Describing Sets with Covers and Sets of Ordinary Assignments

Matthew Stone
Department of Computer Science and Center for Cognitive Science
Rutgers University
110 Frelinghuysen Road, Piscataway NJ 08854-8019
mdstone@cs.rutgers.edu

Abstract

A range of research has explored the problem of generating referring expressions that uniquely identify a single entity from the shared context. But what about expressions that identify sets of entities? In this paper, I adapt a state-of-the-art semantics for plural descriptions—using covers to abstract collective and distributive readings and using sets of assignments to represent dependencies among references—to describe a search problem for set-identifying expressions that largely avoids the computational explosions inherent in computing or searching over the power set representation of salient collections.

1 Introduction

Natural language interaction lends itself to tasks like generalization, abstraction, comparison, and summarization which call for SETS of objects to be identified using definite referring expressions. How are such referring expressions to be constructed in natural language generation (NLG)? This is a difficult problem, and a relatively fresh one. This paper is devoted to delineating the challenges involved and exploring, in a preliminary way, one possible approach.

I begin in this introduction by situating my formulation of the problem of identifying collections within the broader context of the NLG task (section 1.1). It is important to recognize, in the face of the complexity and richness of NLG, that identification is but one of many generation processes that might result in NL descriptions of sets. I introduce the distinctive complications of identifying sets in section 1.3 by comparison with the problem of identifying singular discourse entities in NLG—a problem, which, as section 1.2 reviews, has been the focus of extensive prior work. While the details of these complications depend on the model of NP interpretation and the organization of the search space for referring expressions, the basic problem is simple: there are TOO MANY SETS.

The remainder of the paper proposes an algorithm for the construction of plural noun phrases that avoids explicit calculation or search over the exponential space of collections of salient entities. The algorithm depends on two independently-motivated observations from formal semantics, described in section 2. The first is a COVER semantics for plural predication, a simple scheme of implicit quantification by which diverse lexical meanings can apply to collections [Gillon, 1987, Verkuyl and van der Does, 1991, Schwarzschild, 1994, Schwarzschild, 1996]. The second is an ASSIGNMENT SET semantics for reference to plurals, which provides a way to evoke and describe collections with variables that range only over individuals [van den Berg, 1993, van den Berg, 1996]. Together these observations suggest the treatment of set identification explored in section 3. The assignment set semantics justifies interpreting plurals with constraint networks over individuals; the cover semantics allows us to enforce collective constraints in these networks in a particularly simple way. By preserving or conservatively adapting the representations of singular referring generation in this way, the new algorithm defuses a number of potential combinatorial explosions that would otherwise arise with reference to sets.

While this proposal seems suitable for identification of sets of objects in practical NLG systems, its efficiency and effectiveness in many respects depend on the lexical semantics of possible descriptors. A systematic study of lexical semantics and plural descriptions remains for the future, then—so long as the characterization of set identification in NLG, which I outline next, retains its force.

1.1 Setting

Descriptions of sets obviously have much in common with expressions that describe a single entity from the shared context. In particular, adopting the standard view of NLG as goal-directed activity [Appelt, 1985, Dale, 1992, Moore, 1994, Moore and Paris, 1993], singular and plural descriptions agree both in the kinds of intentions that they can achieve and the stages of generation at which they can be formulated. We cannot expect a single process to be responsible for set descriptions across all intentions or stages of NLG.

For example, as with a singular description, a description of a set may appeal to properties that play a role in the argument the speaker is trying to make, and may therefore address goals above and beyond simple identification of discourse entities. (See [Donellan, 1966, Kronfeld, 1986] on the distinction.) [Green et al., 1998a, Green et al., 1998b] show how such descriptions may be represented and formulated in NLG at a high-level process of content or rhetorical planning. Their representations and algorithms are neutral as to whether a description picks out a set or a single object—capturing both the plural of (1a) and the singular of (1b), for example:

(1) a three newspapers that carry only national news
b the number of readers of the Post-Gazette

At the same time, plurals and singulars are alike in offering resources for reference—such as pronouns, *one*-anaphora or aggregated expressions—that bypass explicit description altogether. In stark contrast to descriptions like (1) that reflect high-level goals for NLG, the use of these resources may be quite closely dependent on the surface form being generated and so could reflect a relatively late decision in the generation process [Dale and Haddock, 1991, Reiter, 1994, Dalianis, 1996].

These complexities notwithstanding, we can expect many descriptions of sets, like descriptions of individuals, to be formulated from scratch to achieve purely referential goals during the SENTENCE PLANNING phase of NLG, between content planning and surface realization [Rambow and Korelsky, 1992, Reiter, 1994]. For, internal representations of sets—most likely simple lists of individuals, perhaps augmented with more abstract information recounting their derivation from processes of matching, clustering or search—are no more likely than internal representations of singulars to identify referents uniquely based on information in the shared context. Indeed, internal representations of plurals and singulars alike frequently may be unintelligible to a human interlocutor or not directly realizable as natural language. It is this process of referring expression generation for sets of entities during sentence planning that this paper addresses.

1.2 Framework

A range of research has explored the simpler problem of generating referring expressions that uniquely identify a single entity from the shared context, including [Dale and Haddock, 1991, Dale and Reiter, 1995, Horacek, 1996, Stone and Doran, 1997]. This section introduces this research, following [Dale and Haddock, 1991] most closely. Although these proposals differ in their details, they share a common perspective and common data structures which make any extension to descriptions of sets quite challenging. In particular, the operations they use to search and update sets of possible referents for expressions would be swamped, if applied directly to alternative SET referents, by the enormous number of such sets.

These approaches represent a description as a set of CONSTRAINTS. Each constraint is an atomic formula with free variables that specifies the requirement that some lexical meaning contributes to the description; the variables are placeholders for the discourse entities that the description identifies. For example, the referring expression *the rabbit in the hat* corresponds to the set of constraints in (2); the variable x abstracts the rabbit we intend to refer to, while the variable y abstracts the hat:

(2) $\{rabbit(x), hat(y), in(x,y)\}$

The interpretation of such a description is modeled using the notion of a CONTEXT SET. The context set for an entity r and a description L gives the set of entities that are at least as salient as r at the point in the discourse where L appears. For simplicity, we assume a single context set S for all salient entities and leave the dependence on the position of L implicit. Informally, to uniquely identify r in context, a generator must construct a description that is known in context to be true of r and not of any other entity in S (these entities are called r's DISTRACTORS). To model the possible resolutions that the hearer might entertain for an incomplete (ambiguous) description of r, a generator will use the elements of S known in context to satisfy that description.

