Generating Referring Quantified Expressions

James Shaw and **Kathleen McKeown**
Dept. of Computer Science
Columbia University
New York, NY 10027, USA
shaw,kathy@cs.columbia.edu

Abstract

In this paper, we describe how quantifiers can be generated in a text generation system. By taking advantage of discourse and ontological information, quantified expressions can replace entities in a text, making the text more fluent and concise. In addition to avoiding ambiguities between distributive and collective readings in universal quantification generation, we will also show how different scope orderings between universal and existential quantifiers will result in different quantified expressions in our algorithm.

1 Introduction

To convey information concisely and fluently, text generation systems often perform opportunistic text planning (Robin, 1995; Mellish et al., 1998) and employ advanced linguistic constructions such as ellipsis (Shaw, 1998). But a system can also take advantage of quantification and ontological information to generate concise references to entities at the discourse level. For example, a sentence such as *"The patient has an infusion line in each arm"* is a more concise version of *"The patient has an infusion line in his left arm. The patient has an infusion line in his right arm."* Quantification is an active research topic in logic, language, and philosophy(Carpenter, 1997; de Swart, 1998). Since natural language understanding systems need to obtain as few interpretations as possible from text, researchers have studied quantifier scope ambiguity extensively (Woods, 1978; Grosz et al., 1987; Hobbs and Shieber, 1987; Pereira, 1990; Moran and Pereira, 1992; Park, 1995). Research in quantification interpretation first transforms a sentence into predicate logic, raises the quantifiers to the sentential level, and permutes these quantifiers to obtain as many readings as possible related to quantifier scoping. Then, invalid readings are eliminated using various constraints.

Ambiguity in quantified expressions is caused by two main culprits. The first type of ambiguity involves the distributive reading versus the collective reading. In universal quantification, a referring expression refers to multiple entities. There is a potential ambiguity between whether the aggregated entities acted individually (distributive) or acted together as one (collective). Under the distributive reading, the sentence "All the nurses inspected the patient." implies that each nurse individually inspected the patient. Under the collective reading, the nurses inspected the patient together as a group. The other ambiguity in quantification involves multiple quantifiers in the same sentence. The sentence "A nurse inspected each patient." has two possible quantifier scope orderings. In $\forall$patient$\exists$nurse, the universal quantifier $\forall$ has wide scope, outscoping the existential quantifier $\exists$. This ordering means that each patient is inspected by a nurse, who might not be the same in each case. In the other scope order, $\exists$nurse$\forall$patient, a single, particular nurse inspected every patient. In both types of ambiguities, a generation system should make the desired reading clear.

Fortunately, the difficulties of quantifier scope disambiguation faced by the understanding community do not apply to text generation. For generation, the problem is the reverse: given an unambiguous representation of a set of facts as input, how can it generate a quantified sentence that unambiguously conveys the intended meaning? In this paper, we propose an algorithm which selects an appropriate quantified expression to refer to a set of entities using discourse and ontological knowledge. The algorithm first identifies the entities for quantification in the input propositions. Then an appropriate concept in the ontology is selected to refer to these entities. Using discourse and ontological information, the system determines if quantification is appropriate and if it is, which particular quantifier to use to minimize the ambiguity between distributive and collective readings. More importantly, when there are multiple quantifiers in the same sentence, the algorithm generates different expressions for different scope orderings. In this work, we focus on generating referring quantified expressions for entities which have been mentioned before in the discourse or can be inferred from an ontology. There are quantified

expressions that do not refer to particular entities in a domain or discourse, such as generics (i.e. "*All whales* are mammals."), or negatives (i.e., "The patient has *no allergies*."). The synthesis of such quantifiers is currently performed in earlier stages of the generation process.

In the next section we will compare our approach with previous work in the generation of quantified expressions. In Section 3, we will describe the application where the need for concise output motivated our research in quantification. The algorithm for generating universal quantifiers is detailed in Section 4, including how the system handles ambiguity between distributive and collective readings. Section 5 describes how our algorithm generates sentences with multiple quantifiers.

