the set of literals such that (i) $\{x_1, x_2\}$ is in the extension of P and (ii) P is not true of all elements in the distractor set C (which at this stage is the whole universe i.e., $\{x_1, x_2, x_3, x_4, x_5, x_6\}$). Two literals satisfy these criteria: the property of being a board member and that of not being the treasurer[2] Suppose the incremental algorithm first selects the *board-member* property thereby reducing the distractor set to $\{x_1, x_2, x_3, x_4, x_5\}$. Then $\neg$*treasurer* is selected which restricts the distractor set to $\{x_1, x_2, x_4, x_5\}$. There is no other literal which could be used to further reduce the distractor set hence properties of the form $P \vee P'$ are used. At this stage, the algorithm might select the property $pdt \vee secr$ whose intersection with the distractor set yields the target set $\{x_1, x_2\}$. Thus, the description produced is in this case: *board-member* $\wedge\neg$*treasurer* $\wedge(pdt \vee secr)$ which can be phrased as *the president and the secretary who are board members and not treasurers* – whereas the minimal DD *the president and the secretary* would be a much better output.

[2]Note that selecting properties in order of specificity will not help in this case as neither *president* nor *treasurer* meet the selection criterion (their extension does not include the target set).

One problem thus is that, although perfectly well formed minimal DDs might be available, the incremental algorithm may produce "epistemically redundant descriptions" i.e. descriptions which include information already entailed (through what we know) by some information present elsewhere in the description.

Unnecessarily long and ambiguous descriptions. Another aspect of the same problem is that the algorithm may yield unnecessarily long and ambiguous descriptions. Here is an example. Suppose the context is as given in Figure 5 and the target set is $\{x_5, x_6, x_9, x_{10}\}$.

	W	D	C	B	S	M	Pi	Po	H	J
x_1	•									
x_2	•	•								
x_3	•		•							
x_4	•		•			•				
x_5	•		•			•				•
x_6	•		•		•				•	
x_7	•		•		•					
x_8	•	•		•						
x_9	•	•		•				•		
x_{10}	•	•		•			•			
x_{11}										

W = white; D = dog; C = cow; B = big; S = small;
M = medium-sized; Pi = pitbul; Po = poodle; H = Holstein; J = Jersey

Figure 5: Unnecessarily long descriptions.

The most natural and probably shortest description in this case is a description involving a disjunction with four disjuncts namely $Pi \vee Po \vee \vee H \vee J$ which can be verbalised as *the Pitbul, the Pooddle, the Holstein and the Jersey*.

This is not however, the description that will be returned by the incremental algorithm. Recall that at each step in the loop going over the properties of various (disjunctive) lengths, the incremental algorithm adds to the description being built any property that is true of the target set and such that the current distractor set is not included in the set of objects having that property. Thus in the first loop over properties of length one, the algorithm will select the property W, add it to the description and update the distractor set to $C \cap [[W]] = \{x_1, x_2, x_3, x_4, x_5, x_6, x_7, x_8, x_9, x_{10}\}$. Since the new distractor set is not equal to the target set and since no other property of length one satisfies

the selection criteria, the algorithm proceeds with properties of length two. Figure 6 lists the properties P of length two meeting the selection criteria at that stage ($\{x_5, x_6, x_9, x_{10}\} \in [[P]]$ and $\{x_1, x_2, x_3, x_4, x_5, x_6, x_7, x_8, x_9, x_{10}\} \not\subseteq [[P]]$.

$H \vee \neg S$	$\{x_1, x_2, x_3, x_4, x_5, x_6, x_8, x_9, x_{10}\}$
$J \vee \neg M$	$\{x_1, x_2, x_3, x_5, x_6, x_7, x_8, x_9, x_{10}\}$
$B \vee \neg D$	$\{x_1, x_3, x_4, x_5, x_6, x_7, x_8, x_9, x_{10}\}$
$D \vee C$	$\{x_2, x_3, x_4, x_5, x_6, x_7, x_8, x_9, x_{10}\}$
$B \vee C$	$\{x_3, x_4, x_5, x_6, x_7, x_8, x_9, x_{10}\}$

Figure 6: Properties of length 2 meeting the selection criterion

The incremental algorithm selects any of these properties to increment the current DD. Suppose it selects $B \vee C$. The DD is then updated to $W \wedge (B \vee C)$ and the distractor set to $\{x_3, x_4, x_5, x_6, x_7, x_8, x_9, x_{10}\}$. Except for $D \vee C$ and $\neg D \vee B$ which would not eliminate any distractor, each of the other property in the table can be used to further reduce the distractor set. Thus the algorithm will eventually build the description $W \wedge (B \vee C) \wedge (H \vee \neg S) \wedge (J \vee \neg M)$ thereby reducing the distractor set to $\{x_3, x_5, x_6, x_8, x_9, x_{10}\}$.