To formalize this model, we need to make it explicit that the description L consists of a set of constraints $R_i(\boldsymbol{x})$ formulated in terms of a tuple of variables $\boldsymbol{x} = \langle x_1, \ldots, x_k \rangle$; L is intended to refer SIMULTANEOUSLY to a TUPLE of referents $\boldsymbol{r} = \langle r_1, \ldots, r_k \rangle$. (I adopt the notation throughout that $\boldsymbol{v}$ is a tuple and v_i is component i of $\boldsymbol{v}$.) Assuming a single context set for individuals, we can adopt a pointwise definition of context sets for tuples:

(3) $S(\boldsymbol{r}) := \{\boldsymbol{a} : a_i \in S\}$

There is room for a more subtle definition of context sets for tuples to better encode the internal and external attentional dynamics of referring expressions, but (3) suffices for present purposes.

The interpretation $I(L)$ of a description of $\boldsymbol{r}$ is the set of salient tuples in the context that satisfy the description. $I(L)$ is defined in (4a), using $[\textsc{c}]p$ to indicate that p is known in the current context. The uniqueness condition required for the description to identify $\boldsymbol{r}$ is that its interpretation is compatible only with $\boldsymbol{r}$—as in (4b).

(4) a $I(L) := \{\boldsymbol{a} \in S(\boldsymbol{r}) : [\textsc{c}]R_i(\boldsymbol{a}) \text{ for all } R_i(\boldsymbol{x}) \in L\}$
b $I(L) = \{\boldsymbol{r}\}$

Implementations approximate $I(L)$ using CONSTRAINT SATISFACTION heuristics from Artificial Intelligence [Mackworth, 1987]. A CONSTRAINT NETWORK for a description L determines a tuple $\boldsymbol{C}$ which specifies a generous set of possible values C_i for each variable x_i in L. The network recognizes L as referring uniquely when each C_i is the singleton set $\{r_i\}$. (Note that since the network is approximate, there may be descriptions L for which the interpretation $I(L)$ is a singleton but which are not recognized as uniquely referring by the constraint network.) Typically, $\boldsymbol{C}$ is computed by an efficient heuristic consistency test that refines an initial vector $\boldsymbol{D}$ of possible values (or DOMAINS) for variables by comparison with the constraints $R_i(\boldsymbol{x})$—or rather, the set of commonly known satisfying tuples $\mathcal{R}_i$ giving values for variables free in $R_i(\boldsymbol{x})$. Whereas the sets $S(\boldsymbol{r})$ and $I(L)$ may grow exponentially in the size of the description L, the constraint network can be represented and updated in polynomial space and time. We can denote the constraint network for description L on initial values $\boldsymbol{D}$ as $N(L, \boldsymbol{D})$. The usual case, where the domain D_i for each variable is just the context set S, we can abbreviate as $N(L)$.

In this formalism, the task of constructing a description to identify some entity r can be formulated as a state-space search problem. Each state is a tuple Σ as in (5):

(5) $\Sigma : \langle L, \boldsymbol{x} = \langle x, \ldots \rangle, \boldsymbol{r} = \langle r, \ldots \rangle, N(L) \rangle$

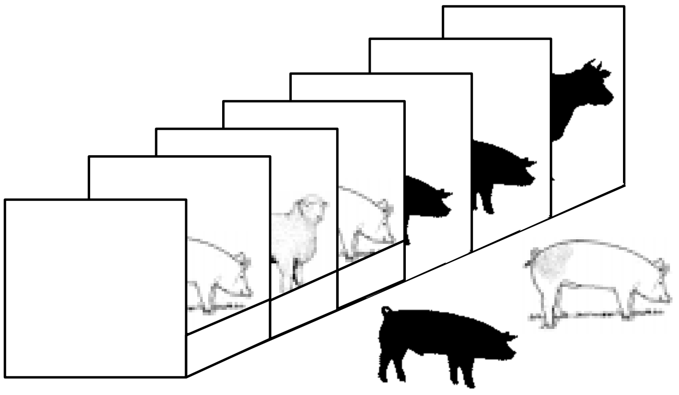

Figure 1: A scene

The state consists of a description L, the free variables of description $\boldsymbol{x}$ (which include a distinguished head variable x), the intended values for the free variables $\boldsymbol{r}$ (including the head referent r for x), and the consistent values for the free variables, as represented heuristically by a constraint network $N(L)$. The initial state Σ_0 is built on an empty description E:

(6) $\Sigma_0 = \langle E, \langle x \rangle, \langle r \rangle, N(E) \rangle$

Goal states are those where $N(L)$ uniquely identifies $\boldsymbol{r}$—with $\boldsymbol{r}$ represented as the unique consistent values for $\boldsymbol{x}$.

At any state Σ, the grammar and knowledge base define a set of constraints of the form $p = R(\boldsymbol{x};\boldsymbol{y})$ that can be added to the description—R is some domain relation, $\boldsymbol{x}$ names old variables from L while $\boldsymbol{y}$ names fresh variables. The availability of such a constraint is a function of the shared status of some fact $R(\boldsymbol{r};\boldsymbol{s})$ in the knowledge base and a function of an available syntactic relation for combining p with L which coindexes some of the variables of p with variables of L. The new state $\Sigma \oplus \langle p, \boldsymbol{s} \rangle$ obtained by incorporating p into the description not only updates the constraints and constraint networks, but also extends the variables by $\boldsymbol{y}$, and extends the intended referents by $\boldsymbol{s}$:

(7) $\Sigma \oplus \langle p, \boldsymbol{s} \rangle := \langle L \cup \{p\}, \boldsymbol{xy}, \boldsymbol{rs}, N(L \cup \{p\}) \rangle$

Thus any algorithm for identifying single entities by description carries out a search process in this space; it starts at Σ_0 and repeatedly explores states $\Sigma \oplus K$ accessible from the current state Σ until it arrives at a goal state. While this characterization of description generation as search exposes the logical problem the generator faces, in practice the search is often managed quite simply: for example, [Dale and Haddock, 1991] select transitions among states according to a greedy heuristic based on the number of values remaining in the constraint network, while [Dale and Reiter, 1995] select alternatives by exploring different kinds of constraints in a fixed order.

1.3 Problems of Plurality

In linguistic semantics, the traditional view of reference to pluralities, found for example in [van Eijck, 1983, Kamp and Reyle, 1993], is that discourse referents sometimes take on sets or sums of individuals as values and that explicit operators of distributivity mediate predication over individuals within those sets. If we adopt this perspective, several problems arise immediately in extending the kind of NLG approach sketched in section 1.2 to generate plural expressions that identify sets of salient objects. These problems can be illustrated by considering the scenario illustrated in Figure 1 and the referring expressions in (8).