2 Related Work

Because a quantified expression refers to multiple entities in a domain, our work can be categorized as referring expression generation (Dale, 1992; Reiter and Dale, 1992; Horacek, 1997). Previous work in this area did not address the generation of quantified expressions directly. In this paper, we are interested in how to systematically derive quantifiers from input propositions, discourse history, and ontological information. Recent work on the generation of quantifiers (Gailly, 1988; Creaney, 1996; Creaney, 1999) follows the analysis viewpoint, discussing scope ambiguities extensively. Though our algorithm generates different sentences for different scope orderings, we do not achieve this through scoping operations as they did. Creaney also discussed various imprecise quantifiers, such as `some`, `at least`, and `at most`. In regards to generating generic quantified expressions, (Knott et al., 1997) has proposed an algorithm for generating defeasible, but informative descriptions for objects in museums.

Other researchers (van Eijck and Alshawi, 1992; Copestake et al., 1999) proposed representations in a machine translation setting which allow underspecification in regard to quantifier scope. Our work is different in that we perform quantification directly on the instance-based representation obtained from database tuples. Our input does not have the information about which entities are quantified as is the case in machine translation, where the quantifiers are already specified in the input from a source language.

3 The Application Domain

We implemented our quantification algorithm as part of MAGIC (Dalal et al., 1996; McKeown et al., 1997). MAGIC automatically generates multimedia briefings to describe the post-operative status of a patient after undergoing Coronary Artery Bypass Graft surgery. The system embodies a standard text generation system architecture with three modules (Rambow and Korelsky, 1992): a content planner, a sentence planner, and a linguistic realizer. Once the bypass surgery is finished, information that is automatically collected during surgery such as blood pressure, heart rate, and medications given, is sent to a domain-specific medical inference module. Based on the medical inferences and schemas (McKeown, 1985), the content planner determines the information to convey and the order to convey it.

```
((TYPE EVENT)
 (PRED ((PRED receive) (ID id1)))
 (ARG1 ((PRED patient) (ID pt1)))
 (ARG2 ((PRED aprotinin) (ID ap1)))
 (MODS ((PRED after) (ID id2)
        (TYPE TIME)
        (ARG2 ((PRED critical-point)
               (NAME intubation) (ID c1)))
))))
```

Figure 1: The predicate-argument structure of *"After intubation, a patient received aprotinin."*

The sentence planner takes a set of propositions (or predicate-argument structures) with rhetorical relations from the content planner and uses linguistic information to make decisions about how to convey the propositions fluently. Each proposition is represented as a feature structure (Kaplan and Bresnan, 1982; Kay, 1979) similar to the one shown in Figure 1. The sentence planner's responsibilities include referring expression generation, clause aggregation, and lexical choice (Wanner and Hovy, 1996). Then the aggregated predicate-argument structure is sent to FUF/SURGE (Elhadad and Robin, 1992), a linguistic realizer which transforms the lexicalized semantic specification into a string. The quantification algorithm is implemented in the sentence planner.

4 Quantification Algorithm

In this work, we prefer generating expressions with universal quantifiers over conjunction because, assuming that the users and the system have the same domain model, the universally quantified expressions are more concise and they represent the same amount of information as the expression with conjoined entities. In contrast, when given a conjunction of entities and an expression with a cardinal quantifier, the system, by default, would use the conjunction if the conjoined entities can be distinguished at the surface level. This is because once the system generates a cardinal quantifier when the universal quantification does not hold, such as "three

patients", it is impossible for the hearer to recover the identities of these patients based on the context. The default heuristics to prefer universal quantifier over conjunction over cardinal quantifier can be superseded by directives from the content planner which are application specific.