At this point success still has not been reached (the distractor set is not equal to the target set). It will eventually be reached (at the latest when incrementing the description with the disjunction $Pi \vee Po \vee \vee H \vee J$). However, already at this stage of processing, it is clear that the resulting description will be awkward to phrase. A direct translation from the description built so far ($W \wedge (B \vee C) \wedge (H \vee \neg S) \wedge (J \vee \neg M)$) would yield e.g.,

(1) The white things that are big or a cow, a Holstein or not small, and a Jersey or not medium size

Another problem then, is that when generalised to disjunctive and negative properties, the incremental strategy might yield descriptions that are unnecessarily ambiguous (because of the high number of logical connectives they contain) and in the extreme cases, incomprehensible.

4 An alternative based on set constraints

One possible solution to the problems raised by the incremental algorithm is to generate only **minimal descriptions** i.e. descriptions which use the smallest number of literals to uniquely identify the target set. By definition, these will never be redundant nor will they be unnecessarily long and ambiguous.

As (Dale and Reiter, 1995) shows, the problem of finding minimal distinguishing descriptions can be formulated as a set cover problem and is therefore known to be NP hard. However, given an efficient implementation this might not be a hindrance in practice. The alternative algorithm I propose is therefore based on the use of constraint programming (CP), a paradigm aimed at efficiently solving NP hard combinatoric problems such as scheduling and optimization. Instead of following a *generate-and-test* strategy which might result in an intractable search space, CP minimises the search space by following a *propagate-and-distribute* strategy where propagation draws inferences on the basis of efficient, deterministic inference rules and distribution performs a case distinction for a variable value.

The basic version. Consider the definition of a distinguishing description given in (Dale and Reiter, 1995).

> Let r be the intended referent, and C be the distractor set; then, a set L of attribute-value pairs will represent a distinguishing description if the following two conditions hold:
>
> **C1:** Every attribute-value pair in L applies to r: that is, every element of L specifies an attribute value that r possesses.
>
> **C2:** For every member c of C, there is at least one element l of L that does not apply to c: that is, there is an L in L that specifies an attribute-value that c does not possess. l is said to rule out c.

The **constraints** (cf. Figure 7) used in the proposed algorithm directly mirror this definition.

A description for the target set S is represented by a pair of set variables constrained to be a subset of the set of positive(i.e., properties that are true of all elements in S) and of negative (i.e., properties that are true of none of the elements in S) properties

$\mathcal{I}$: the universe;
$\mathcal{P}_x^+$: the set of properties x has;
$\mathcal{P}_x^- = \mathcal{P} \setminus \mathcal{P}_x^+$: the set of properties x does not have;
$\mathcal{P}_S^+ = \bigcap_{x \in S} \mathcal{P}_x^+$: the set of properties true of all elements of S;
$\mathcal{P}_S^- = \mathcal{P} \setminus \bigcup_{x \in S} \mathcal{P}_x^+$: the set of properties false of all elements of S;
$D_S = \langle P_S^+, P_S^- \rangle$ is a **basic distinguishing description for S** iff:

1. $P_S^+ \subseteq \mathcal{P}_S^+$,
2. $P_S^- \subseteq \mathcal{P}_S^-$ and
3. $\forall c \in C_S, |(P_S^+ \setminus \mathcal{P}_c^+) \cup (P_S^- \cap \mathcal{P}_c^+)| > 0$

Figure 7: A constraint-based approach

of S respectively. The third constraint ensures that the conjunction of properties thus built eliminates all distractors i.e. each element of the universe which is not in S. More specifically, it states that for each distractor c there is at least one property P such that either P is true of (all elements in) S but not of c or P is false of (all elements in) S and true of c.

The constraints thus specify what it is to be a DD for a given target set. Additionally, a **distribution strategy** needs to be made precise which specifies how to search for solutions i.e., for assignments of values to variables such that all constraints are simultaneously verified. To ensure that solutions are searched for in increasing order of size, we distribute (i.e. make case distinctions) over the cardinality of the output description $|P_S^+ \cup P_S^-|$ starting with the lowest possible value. That is, first the algorithm will try to find a description $\langle P_S^+, P_S^- \rangle$ with cardinality one, then with cardinality two etc. The algorithm stops as soon as it finds a solution. In this way, the description output by the algorithm is guaranteed to always be the shortest possible description.