(8) a the black pigs in the stalls

b the pigs in the open stalls
c the pigs in the stalls

Suppose the stalls are denoted s_1 through s_6 according to their order from left to right in the picture, so that, for example, stalls s_1, s_2 and s_3 are closed and s_4, s_5 and s_6 are open. Similarly, represent the animals in stalls as a_1 through a_6, so that for example, a_2 is the sheep, a_6 is the cow, and a_1, a_3, a_4 and a_5 are pigs. Call the other black pig a_7 and the other white pig a_8. These symbols allow us to describe the pigs identified in (8): (8a) and (8b) are alike in identifying the set $\{a_4, a_5\}$ while (8c) identifies $\{a_1, a_3, a_4, a_5\}$.

What kind of semantic representation underlies these descriptions, on a theory like [van Eijck, 1983, Kamp and Reyle, 1993]? We use upper-case variables like X, Y, etc., to range over (discourse referents for) collections, and we use the operator of DISTRIBUTIVITY defined in (9) to lift a predicate P that describes ordinary individuals to a predicate ${}^{D}P$ that describes collections:

(9) $${}^{D}P := \lambda X.\forall e(e \in X \supset Pe)$$

That is, ${}^{D}P$ is true of X only if P is true of all the elements of X.

Now, we naturally regard *black, pig, open, stall* and the property of being *in some place* as properties of individuals only. The property of *having something inside*, however, is perhaps best treated as a property that is generally COLLECTIVE, applying to sets directly. Here any pig happens to be *in the stalls* in virtue of some stall it's in, but if you see a figure *in the trees* there may be no tree you see the figure in—a collective treatment could account for this, if a collection of entities defined a container that included both the space each defined individually as well as the space between them. Assuming this analysis, then, the noun phrases of (8) are modeled using the following constraints on X and Y:

(10) a $\{({}^{D}\mathit{black})(X), ({}^{D}\mathit{pig})(X), ({}^{D}\lambda p.\mathit{in}(p,Y))(X), ({}^{D}\mathit{stall})(Y)\}$
b $\{({}^{D}\mathit{pig})(X), ({}^{D}\lambda p.\mathit{in}(p,Y))(X), ({}^{D}\mathit{open})(Y), ({}^{D}\mathit{stall})(Y)\}$
c $\{({}^{D}\mathit{pig})(X), ({}^{D}\lambda p.\mathit{in}(p,Y))(X), ({}^{D}\mathit{stall})(Y)\}$

Constraints like (10) raise computational problems over constraints like (2) for every aspect of our account of referring expression generation. For one thing, the representation of salient alternatives in terms of context sets is intractable. By analogy with context sets for singular entities, the context set for a collection P should give the set of COLLECTIONS that are currently as salient as P. Explicit representation of such a context set is hopeless for all but the most salient collections—Figure 1, for example, introduces more than sixteen thousand collections, just with its six stalls and eight animals.

Even if we accept, for the moment, these profligate context sets, the characterization of referent identification in terms of constraint satisfaction is inadequate for collections. Take (8c) and its associated constraint L given in (10c). Following the definition of interpretation in (4a), $I(L)$ gives the tuples $\langle P, Q\rangle$ where P is a set of pigs, Q is a set of stalls, and each pig in P is in Q. Far from specifying a unique tuple, $I(L)$ is compatible with dozens of collections of pigs and stalls from the scene. Yet (8c) can be used felicitously—to designate all four stalled pigs (and, plausibly, all six stalls), the most inclusive of possible interpretations. Evidently, we must reformulate the uniqueness condition of (4) into a MAXIMALITY condition, and adapt our intermediate representation of interpretations accordingly.

The combinatorics of collections not only could blow up the representation of each state in searching for a description—it could also explode the number of successor states at any point in search. Adding a new constraint to the description may involve selecting a related collection of entities to describe simultaneously. For example, according to the search step outlined in (7), in going from *the pigs* to *the pigs in the stalls*, we would select some set of stalls that the pigs we intend to describe are inside, and fold in the goal of identifying that set of stalls. Again, there may be an inordinate number of such sets.

Informal analysis of the examples in (8) suggests that this choice need not be made immediately in this way, however. Note that the referring expressions in (8a) and (8b) identify the same pigs, but one expression refers to all of the stalls while the other refers to just the open stalls. Both expressions might sensibly be viewed as possible refinements of (8c) which narrow down the set of pigs to the intended set. In so doing, they may (or may not) narrow the set of stalls that the description is INTENDED to identify. In this way, the demands of search in the case of plural referring expressions motivates not only the development of better representations INSIDE search states but also the formulation of more flexible relationships BETWEEN search states.

1.4 Solution Sketch

In this paper, I propose an extension of the framework presented in section 1.2 which can address these problems in a linguistically-motivated way. In fact, the extension is quite direct—in some sense it involves more of a REINTERPRETATION of the constraint representation for referring expressions than a REDEFINITION of the representation. Here is the basic idea.

We continue to assume a description L with free variables $\boldsymbol{x}$, and to associate L with a constraint network specifying values $\boldsymbol{C}$ for the variables $\boldsymbol{x}$. However, instead of regarding $\boldsymbol{C}$ as describing ALTERNATIVE resolutions for an AMBIGUOUS referring expression, we regard each C_i as the COLLECTION that the description associates with x_i.

To illustrate, take (8c) *the pigs in the stalls*. We can interpret this as the constraint network of (11):

(11) $$(\{pig(x), in(x,y), stall(y)\}, \langle C_x = \{a_1, a_3, a_4, a_5\}, C_y = \{s_1, s_2, s_3, s_4, s_5, s_6\}\rangle)$$

The network associates the variable x with the entities $\{a_1, a_3, a_4, a_5\}$; the network therefore interprets x as a plural reference to these four pigs. Similarly, by associating y with $\{s_1, s_2, s_3, s_4, s_5, s_6\}$, the network interprets y as a plural reference to all six stalls.

This reinterpretation of a constraint network like (11) requires a corresponding change in the treatment and interpretation of constraints. For, as indicated already in (10) and reflected in the values for variables in (11), the constraints not only describe collections distributively but in some cases describe them collectively. But (11) does not appeal to explicit operators for distributive or collective predication. Instead, my proposal will be as follows: an individual value for a variable x_i maintains its membership in C_i in the presence of a collective constraint R whenever it belongs to a SUBSET M of C_i which participates directly in R (with sets of possible values of other variables). In other words, the constraint R must COVER the possible values for variables. For example, in (10), relationships of the form $in(\{a_i\}, C_y)$ may be used to cover both C_x and C_y, and hence to show that C_x and C_y satisfy the *in* constraint.

The assumption that lexical constraints apply to collections by covering and the assumption that plural reference can be represented in terms of variables with individual values both have a strong grounding in formal semantics, which section 2 outlines. The computational consequences of these assumptions are worked out in section 3. First, section 3.1 describes the operation of covering-constraint networks. With this interpretation of variables and constraints, the addition of an additional constraint triggers a relaxation process in which values for variables are discarded when they do not contribute to any tuple in a corresponding constraint, while tuples from constraints are discarded when some elements of their component sets fall outside the possible values for corresponding variables. The need for compact representations of constraints poses a special challenge, as described in section 3.2.