The input to our quantification algorithm is a set of predicate-argument structures after the referring expression module selected the properties to identify the entities (Dale, 1992; Dale and Reiter, 1995), but without carrying out the assignment of quantifiers. Our quantification algorithm first identifies the set of distinct entities which can be quantified in the input propositions. A *generalization* of the entities in the ontology is selected to potentially replace the references to these entities. If universal quantification is possible, then the replacement is made and the system must select which particular quantifier to use. In our system, we have six realizations for universal quantifiers: `each`, `every`, `all`[1], `both`, `the`, and `any`, and two for existential quantifiers: the indefinite article, `a/an`, and cardinal `n`.

4.1 Identify Thematic Roles with Distinct Entities

Our algorithm identifies the roles containing distinct entities among the input propositions as candidates for universal and existential quantification. Suppose the system is given two propositions similar to the one in Figure 1, "After intubation, Alice received aprotinin" and "After start of bypass, Alice received aprotinin", each with four roles – PRED, ARG1, ARG2, and MODS-TIME. By computing similarity among entities in the same role, the system determines that the entities in ARG1, PRED, and ARG2 are identical in each role, and only the entities in MODS-TIME are different. Based on this result, the distinct entities in MODS-TIME, "after intubation" and "after start of bypass", are candidates for quantification.

4.2 Generalization and Quantification

We used the axioms in Figure 2 to determine if the distinct entities can be universally or existentially quantified. Though the axioms are similar to those used in Generalized Quantifier (Barwise and Cooper, 1981; Zwarts, 1983; de Swart, 1998), the semantics of set X and set D are different. In the previous step, the entities in set X have been identified. To compute set D in Figure 2, we introduce a concept, Class-X. Class-X is a *generalization* of the distinct entities in set X. Quantification can replace the distinct entities in the propositions with a reference to their type restricted by a quantifier, accessing discourse and ontological information to provide a context. Our ontology is implemented in

[1] `all` is realized as "`all the`".

- `both`: $|D - X| = 0$ and $|X| = 2$, can have collective reading
- `every`, `all`, `the`: $|D - X| = 0$ and $|X| > 2$, can have collective reading
- `each`: $|D - X| = 0$ and $|X| \geq 2$, only distributive reading
- `any`: $|D - X| = 0$, when under the scope of negation
- `a/an`: $|D \cap X| > 0$ and $|X| = 1$
- `n` (cardinal): $|D \cap X| > 0$ and $|X| = n$

Figure 2: Axioms of the quantifiers discussed in this paper.

CLASSIC(Borgida et al., 1989) and is a subset of WordNet(Miller et al., 1990) and an online medical dictionary (Cimino et al., 1994) designed to support multiple applications across the medical institution. Given the entities in set X, queries in CLASSIC determine the class of each instance and its ancestors in the ontology. Based on this information, the generalization algorithm identifies Class-X by computing the most specific class which covers all the entities. Earlier work (Passonneau et al., 1996) provided a framework for balancing specificity and verbosity in selecting appropriate concepts for generalization. However, given the precision needed in medical reports, our generalization procedure selects the most specific class.

Set D represents the set of instances of Class-X in a context. Our system currently computes set D for three different contexts:

- discourse: Previous references can provide an appropriate context for universal quantification. For example, if "Alice" and "Bob" were mentioned in the previous sentence, the system can refer to them as "both patients" in the current sentence.
- domain ontology: The domain ontology provides a closed world from which we can obtain the set D by matching all the instances of a concept in the knowledge base, such as "every patient". In addition, certain concepts in the ontology have limited types. For example, knowing that cell savers, platelets and packed red blood cells are the only possible types of blood products in the ontology, the quantified expression "every blood product" can be used instead of referring to each entity.
- domain knowledge: The possessor of the distinct entities in a role might contain a maximum number of instances allowed for Class-X. For ex-

ample, because a person has only two arms, the entities "the patient's left arm" and "the patient's right arm" can be referred to as "each arm".