Extending the algorithm with disjunctive properties. To take into account disjunctive properties, the constraints used can be modified as indicated in Figure 8.

That is, the algorithm looks for a tuple of sets such that their union $S_1 \cup \ldots \cup S_n$ is the target set S and such that for each set S_i in that tuple there is a basic

$D_S = D_{S_1} \vee \ldots \vee D_{S_M}$ is a distinguishing description for a set of individuals S iff:

- $1 \leq M \leq |S|$
- $S = S_1 \cup \ldots \cup S_M$
- for $1 \leq i \leq M$, D_{S_i} is a basic distinguishing description for S_i

Figure 8: With disjunctive properties

DD D_{S_i}. The resulting description is the disjunctive description $D_{S_1} \vee \ldots \vee D_{S_M}$ where each D_{S_i} is a conjunctive description.

As before solutions are searched for in increasing order of size (i.e., number of literals occurring in the description) by distributing over the cardinality of the resulting description.

5 Discussion and comparison with related work

Integration with surface realisation As (Stone and Webber, 1998) clearly shows, the two-step strategy which consists in first computing a DD and second, generating a definite NP realising that DD, does not do language justice. This is because, as the following example from (Stone and Webber, 1998) illustrates, the information used to uniquely identify some object need not be localised to a definite description.

(2) Remove the rabbit from the hat.

In a context where there are several rabbits and several hats but only one rabbit in a hat (and only one hat containing a rabbit), the sentence in (2) is sufficient to identify the rabbit that is in the hat. In this case thus, it is the presupposition of the verb "remove" which ensures this: since *x remove y from z* presupposes that y was in z before the action, we can infer from (2) that the rabbit talked about is indeed the rabbit that is in the hat.

The solution proposed in (Stone and Webber, 1998) and implemented in the SPUD (Sentence Planning Using Descriptions) generator is to integrate surface realisation and DD computation. As a property true of the target set is selected, the corresponding lexical entry is integrated in the phrase structure

tree being built to satisfy the given communicative goals. Generation ends when the resulting tree (i) satisfies all communicative goals and (ii) is syntactically complete. In particular, the goal of describing some discourse old entity using a definite description is satisfied as soon as the given information (i.e. information shared by speaker and hearer) associated by the grammar with the tree suffices to uniquely identify this object.

Similarly, the constraint-based algorithm for generating DD presented here has been integrated with surface realisation within the generator INDIGEN (`http://www.coli.uni-sb.de/cl/projects/indigen.html`) as follows.

As in SPUD, the generation process is driven by the communicative goals and in particular, by informing and describing goals. In practice, these goals contribute to updating a "goal semantics" which the generator seeks to realise by building a phrase structure tree that (i) realises that goal semantics, (ii) is syntactically complete and (iii) is pragmatically appropriate.

Specifically, if an entity must be described which is discourse old, a DD will be computed for that entity and added to the current goal semantics thereby driving further generation.

Like SPUD, this modified version of the SPUD algorithm can account for the fact that a DD need not be wholy realised within the corresponding NP – as a DD is added to the goal semantics, it guides the lexical lookup process (only items in the lexicon whose semantics subsumes part of the goal semantics are selected) but there is no restriction on how the given semantic information is realised.

Unlike SPUD however, the INDIGEN generator does not follow an incremental greedy search strategy mirroring the incremental D&R algorithm (at each step in the generation process, SPUD compares all possible continuations and only pursues the best one; There is no backtracking). It follows a chart based strategy instead (Striegnitz, 2001) producing all possible paraphrases. The drawback is of course a loss in efficiency. The advantages on the other hand are twofold.

First, INDIGEN only generates definite descriptions that realize *minimal* DD. Thus unlike SPUD, it will not run into the problems mentioned in section 2 once generalised to negative and disjunctive properties.

Second, if there is no DD for a given entity, this will be immediately noticed in the present approach thus allowing for a non definite NP or a quantifier to be constructed instead. In contrast, SPUD will, if unconstrained, keep adding material to the tree until all properties of the object to be described have been realised. Once all properties have been realised and since there is no backtracking, generation will fail.

N-ary relations. The set variables used in our constraints solver are variables ranging over sets of *integers*. This, in effect, means that prior to applying constraints, the algorithm will perform an encoding of the objects being constrained – individuals and properties – into (pairwise distinct) integers. It follows that the algorithm easily generalises to n-ary relations. Just like the proposition *red(*e_1*)* using the unary-relation "red" can be encoded by an integer, so can the proposition *on(*e_1, e_2*)* using the binary-relation "on" be encoded by two integers (one for *on(*$\_, e_2$*)* and one for *on(*$e_1, \_$*)*.