As section 3.3 shows, these constraint techniques enable a a state-space definition for set identification that exactly mirrors the singular state-space of section 1.2. This definition just generalizes the goal for description so that identification is complete when the constraint interpretation of a referring expression matches the speaker's intended interpretation. However, it is better to diverge from the singular treatment by relaxing the speaker's intentions as described in section 3.4, so that the description need not fix intended referents of all variables in advance. This allows new constraints to narrow the intended referents for those variables when convenient; a constraint network, describing the entities that speaker could still plan the description to pick out, can be used to find new descriptors that do not rule out those elements that the speaker DOES explicitly intend to identify.

The result is a natural response to the problems of plurality introduced in section 1.3. By maintaining only sets of referents, rather than sets of sets of referents, this algorithm maintains a compact representation of context and interpretation. By representing the speaker's intended interpretation dynamically, this algorithm maintains the needed flexibility in intermediate states. Finally, by initializing the values of variables to the full set of salient referents, subsequently narrowing referents based on constraints, and stopping as soon as hearer's and speaker's interpretations agree (regardless of the cardinality of referents), the algorithm implements a maximization interpretation of plural reference.

2 Plurals in Formal Semantics

2.1 Covers

The discussion of (8) alluded to two ways that linguistic predicates can describe collections. DISTRIBUTIVE predicates, like ^{D}pig, characterize collections based on properties of the individuals involved. COLLECTIVE predicates describe collections that jointly participate in some relation. A clear example is *met* in (12).

(12) The workers met.

(12) is naturally understood as claiming that the workers engaged in a single joint meeting. Many descriptions of sets, such as (13), can be made true either distributively or collectively.

(13) The workers lifted the cabinet.

This sentence could describe a single event in which the workers jointly lifted the cabinet, or it could describe a series of events in which each worker individually lifted the cabinet. We can represent this as an explicit ambiguity using the distributive operator (and a lexical semantics for *lift* that applies indifferently to individuals or collections of individuals). Letting W denote the set of workers and c denote the cabinet, the alternative readings would be represented in (14).

(14) a $lift(W,c)$
b $(^{D}\lambda x.lift(w,c))W$

By contrast, the treatment of plural predication suggested in section 1.4 is inspired by the idea, explored by [Gillon, 1987, Verkuyl and van der Does, 1991, Schwarzschild, 1994, Schwarzschild, 1996], that collective and distributive readings of plurals represent only the extremes in a larger space of readings based on a flexible decomposition of a plural into constituents. The motivation for this view comes from examples such as (15):

(15) Rogers, Hammerstein and Hart wrote musicals.

This sentence is true, but only in virtue of the joint action of Rogers and Hammerstein in writing some musicals and the joint action of Rogers and Hart in writing other musicals. As a matter of fact, the three never wrote a musical individually or as a single team, so both the collective and distributive readings represented in (16) are false.

(16) a $(^{D}write\text{-}musicals)\{rogers, hammerstein, hart\}$
b $write\text{-}musicals(\{rogers, hammerstein, hart\})$

Similar examples illustrate that relations intermediate between distributive and collective can be used in definite plural descriptions. For example, imagine a context in which the speaker is contrasting a first rail system, in which a single track runs along each route, with a second, in which pairs of tracks run side-by-side. The speaker may go on with:

(17) The parallel tracks between cities let traffic move in both directions simultaneously.

The noun phrase *the parallel tracks between cities* may pick out exactly the tracks in the second rail system. Of course, there is no suggestion that those tracks form one collection all of whose elements are parallel to one another; it suffices that the tracks cluster into subgroups whose elements are parallel.

The new algorithm follows Schwarzschild's proposal most closely. Schwarzschild argues that we establish that a linguistic predicate applies to a plural argument by recovering a salient cover of that argument from the context. A cover here means a set of pluralities whose union or sum is the overall plural argument. That is, a cover for S is a family $\mathcal{S}$ with $S = \bigcup \mathcal{S}$.

Given the cover, the overall plural predication holds just in case the basic property denoted by the predicate is true (collectively) of each of the sets in the cover. Formally, predicate $\mathcal{R}$ applies to S on cover $\mathcal{S}$ just in case if $S' \in \mathcal{S}$ then $S' \in \mathcal{R}$. For example, the sets consisting of Rogers and Hammerstein and of Rogers and Hart form the salient cover of Roger, Hammerstein and Hart in (15); the example is true because each of the sets in this cover directly enjoys the property of having written a musical.

In the case of definite reference to a collection S, we can regard the tuples in any predicate $\mathcal{R}$ as defining the appropriate salient cover $\mathcal{S}$ of S for plural predication. The tuples involved in using $\mathcal{R}$ to identify S are part of the shared context. In this case, we must simply find $\mathcal{S} \subseteq \mathcal{R}$ with $\mathcal{S}$ a cover of S. (Things would be more difficult if we were considering descriptions that provide the hearer with NEW information about the collection S.) Generalizing this to relations of multiple arguments, we can say relation $\mathcal{R}$ characterize a tuple of sets $\boldsymbol{S}$ just in case there is an $\mathcal{F} \subseteq \mathcal{R}$ with $S_n = \bigcup\{F_n : \boldsymbol{F} \in \mathcal{F}\}$ for each n. This formalizes the covering interpretation of constraints suggested earlier and adopted in section (3).

2.2 Sets of Assignments

The use of individual values for plural variables follows van den Berg's treatment of dependent plurals in dynamic semantics [van den Berg, 1993, van den Berg, 1996]. Van den Berg's central observation is that discourse can both set up and maintain dependencies between the individuals in one set and the individuals in another.

(18) a Every man loves a woman.
b They prove this by giving them flowers.

In (18) for example, the first sentence introduces a set of men and a set of women, where each man in the one set is related to a woman in the other set (by love); the second sentence builds on that relationship, indicating another connection (of giving) between each man and the corresponding woman.

Van den Berg argues that these dependencies are best formalized by dispensing with set-valued discourse referents altogether. Instead, he proposes to model plurality indirectly by representing the state of the discourse as a *set of assignments*. When a set of assignments G is in force, a discourse referent x picks out a set $G(x)$ defined by $\{g(x) \mid g \in G\}$. An atomic relation such as $p(x,y)$ is always applied collectively to the sets of individuals related to its arguments under the current set of assignments.

Now, van den Berg accounts for distributive contexts by allowing the assignments G in force to be partitioned and quantified over, leading to a narrowed set of values for discourse referents—perhaps just a single individual. This is accomplished by an operator $\Delta_x p$ that distributes over the value of discourse referent x in evaluating the truth of p. Formally, $\Delta_x p$ is true at G just in case for each element u of $G(x)$, p is true at $\{g \in G : g(x) = u\}$.