The computation of set D can also involve interactions with a referring expression module(Dale and Reiter, 1995). For example, instead of the expression "Alice and Bob" and "both patients" covered by the current algorithm, by interacting with a referring expression module, the system might determine that "both CABG patients operated on this morning by Dr. Rose" is a clearer expression to refer to the entities. Though this is desirable, we did not incorporate this capability into our system.

Although `the` is often used to indicate a generic reference (i.e., "*The* lion is the king of jungle."), in English, `the` can also be used as an unmarked universal quantifier when its head noun is plural, such as "the patients." Like the quantifier `all`, `the` can be both distributive and collective. However, `the` cannot always replace `all` as a universal quantifier. `the` cannot be used when universal quantification is based on the domain ontology. For example, it is not obvious that the quantified expression in "John received the blood products." refers to "each blood product" in the ontology. Although unmarked universal quantifiers can be used to refer to body parts, as in "The lines include an IV in the arms.", the expression is ambiguous between the distributive and collective readings. Of the three contexts discussed above, the system occationally generates `the` instead of `every` and `both` in a discourse context, yielding more natural output.

When the computed set D matches set X exactly ($|D - X| = 0$), a quantified expression with either `each`, `all`, `every`, `both`, `the`, and `any`, replaces the entities in set X.

4.3 Selecting a Particular Quantifier

In general, the universal quantification of a particular type of entity, such as "every patient", refers to all such entities in a context. As a result, readers can recover what a universally quantified expression refers to. In contrast, readers cannot pinpoint which entity has been referred to in an existentially quantified expression, such as "a patient" or "two patients". Because a universally quantified expression preserves original semantics and is more concise than listing each entity, it is the focus of our quantification algorithm. The universal quantifiers implemented in our system include the six possible realizations of $\forall$ in English: `every`, `all`, `each`, `both`, `the`, and `any`. The only existential quantifiers implemented in our system are the singular indefinite quantifier, `a/an`, and cardinal quantifiers, `n`. They are used in sentences with multiple quantifiers and when the entities being referred to do not have distinguishable expressions at surface level. A more developed pragmatic module is needed before quantifiers such as `some`, `most`, `at least`, and `few`, can be systematically generated. Indiscriminate application of imprecise quantification can result in vague or inappropriate text in our domain, such as *"The patient received **some** blood products."* In our application, knowing exactly what blood products are used is very important. To avoid generating such inappropriate sentences, the system only performs generalization on the entities which can be universally quantified. If the distinct entities cannot be universally quantified, the system will realize these entities using coordinated conjunction.

Once the system decides that a universally quantified expression can be used to replace the entities in set X, it must select which universal quantifier. Because our sentence planner opportunistically combines distinct entries from separate database entries for conciseness, it is not the case that these aggregated entities acted together (the collective reading). Given such input, the referring expression for aggregated entities should have only the distributive reading[2]. The universal quantifier, `each`, always imposes a distributive reading when applied. In general, `each` requires a "matching" between the domain of the quantifier and the objects referred to(McCawley, 1981, pp. 37). In our algorithm, this matching process is exactly what happened, thus it is the default universal quantifier in our algorithm. Of course, indiscriminate use of `each` can result in awkward sounding text. For example, the sentence *"Every patient is awake"* sounds more natural than *"Each patient is awake."* However, since quantified expressions with the universal quantifiers `all` and `every`[3] can have collective readings (Vendler, 1967; McCawley, 1981), our system generates `every` and `all` under two conditions when the collective reading is unlikely. First if the proposition is a state, as opposed to an event, we assume only the distributive reading is possible[4]. The quantifier `every` is used in *"Every patient had tachycardia."* because the proposition is a state proposition and contains the predicate *has-attribute*, an attributive relation.

[2]For our system to generate noun phrases with collective readings, the quantification process must be performed at the content planner level, not in the clause aggregation module.

[3]`every` is also distributive, but it stresses completeness or rather, exhaustiveness(Vendler, 1967). The sentence *"John took a picture of everyone in the room."* is ambiguous while *"John took a picture of each person in the room."* is not.