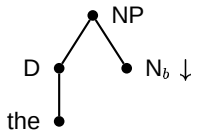
Thus the present algorithm improves on (van Deemter, 2001) which is restricted to unary relations. It also differs from (Krahmer et al., 2001), who use graphs and graph algorithms for computing DDs – while graphs provides a transparent encoding of unary and binary relations, they lose much of their intuitive appeal when applied to relations of higher arity.

It is also worth noting that the infinite regress problem observed (Dale and Haddock, 1991) to hold for the D&R algorithm (and similarly for its van Deemter's generalisation) when extended to deal with binary relations, does not hold in the present approach.

In the D&R algorithm, the problem stems from the fact that DD are generated recursively: if when generating a DD for some entity e_1, a relation r is selected which relates e_1 to e.g., e_2, the D&R algorithm will recursively go on to produce a DD for e_2. Without additional restriction, the algorithm can thus loop forever, first describing e_1 in terms of e_2, then e_2 in terms of e_1, then e_1 in terms of e_2 etc.

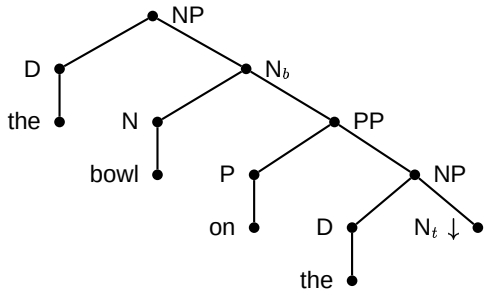
The solution adopted by (Dale and Haddock, 1991) is to stipulate that facts from the knowledge base can only be used once within a given call to the algorithm.

In contrast, the solution follows, in the present algorithm (as in SPUD), from its integration with surface realisation. Suppose for instance, that the initial goal is to describe the discourse old entity e_1. The initially empty goal semantics will be updated with its DD say, $\{bowl(b), on(b,t)\}$.



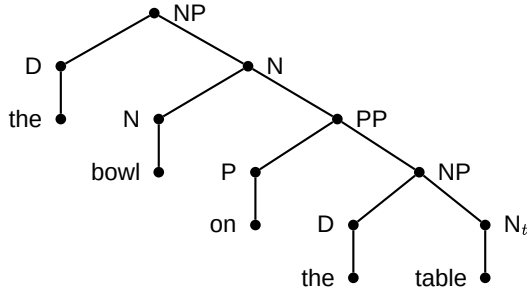
Goal Semantics = $\{bowl(b), on(b,t)\}$

This information is then used to select appropriate lexical entries i.e., the noun entry for "bowl" and the preposition entry for "on". The resulting tree (with leaves "the bowl on") is syntactically incomplete hence generation continues attempting to provide a description for t. If t is discourse old, the lexical entry for *the* will be selected and a DD computed say, $\{table(t), on(b,t)\}$. This then is added to the current goal semantics yielding the goal semantics $\{table(t), bowl(b), on(b,t)\}$ which is compared with the semantics of the tree built so far i..e., $\{bowl(b), on(b,t)\}$.



Goal Semantics = $\{bowl(b), on(b,t), table(t)\}$
Tree Semantics = $\{bowl(b), on(b,t)\}$

Since goal and tree semantics are different, generation continue selecting the lexical entry for "table" and integrating it in the tree being built.



Goal Semantics = $\{bowl(b), on(b,t), table(t)\}$
Tree Semantics = $\{bowl(b), on(b,t), table(t)\}$

At this stage, the semantics of that tree is $\{table(t), bowl(b), on(b,t)\}$ which is equivalent to the goal semantics. Since furthermore the tree is syntactically and pragmatically complete, generation stops yielding the NP *the bowl on the table.*

In sum, infinite regress is avoided by using the computed DDs to control the addition of new material to the tree being built.

Minimality and overspecified descriptions. It has often been observed that human beings produce overspecified i.e., non-minimal descriptions. One might therefore wonder whether generating minimal descriptions is in fact appropriate. Two points speak for it.

First, it is unclear whether redundant information is present because of a cognitive artifact (e.g., incremental processing) or because it helps fulfill some other communicative goal besides identification. So for instance, (Jordan, 1999) shows that in a specific task context, redundant attributes are used to indicate the violation of a task constraint (for instance, when violating a colour constraint, a task participant will use the description "the red table" rather than "the table" to indicate that s/he violates a constraint to the effect that red object may not be used at that stage of the task).