To illustrate, suppose (18a) sets up a set of assignments of the form $\langle x \to m, y \to w \rangle$ where man m loves woman w. Then we get the right interpretation for *they give them flowers* (as in (18b)) by translating it as in (19):

(19) Δ_x *give-flowers*(x,y)

The distribution over assignments takes into account the dependency of y on x, so that (19) is true only if m gives flowers to w for each assignment $\langle x \to m, y \to w \rangle$.

As outlined in section 1.2, a singular constraint network $N(L)$ is an approximation to a set of tuples $I(L)$; $I(L)$ can just as well be regarded as a set of ordinary assignments of individuals to variables. Section 1.4 proposed, and section 3 adopts, a constraint network interpretation of plural referring expressions where variables again take on singular values. If such a network is to be regarded as an approximation to some plural interpretation, it must be an interpretation such as van den Berg's which involves sets of ordinary assignments. (Of course, we would expect this interpretation to be reformulated using cover-operators rather than distributivity operators.) Van den Berg's success in representing discourse using such sets of ordinary assignments helps justify what might otherwise seem an unprincipled or inadequate representation for plurality.

An important question for future research is how well constraint network representations of assignments can encode the dependencies among discourse referents illustrated by (18). Constraint algorithms generally involve a collapse of dependencies in assignments by using independent values for variables.

3 Data Structures and Algorithms

This research allows us to see a constraint network like (11) (repeated as (20a)) as a natural heuristic approximation to the interpretation of a plural description like (8c) (repeated as (20b)).

(20) a $(\{pig(x), in(x,y), stall(y)\}, \langle C_x = \{a_1,a_3,a_4,a_5\}, C_y = \{s_1,s_2,s_3,s_4,s_5,s_6\}\rangle)$
b the pigs in the stalls

I now develop a detailed computational account showing how we can manipulate plural descriptions in generation using such constraint representations. Naturally, the first ingredient of this account is a constraint-satisfaction heuristic that accounts for cover-constraints on collections.

3.1 Collective Constraints

Let us start by considering a constraint network over a pair of variables x and y. Suppose the possible values for x are C_x and the possible values for y are C_y, and we have a single constraint $R(x,y)$ which is interpreted as a set $\mathcal{R}$ of pairs of collections $\{\boldsymbol{R} = \langle R_x, R_y\rangle\}$.

To interpret the constraint computationally, we need a scheme to restrict the sets C_x and C_y for compatibility with a covering interpretation of $R(x,y)$. First, we construct a set of tuples $l(\mathcal{R}, \langle C_x, C_y\rangle)$ that LIMITS $R(x,y)$ to C_x and C_y:

(21) $l(\mathcal{R}, \langle C_x, C_y\rangle) := \{\langle R_x, R_y\rangle \in \mathcal{R} : R_x \subseteq C_x, R_y \subseteq C_y\}$

In words, $l(\mathcal{R}, \langle C_x, C_y\rangle)$ contains the tuples of $\mathcal{R}$ that relate subsets of C_x to subsets of C_y. Call the limited relation $\mathcal{T}$. We construct narrowed sets of values for x and y that TEST out consistent with $\mathcal{T}$—sets $t(C_x, \mathcal{T}, 1)$ and $t(C_y, \mathcal{T}, 2)$ defined in (22).

(22) $t(C, \mathcal{T}, n) := \{u \in C : \text{for some } \boldsymbol{T} \in \mathcal{T}, u \in T_n\}$

Again, in words, $t(C, \mathcal{T}, n)$ restricts C to the elements of n-components of $\mathcal{T}$ relationships. So, then, $t(C_x, \mathcal{T}, 1)$ consists of those elements of C_x that are part of a subgroup of C_x that enjoy an $\mathcal{R}$ relationship to a subgroup of C_y. Meanwhile, $t(C_y, \mathcal{T}, 2)$ consists of those elements of C_y that are part of a subgroup of C_y that a subgroup of C_x enjoys an $\mathcal{R}$ relationship to.

The key observation here is that $\mathcal{T}$ implicitly defines an $\mathcal{R}$-cover of $\langle t(C_x, \mathcal{T}, 1), t(C_y, \mathcal{T}, 2)\rangle$. To see this, consider any tuple $\boldsymbol{R} \in \mathcal{T}$. We must have $R_x \subseteq t(C_x, \mathcal{T}, 1)$ (and likewise for y). For, given any element $u \in R_x$, by (21), we have $u \in C_x$ and therefore by (22) we have $u \in t(C_x, \mathcal{T}, 1)$ (using R_x itself as the needed witness T_n). Meanwhile, of course, there can be no $u \in t(C_x, \mathcal{T}, 1)$ without some $\boldsymbol{R} \in \mathcal{T}$ and $u \in R_x$.

Abstractly, it is convenient to generalize the notation introduced in (21) and (22) to handle relations of arbitrary arity. The relevant definitions are given in (23).

```
Algorithm: Plural Relaxation
Input:  Initial variable values $\boldsymbol{C}'$
        Initial relations $\{\mathcal{R}_i\}$
Output: Final variable values $\boldsymbol{C}'$
        Final relations $\{\mathcal{R}_i\}$
Steps:  do {
            $\boldsymbol{C} \leftarrow \boldsymbol{C}'$
            loop over $\mathcal{R}_i$ {
                $\mathcal{R}_i \leftarrow l(\mathcal{R}_i, \boldsymbol{C}')$
                $\boldsymbol{C}' \leftarrow t(\boldsymbol{C}', \mathcal{R}_i)$ }
        } until $\boldsymbol{C}' = \boldsymbol{C}$
```

Figure 2: Constraint processing for plurals

(23) a $l(\mathcal{R}, \boldsymbol{C}) := \{\boldsymbol{R} \in \mathcal{R}$: for each variable n of $\mathcal{R}$, $R_n \subseteq C_n\}$
b $t(\boldsymbol{C}, \mathcal{R}) := \langle C'_1, \ldots C'_k \rangle$, where each $C'_i = t(C_i, \mathcal{R}, i)$ if i is a variable of $\mathcal{R}$ (and C_i otherwise).

These notions induce a constraint-satisfaction heuristic for plural constraints that mirrors the relaxation algorithm for singular constraints. The algorithm is given in Figure 2. The algorithm works by eliminating possible values for variables (and eliminating relevant tuples from the constraints) until a fixed-point is reached. Since some value for a variable must be discarded on each iteration through the main loop, there can be no more iterations than candidate values (roughly the product of the size of the context set and the number of variables). Note that the overall polynomial complexity of a singular relaxation algorithm cannot be guaranteed without some assumptions about the representation of the constraint sets $\mathcal{R}_i$—since these relate sets, not individuals, they may require a number of tuples that grows exponentially in the size of the context set. For input constraints $L = \{R_i(\boldsymbol{x})\}$ and initial values $\boldsymbol{D}$ (or the context set S by default), we use $P(L, \boldsymbol{D})$ (or just $P(L)$) to refer to the PLURAL constraint network computed by this algorithm.