[4]There are cases where state propositions do have distributed readings (e.g., "Mountains surround the village."). Sentences with collective readings are handled earlier in the content planner and thus, this type of problem does not occur at this point in our system. Though this observation seems to be true in our medical application, when implementing quantifiers in a new domain, we can limit this assumption to only the subset of state relations for which it holds.

Second, when the concept being universally quantified is marked as having a distributive reading in the lexicon, such as the concept `episode`, quantifiers `every` will be used instead of `each`. These quantifiers make the quantified sentences more natural because they do not pick out the redundant distributive meaning.

The use of prepositions can also affect which quantifier to use. For example, "**After** all the episodes, the patient received dobutamine" is ambiguous in regards to whether the dobutamine is given once during the surgery, or given after each episode. In contrast, the sentence "**In** all the episodes, the patient received dobutamine." does not have this problem. The current system looks at the particular preposition (i.e., "before", "after", or "in") before selecting the appropriate quantifier.

4.4 Examples of a Single Quantifier

Given the four propositions, "After intubation, Mrs. Doe had tachycardia", "After skin incision, Mrs. Doe had tachycardia", "After start of bypass, Mrs. Doe had tachycardia", and "After coming off bypass, Mrs. Doe had tachycardia.", the algorithm first identifies roles with similar entities, ARG1, PRED, ARG2 and removes them from further quantification processing while the distinct entities in the role MODS-TIME, "after intubation", "after skin incision", "after start of bypass", and "after coming off bypass", are further processed for universal quantification. The role MODS-TIME is further separated into two smaller roles, one role with the prepositions and the other role with different critical points. Since the prepositions are all the same, universal quantification is only applied to the distinct entities in set X, in this case, the four critical points. Queries to the CLASSIC ontology indicate that the entities in set X, "intubation", "skin-incision", "start-of-bypass", and "coming-off-bypass" match all the possible types of the concept `critical-point`, satisfying the domain ontology context in Section 4.2. Since set D and set X match exactly, generalization and universal quantification can be used to replace the references to these entities: *"After **each** critical point, Mrs. Doe had tachycardia."* The system currently does not perform generalization on entities which failed the univeral quantification test. In such cases, a sentence with conjunction will be generated, i.e., "After intubation and skin incision, Mrs. Doe had tachycardia."

In addition to `every`, the system generates `both` when the number of entities in set X is two. In our application, `both` is used as a universal quantifier under discourse context: *"Alice had episodes of bradycardia before induction and start of bypass. In **both** episodes, she received Cefazolin and Phenylephrine."*

When a universal quantifier is under the government of negation, `each`, `all`, `every` and `both` are inappropriate, and `any` should be used instead. Given that the patient went on bypass without complications, the system should generate *"The patient went on bypass without* **any** *problem."* In contrast, "*The patient went on bypass without* **every** *problem.*" has a different meaning. Our system currently uses `any` as a universal quantifier when the universal quantification is under the government of negation, such as *"The patient denied any drug allergy."*, or *"Her hypertension was controlled without any medication."* Currently, the generation of negation sentences about surgery problems and allergies are handled in the content planner. They are not synthesized from multiple negation sentences: "The patient is not allergic to aspirin. The paitent is not allergic to penicillin..."

5 Generation of Multiple Quantifiers

When there are two distinct roles across the propositions, the algorithm tries to use a universal quantifier for one role and an existential quantifier for another. To generate sentences with $\exists\exists$, both entities being referred to must have no proper names; this triggers the use of existential quantifiers. We intentionally ignore the cases where two universal quantifiers are generated in the same sentence. The likelihood for input specifying sentences with $\forall\forall$ to a text generation system is slim.