More generally, it seems unlikely that no rule at all governs the presence of redundant information in definite descriptions. If redundant descriptions are to be produced, they should therefore be produced in relation to some general principle (i.e., because the algorithm goes through a fixed order of attribute classes or because the redundant information fulfills a particular communicative goal) not randomly, as is done in the generalised incremental algorithm.

Second, the psycholinguistic literature bearing on the presence of redundant information in definite descriptions has mainly been concerned with unary atomic relations. Again once binary, ternary and disjunctive relations are considered, it is unclear that the phenomenon generalises. As (Krahmer et al., 2001) observed, "it is unlikely that someone would describe an object as "the dog next to the tree in front of the garage" in a situation where "the dog next to the tree" would suffice.

Implementation. The ideas presented in this paper have been implemented within the generator INDIGEN using the concurrent constraint programming language Oz (Programming Systems Lab Saarbrücken, 1998) which supports set variables ranging over finite sets of integers and provides an efficient implementation of the associated constraint theory. The proof-of-concept implementation includes the constraint solver described in section 4 and its integration in a chart-based generator integrating surface realisation and inference. For the examples discussed in this paper, the constraint solver returns the minimal solution (i.e., *The cat and the dog* and *The poodle, the Jersey, the pitbul and the Holstein*) in 80 ms and 1.4 seconds respectively. The integration of the constraint solver within the generator permits realising definite NPs including negative information (*the cat that is not white*) and simple conjunctions (*The cat and the dog*).

6 Conclusion

One area that deserves further investigation is the relation to surface realisation. Once disjunctive and negative relations are used, interesting questions arise as to how these should be realised. How should conjunctions, disjunctions and negations be realised within the sentence? How are they realised in practice? and how can we impose the appropriate constraints so as to predict linguistically and cognitively acceptable structures? More generally, there is the question of which communicative goals refer to sets rather than just individuals and of the relationship to what in the generation literature has been baptised "aggregation" roughly, the grouping together of facts exhibiting various degrees and forms of similarity.

Acknowledgments

I thank Denys Duchier for implementing the basic constraint solver on which this paper is based and Marilisa Amoia for implementing the extension to disjunctive relations and integrating the constraint solver into the INDIGEN generator. I also gratefully acknowledge the financial support of the Conseil Régional de Lorraine and of the Deutsche Forschungsgemeinschaft.

References

R. Dale and N. Haddock. 1991. Content determination in the generation of referring expressions. *Computational Intelligence*, 7(4):252–265.

R. Dale and E. Reiter. 1995. Computational interpretations of the gricean maxims in the generation of referring expressions. *Cognitive Science*, 18:233–263.

W. Garey and D. Johnson. 1979. *Computers and Intractability: a Guide to the Theory of NP-Completeness*. W.H.Freeman, San Francisco.

H. Horacek. 1997. An algorithm for generating referential descriptions with flexible interfaces. In *Proceedings of the 35^{th} Annual Meeting of the Association for Computational Linguistics)*, pages 206–213, Madrid.

P. W. Jordan. 1999. An empirical study of the communicative goals impacting nominal expressions. In *the Proceedings of the ESSLLI workshop on The Generation of Nominal Expression*.

E. Krahmer, S. van Eerk, and André Verleg. 2001. A meta-algorithm for the generation of referring expressions. In *Proceedings of the 8th European Workshop on Natural Language Generation*, Toulouse.

Programming Systems Lab Saarbrücken. 1998. Oz Webpage: `http://www.ps.uni-sb.de/oz/`.

M. Stone and Bonnie Webber. 1998. Textual economy through closely coupled syntax and semantics. In *Proceedings of the Ninth International Workshop on Natural Language Generation*, pages 178–187, Niagara-on-the-Lake, Canada.

M. Stone. 1998. *Modality in Dialogue: Planning, Pragmatics and Computation*. Ph.D. thesis, Department of Computer & Information Science, University of Pennsylvania.

M. Stone. 2000. On Identifying Sets. In *Proceedings of the First international conference on Natural Language Generation*, Mitzpe Ramon.

Kristina Striegnitz. 2001. A chart-based generation algorithm for LTAG with pragmatic constraints. To appear.

K. van Deemter. 2001. Generating Referring Expressions: Boolean Extensions of the Incremental Algorithm. To appear in Computational Linguistics.