Informally, since the algorithm is conservative about discarding values for variables, all values that actually satisfy the constraints survive the execution of the algorithm. Among other things, this ensures a maximality interpretation for noun phrases modeled using this algorithm. To justify this formally, consider the computation of $P(L, \boldsymbol{C}')$. Denote by $\boldsymbol{F}$ the final variable values output. Suppose $\boldsymbol{V}$ is tuple of sets with two properties: first, for each $\mathcal{R}_i$ and for each $u \in V_j$, there is a tuple $R \in \mathcal{R}_i$ with $u \in R_j$ and moreover $R_k \subseteq V_k$ for each variable k governed by $\mathcal{R}_i$; second, $V_j \subseteq C'_j$. This means that $\boldsymbol{V}$ represents a possible solution to the constraints given by L over the domains $\boldsymbol{C}'$. Under these assumptions, it follows that $V_j \subseteq F_j$. To show this, it suffices to show that any time our two conditions on V_j hold at the beginning of the inner loop of the algorithm, they hold at the end of the loop. So assume the conditions are true at the beginning, and let's consider the conditions at the end. Consider $u \in V_j$ and its witness tuple from the hypothesis, R. Since $R_k \subseteq V_k \subseteq C'_k$, $\boldsymbol{R} \in l(\mathcal{R}_i, \boldsymbol{C}')$ too. This establishes the first condition. Moreover, since $V_j \subseteq C'_j$, and $u \in R_j$, $u \in t(C'_j, l(\mathcal{R}_i, \boldsymbol{C}'), j)$. This establishes the second condition.

3.2 Representing Constraints

The operations of (23) and the algorithm of Figure 2 provide an abstract framework for computing the interpretation of plural noun phrases. In some cases, the simple representations that these definitions suggest may be suitable for direct use in implementation. For example, inherently distributive predicates and relations, like *pig*, *black*, *stall* and *open*, can be represented compactly as a list of tuples. There will not be inordinately many tuples because each tuple relates only singleton sets. In the example scene of Figure 1, for example, we could enumerate:

(24) a $pig = \{\{a_1\}, \{a_3\}, \{a_4\}, \{a_5\}, \{a_7\}, \{a_8\}\}$
b $black = \{\{a_4\}, \{a_5\}, \{a_6\}, \{a_8\}\}$
c $stall = \{\{s_1\}, \{s_2\}, \{s_3\}, \{s_4\}, \{s_5\}, \{s_6\}\}$
d $open = \{\{s_1\}, \{s_2\}, \{s_3\}\}$

However, it will not always be so easy. Take the property of *having something inside*, for example. This is a collective property—and a collection Y has an entity X inside as long as the space collectively carved out by Y includes the space taken up by X. Listing this relation is hopeless; in Figure 1, we can pick, for instance, any set Y including s_1 and find that $in(\{a_1\},Y)$ holds.

In this case, the right strategy seems to be to reason about general containment tractably by maintaining an intermediate relation of IMMEDIATE CONTAINMENT which I will call $in_*(x,y)$. This relation holds between a singleton X and a collection Y when Y contains X but no subset of Y contains X. We can list tuples for in_* in the scene of Figure 1 straightforwardly:

(25) $\{\langle\{a_1\},\{s_1\}\rangle,\langle\{a_2\},\{s_2\}\rangle,\langle\{a_3\},\{s_3\}\rangle,\langle\{a_4\},\{s_4\}\rangle,\langle\{a_5\},\{s_5\}\rangle,\langle\{a_6\},\{s_6\}\rangle\}$

(In fact, the non-overlapping occupation of three-dimensional space by solid objects might be expected to keep this relation concise in general.)

Now we can adopt a meaning postulate to relate the constraint $in(x,y)$ to the constraint $in_*(x,y)$:

(26) $in(x,y) \equiv \exists c.in_*(x,c) \wedge c \subseteq y$

This equivalence ensures that $\langle p,q\rangle \in l(in(x,y),\langle C_x,C_y\rangle)$ just in case $\langle p,c\rangle \in l(in_*(x,y),\langle C_x,C_y\rangle)$ and $c \subseteq y \subseteq C_y$. More importantly, $t(C_x,in(x,y),x) = t(C_x,in_*(x,y),x)$; similarly, $t(C_y,in(x,y),y) = C_y$ as long as the $in_*(x,y)$ relation is not empty on $\langle C_x,C_y\rangle$ (in which case $t(C_y,in(x,y),y)$ is also empty). So, we have a natural way of using our concise relation in_* as a proxy for updating a constraint network using the explosive relation *in*.

Clearly, this construction depends on the meaning of *in*; different constructions will be needed to encode the lexical semantics of different words. As a question of cognitive science, we might expect that the lexical concepts that people avail themselves of in identifying objects to one another admit an efficient computational treatment. However, whether the present framework allows such representations in general remains an important question for future research.

3.3 Search for Referring Expressions

By exploiting the representations for lexical constraints suggested in section 3.2 and the algorithm of section 3.1 for keeping track of the interpretation of a plural referring expression, we can carry over the presentation of referring expression construction as a search task from section 1.2 directly to the plural setting. This section outlines this result.

In the plural referring expression search task, each state takes the form

(27) $\Sigma : \langle L, \boldsymbol{x}, \boldsymbol{V}, P(L)\rangle$

L is a description providing a set of constraints on the free variables listed in $\boldsymbol{x}$. $\boldsymbol{V}$ is a tuple of sets recording the intended referents of the description: V_i gives the collection that the description aims to identify with x_i. Finally, $P(L)$ is a plural constraint network, maintained as in sections 3.1 and 3.2, which describes the entities that the description could refer to—and, indeed, that the description would refer to if uttered in its current (possibly incomplete) form.

Again, the initial state is constructed from an empty description E and an intention to identify a set V_x as the value of a variable x:

(28) $\Sigma_0 : \langle E, \langle x\rangle, \langle V_x\rangle, P(E)\rangle$

And again a goal state is one in which the constraint network $P(L)$ associates each variable x_i with V_i as its set of possible values: this is the case where the constraints identify the intended values for each of the variables in the description.

A transition from one state to another is accomplished by adding a constraint to the description, and at the same time possibly taking on new variables and corresponding new intended referents. We continue to assume that the grammar and knowledge base provide these constraints in the form $p = R(\boldsymbol{x};\boldsymbol{y})$, where syntactic combination determines some coindexation between arguments of R and the existing variables $\boldsymbol{x}$ of the description.