When generating multiple quantifiers in the same sentence, we differentiate between cases where there is or isn't a dependency between the two distinct roles. Two roles are independent of each other when one is not a modifier of the other. For example, the roles ARG1 and ARG2 in a proposition are independent. In *"Each patient is given a high severity rating"*, performing universal quantification on the patients (ARG3) is a separate decision from the existential quantification of the severity ratings (ARG2). Similarly, in *"An abnormal lab result was seen in each patient with hypertension after bypass"*, the quantification operations on the abnormal lab results and the patients can be performed independently.

When there is a dependency between the roles being quantified, the quantification process of each role might interact because modifiers restrict the range of the entities being modified. We found that when universal quantification occurs in the MODS role, the quantification of PRED and MODS can be performed independently, just as in the cases without dependency. Given the input propositions *"Alice has IV-1 in Alice's left arm. Alice has IV-2 in Alice's right arm."*, the distinct roles are ARG2 "IV-1" and "IV-2", and ARG2-MODS "in Alice's left arm" and "in Alice's right arm". The ARG2-MODS is universally quantified based on domain knowledge that

- Roles without dependency, ∀ Role-1,∃ Role-2
  Each patient is given a high severity rating.
- Roles without dependency, ∃ Role-1, ∀ Role-2
  An abnormal lab result was seen in each patient with hypertension after bypass.
- Roles with dependency, ∀ PRED, ∃ MODS
  Every patient with a balloon pump had hypertension.
- Roles with dependency, ∃ PRED, ∀ MODS
  Alice has an IV in each arm.

Figure 3: Sentences with two quantifiers

a patient is a human and a human has a left arm and a right arm. In this example, "an IV in each arm", the decision to generate universal and existential quantified expressions are independent. But in *"Every patient with a balloon pump had hypertension"*, the existentially quantified expression *"with a balloon pump"* is a restrictive modifier of its head. In this case, the set D does not include all the patients, but only the patients *"with a balloon pump"*. When computing set D for universal quantification, the algorithm takes this extra restriction into account by eliminating all patients without such a restriction. Once a role is universally quantified and the other is existentially quantified, our algorithm replaces both roles with the corresponding quantified expressions. Figure 3 shows the sentences with multiple quantifiers generated by applying our algorithm.

5.1 Ambiguity Revisited

In Section 4.3, we described how to minimize the ambiguity between distributive and collective readings when generating universal quantifiers. What about the scope ambiguity when there are multiple quantifiers in the same sentence? If we look at the roles which are being universally and existentially quantified in our examples in Figure 3, it is interesting to note that the universal quantifiers always have wider scope than the existential quantifiers. In the first example, the scope order is ∀patient∃high-severity-rating, the second example is ∀patient∃lab-result, the third is ∀patient∃balloon-pump, and the fourth is ∀arm∃IV. The scope orderings are all ∀∃.

What happens if a sentence contains an existential quantifier which has a wider scope than a universal quantifier? In *"A surgeon operated on each patient."*, the normal reading is ∀patient∃surgeon. But if the existentially quantified noun phrase *"a surgeon"* refers to the same surgeon, as in ∃surgeon∀patient, the system would generate *"(A particular/The same) surgeon operated on each patient."* In an applied generation system, the surgeon's name is likely to be known, and the input is likely to be "Dr. Rose operated on Alice", "Dr. Rose operated on Bob", and "Dr. Rose operated on Chris". Given these three propositions, the entities in ARG1 and PRED are identical, and only the distinct entities in ARG2, "Alice", "Bob" and "Chris", will be quantified. With an appropriate context, the sentence "Dr. Rose operated on each patient" will be generated. If the name of the surgeon is not available but the identifiers for the surgeon entities across the propositions are the same, the system will generate "The same surgeon operated on each patient." As this example indicates, when ∃ has a wider scope than ∀, the first step in our algorithm (described in Section 4.1), identifying roles with distinct entities, would eliminate the roles with identical entities from further quantification processing. Based on our algorithm, the sentences with ∃∀ readings are taken care of by the first step, identifying roles with distinct entities, while ∀∃ cases are handled by quantification operations for multiple roles, as described in Section 5.