The semantic condition which determines whether a particular relation is appropriate is reformulated to reflect the cover semantics for plural predication. To choose $R(\boldsymbol{x};\boldsymbol{y})$ we must find a set of tuples S which stand in the relation $R(\boldsymbol{x};\boldsymbol{y})$ according to shared knowledge, and which cover the intended referents of the description. Formally, for each old variable x_j constrained by $R(\boldsymbol{x};\boldsymbol{y})$, we must have

(29) $\quad V_j = \bigcup\{R_j : \boldsymbol{R} \in S\}$

The selection of these tuples S allows us to determine appropriate intended referents for the new variables $\boldsymbol{y}$ introduced by the constraint. For each y_i, we define $V'_i = \bigcup\{R_i : \boldsymbol{R} \in S\}$. Thus under these conditions we can describe the new state $\Sigma \oplus \langle p, \boldsymbol{V}' \rangle$ obtained by adding the constraint p (intended to refer to $\boldsymbol{V}'$) to Σ as in (30).

(30) $\quad \Sigma \oplus \langle p, \boldsymbol{V}' \rangle := \langle L \cup \{p\}, \boldsymbol{xy}, \boldsymbol{VV}', P(L \cup \{p\}) \rangle$

To get a flavor for the structure of this search space, consider *the black pigs in the stalls*—example (8a), interpreted as before against Figure 1. We can now trace a path through the search space which yields this successful description.

The path begins with an initial state in which we have not yet added any constraints to the description, but we intend to use the variable x to identify the black pigs in the stalls—the set B defined as $\{a_4, a_5\}$. The empty description induces a constraint network in which x may take on any value in the context set S. So we have:

(31) $\quad \langle E, \langle x \rangle, \langle B \rangle, [x = S] \rangle$

The next step adds the constraint $pig(x)$ corresponding to the lexical item *pigs*; the result is to narrow the possibilities for x in the constraint network to just the set of pigs in the context set. We get:

(32) $\quad \langle \{pig(x)\}, \langle x \rangle, \langle B \rangle, [x = \{a_1, a_3, a_4, a_5, a_7, a_8\}] \rangle$

Now we can add the constraint *in*(x,y) corresponding to the word *in*—choosing as the intended reference for y the set of stalls Z, namely $\{s_1, s_2, s_3, s_4, s_5, s_6\}$. At this stage the constraint processing rules out the two pigs a_7 and a_8 that aren't *in* anything. The result is:

(33) $\quad \langle \{pig(x), in(x,y)\}, \langle x,y \rangle, \langle B, Z \rangle, [x = \{a_1, a_3, a_4, a_5\}, y = S] \rangle$

When next we add the constraint $stall(y)$ corresponding to the word *stalls*, the variables, values and constraint networks are updated as expected, narrowing down y to the stalls:

(34) $\quad \langle \{pig(x), in(x,y), stall(y)\}, \langle x,y \rangle, \langle B, Z \rangle, [x = \{a_1, a_3, a_4, a_5\}, y = Z] \rangle$

Finally, we can add the constraint $black(x)$ (corresponding to the word *black*) to yield a description that in fact identifies its intended referents:

(35) $\quad \langle \{black(x), pig(x), in(x,y), stall(y)\}, \langle x,y \rangle, \langle B, Z \rangle, [x = B, y = Z] \rangle$

3.4 Making Search Flexible

The derivation of *the black pigs in the stalls* which is sketched in (31)–(35) illustrates the structure of search space for plural descriptions—for better and for worse. A particular difficulty is the nondeterminism involved in choosing intended referents for fresh variables in constraints—for example, in selecting the set Z to serve as the value for y in the constraint $in(x,y)$ introduced into state (33). The alternative description *the pigs in the open stalls* shows that we could have ended up with a concise referring expression while selecting the different set $Z' = \{s_4, s_5, s_6\}$ in place of Z.

In fact, we can now see how large a fanout in the search space accompanies the choice of the constraint $in(x,y)$. We can choose any set Y and any set of tuples S of the *in* relation, as long as $\{a_4, a_5\} = \bigcup\{R_x : \boldsymbol{R} \in S\}$ and as long as $Y = \bigcup\{R_y : \boldsymbol{R} \in S\}$. Given our interpretation of the *in* relation, we can choose for Y any subset of the context set S containing s_4 and s_5. The two particularly notable values for Y, Z and Z', are just the tip of the iceberg. For an efficient implementation, it seems problematic to select one of these values for Y at random, without any knowledge of the role Y—and constraints placed on Y—could play in a completed description.

Suppose we rethink the representation of intended referents in the search space as follows. We give a partial specification $\boldsymbol{I}$ defining intended collections only for the subset of the variables in the description that we are committed to. In many cases, that may mean specifying only the overall referent featured in the initial referential goal and evoked by the head noun of the referring expression. The remainder of the intended referents we compute as whatever salient entities happen to be compatible with the description and with reference to $\boldsymbol{I}$, using a constraint network. That is, we compute $\boldsymbol{V}$ as $P(L, \boldsymbol{I})$—leading to the representation of states of (36).

(36) $\Sigma = \langle L, \boldsymbol{x}, \boldsymbol{I}, P(L, \boldsymbol{I}), P(L)\rangle$

The description refers successfully when each variable x is associated with the same set V_x in both the constraint network $P(L, \boldsymbol{I})$ and the constraint network $P(L)$. Again this simply means that the hearer's interpretation of the referring expression is the speaker's intended interpretation.

Now, how do we determine semantically whether a new constraint $R(\boldsymbol{x};\boldsymbol{y})$—interpreted by a set of tuples R —represents a viable addition to the description in some state Σ? It is no longer necessary to find $S \subseteq R$ such that $V_x = \bigcup\{R_x : \boldsymbol{R} \in S\}$ for each old variable x involved in the constraint. We may have $\bigcup\{R_x : \boldsymbol{R} \in S\} \subset V_x$, if we are prepared to narrow down the entities we might refer to with variable x in the interest of elaborating an overall description that is more concise. Of course, we cannot rule out any elements specified in $\boldsymbol{I}$—either directly, because they do not satisfy $R(\boldsymbol{x};\boldsymbol{y})$, or indirectly, because $R(\boldsymbol{x};\boldsymbol{y})$ and some other constraints together rule out reference to them. It transpires that the way to test the applicability on this approach of a constraint $p = R(\boldsymbol{x};\boldsymbol{y})$ is by computing a new constraint network $\boldsymbol{V}' = P(L \cup \{p\}, \boldsymbol{I})$ and ensuring, wherever $\boldsymbol{I}$ assigns a value to x, that $I_x \subseteq V'_x$.

If this test admits a new constraint $p = R(\boldsymbol{x};\boldsymbol{y})$, the new state obtained by adding p to the description in state Σ is computed as in (37).

(37) $\Sigma \oplus p := \langle L \cup \{p\}, \boldsymbol{xy}, \boldsymbol{I}, P(L \cup \{p\}, \boldsymbol{I}), P(L \cup \{p\})\rangle$

Note that we have gone from $\Sigma \oplus \langle p, \boldsymbol{V}'\rangle$ to $\Sigma \oplus p$. No specification of new intended referents is required in this account, since these are determined implicitly by constraint propagation!

Let us compare the search space with this more flexible representation with the basic search space defined in section 3.3. Consider again *the black pigs in the stalls*. The initial state is now:

(38) $\langle E, \langle x\rangle, \langle B\rangle, [x = B], [x = S]\rangle$

We add *pigs* to get:

(39) $\langle \{pig(x)\}, \langle x\rangle, \langle B\rangle, [x = B], [x = \{a_1, a_3, a_4, a_5, a_7, a_8\}]\rangle$

We add *in* to get:

(40) $\langle \{pig(x), in(x,y)\}, \langle x,y\rangle, \langle B\rangle, [x = B, y = S], [x = \{a_1, a_3, a_4, a_5\}, y = S]\rangle$

Adding *stalls* gives:

(41) $\langle\{pig(x), in(x,y), stall(y)\}, \langle x,y\rangle, \langle B\rangle, [x=B, y=Z], [x=\{a_1,a_3,a_4,a_5\}, y=Z]\rangle$

Now we can add *black*, describing x, to give a completed description as encoded by the state (42).

(42) $\langle\{black(x), pig(x), in(x,y), stall(y)\}, \langle x,y\rangle, \langle B\rangle, [x=B, y=Z], [x=B, y=Z]\rangle$

But we can ALSO add *open* to describe the stalls—thereby narrowing both our intended and our anticipated reference for y. The result is another completed description: *the pigs in the open stalls*.

(43) $\langle\{pig(x), in(x,y), open(y), stall(y)\}, \langle x,y\rangle, \langle B\rangle, [x=B, y=Z'], [x=B, y=Z']\rangle$

The fact that we are no longer required to guess intended referents in advance (from a potentially exponential space) makes this revised search problem much more computationally attractive.

4 Conclusion

Any algorithm for constructing identifying descriptions of sets of discourse entities must avoid making an inventory of the possible sets of of discourse entities. This paper has suggested reinterpreting and extending the algorithms and data structures of singular referring expression generation to pluralities. Using covers to abstract collective and distributive readings—and using sets of assignments to represent plural references—yields a search problem for set-identifying expressions which largely mirrors that for singulars, and which avoids computation and search over sets of collections. Along the way, I have pointed out a number of further problems for plural computational lexical semantics raised by this avenue of research.

Acknowledgments

This paper benefits from discussions with Roger Schwarzschild and with the generation working group of the University of Edinburgh, and was made possible by a postdoctoral fellowship from RuCCS.

References

[Appelt, 1985] Appelt, D. (1985). *Planning English Sentences*. Cambridge University Press, Cambridge England.

[Dale, 1992] Dale, R. (1992). *Generating Referring Expressions: Constructing Descriptions in a Domain of Objects and Processes*. MIT Press, Cambridge MA.

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

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

[Dalianis, 1996] Dalianis, H. (1996). *Concise Natural Language Generation from Formal Specifications*. PhD thesis, Royal Institute of Technology, Stockholm. Department of Computer and Systems Sciences.

[Donellan, 1966] Donellan, K. (1966). Reference and definite description. *Philosophical Review*, 75:281–304.

[Gillon, 1987] Gillon, B. (1987). The readings of plural noun phrases in english. *Linguistics and Philosophy*, 10(2):199–299.

[Green et al., 1998a] Green, N., Carenini, G., Kerpedjiev, S., Roth, S., and Moore, J. (1998a). A media-independent content language for integrated text and graphics generation. In *CVIR '98 – Workshop on Content Visualization and Intermedia Representations*.

[Green et al., 1998b] Green, N., Carenini, G., and Moore, J. (1998b). A principled representation of attributive descriptions for generating integrated text and information graphics presentations. In *Proceedings of International Natural Language Generation Workshop*, pages 18–27.

[Horacek, 1996] Horacek, H. (1996). A new algorithm for generating referring expressions. In *ECAI 8*, pages 577–581.

[Kamp and Reyle, 1993] Kamp, H. and Reyle, U. (1993). *From Discourse to Logic: Introduction to Modeltheoretic Semantics of Natural Language, Formal Logic and Discourse Representation Theory*. Kluwer, Boston.

[Kronfeld, 1986] Kronfeld, A. (1986). Donellan's distinction and a computational model of reference. In *Proceedings of ACL*, pages 186–191.

[Mackworth, 1987] Mackworth, A. (1987). Constraint Satisfaction. In Shapiro, S., editor, *Encyclopedia of Artificial Intelligence*, pages 205–211. John Wiley and Sons.

[Moore, 1994] Moore, J. (1994). *Participating in Explanatory Dialogues*. MIT Press, Cambridge MA.

[Moore and Paris, 1993] Moore, J. D. and Paris, C. L. (1993). Planning text for advisory dialogues: capturing intentional and rhetorical information. *Computational Linguistics*, 19(4):651–695.

[Rambow and Korelsky, 1992] Rambow, O. and Korelsky, T. (1992). Applied text generation. In *ANLP*, pages 40–47.

[Reiter, 1994] Reiter, E. (1994). has a consensus NL generation architecture appeared, and is it psycholinguistically plausible? In *Seventh International Workshop on Natural Language Generation*, pages 163–170.

[Schwarzschild, 1994] Schwarzschild, R. (1994). Plurals, presuppositions, and the sources of distributivity. *Natural Language Semantics*, 2:201–248.

[Schwarzschild, 1996] Schwarzschild, R. (1996). *Pluralities*. Kluwer, Dordrecht.

[Stone and Doran, 1997] Stone, M. and Doran, C. (1997). Sentence planning as description using tree-adjoining grammar. In *Proceedings of ACL*, pages 198–205.

[van den Berg, 1993] van den Berg, M. H. (1993). Full dynamic plural logic. In Bimbó, K. and Máté, A., editors, *Proceedings of the Fourth Symposium on Logic and Language*, Budapest.

[van den Berg, 1996] van den Berg, M. H. (1996). Generalized dynamic quantifiers. In van der Does, J. and van Eijk, J., editors, *Quantifiers, Logic and Language*. CSLI.

[van Eijck, 1983] van Eijck, J. (1983). Discourse representation theory and plurality. In ter Meulen, A., editor, *Studies in Modeltheoretic Semantics*, GRASS 1. Foris, Dordrecht.

[Verkuyl and van der Does, 1991] Verkuyl, H. and van der Does, J. (1991). The semantics of plural noun phrases. Preprint, ITLI, Amsterdam.