In Section 4.3, we mentioned that it is important to know exactly what blood products are used in our application. As a result, the system would not generate the sentence "Each patient received a blood product." when the input propositions are "Alice received packed red blood cells. Bob received platelets. Chris received platelets." Even though the conjoined entities can be generalized to "blood product", this quantification operation would violate our precondition for using existential quantifiers: the descriptions for each of the conjoined entities must be indistinguishable. Here, one is "red blood cells" and the others are "platelets". Given these three propositions, the system would generate "Alice received packed red blood cells, and Bob and Chris, platelets." based on the algorithm described in (Shaw, 1998). If in our domain the input propositions could be "Alice received blood-product-1. Bob received blood-product-2. Chris received blood-product-2.", where each instance of blood-product-n could be realized as "blood product", then the system would generate "Each patient received a blood product." since the description of conjoined entities are not distinguishable at the surface level.

6 Conclusion

We have described the quantification operators that can make the text more concise while preserving the original semantics in the input propositions. Though we would like to incorporate imprecise quantifiers such as `few`, `many`, `some` into our system because they have potential to drastically reduce the text further, these quantifiers do not have the desired property in which the readers can recover the exact entities in the input propositions. The property of

preserving the original semantics is very important since it guarantees that even though the surface expressions are modified, the information is preserved. This property allows the operators to be domain independent and reusable in different natural language generation systems.

We have described an algorithm which systematically derives quantifiers from input propositions, discourse history and ontological information. We identified three types of information from the discourse and ontology to determine if a universal quantifier can be applied. We also minimized the ambiguity between distributive and collective readings by selecting an appropriate universal quantifier. Most importantly, for multiple quantifiers in the same sentence, we have shown how our algorithm generates different quantifed expressions for different scope orderings.

7 Acknowledgement

We would like to thank anonymous reviewers for valuable comments. The research is supported in part by the National Library of Medicine under grant LM06593-02 and the Columbia University Center for Advanced Technology in High Performance Computing and Communications in Healthcare (funded by the New York State Science and Technology Foundation). Any opinions, findings, or recommendations expressed in this paper are those of the authors and do not necessarily reflect the views of the above agencies.

References

Jon Barwise and Robin Cooper. 1981. Generalized quantifiers and natural language. *Linguistics and Philosophy*, 4:159–219.

Alexander Borgida, Ronald Brachman, Deborah McGuinness, and Lori Alperin Resnick. 1989. CLASSIC: A structural data model for objects. In *ACM SIGMOD International Conference on Management of Data.*

Bob Carpenter. 1997. *Type-Logical Semantics.* MIT Press, Cambridge, Massachusetts.

James J. Cimino, Paul D. Clayton, George Hripcsak, and Stephen B. Johnson. 1994. Knowledge-based approaches to the maintenance of a large controlled medical terminology. *The Journal of the American Medical Informatics Association*, 1(1):35–50.

Ann Copestake, Dan Flickinger, Ivan A. Sag, and Carl J. Pollard. 1999. Minimal recursion semantics: An introduction. Manuscript available via http://lingo.stanford.edu/pubs.html.

Norman Creaney. 1996. An algorithm for generating quantifiers. In *Proc. of the 8th International Workshop on Natural Language Generation*, Sussex, UK.

Norman Creaney. 1999. Generating quantified logical forms from raw data. In *Proc. of the ESSLLI-99 Workshop on the Generation of Nominal Expressions.*

M. Dalal, S. Feiner, K. McKeown, D. Jordan, B. Allen, and Y. alSafadi. 1996. MAGIC: An experimental system for generating multimedia briefings about post-bypass patient status. In *Proc. 1996 AMIA Annual Fall Symp*, pages 684–688, Washington, DC, October 26–30.

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. 1992. *Generating Referring Expressions: Constructing Descriptions in a Domain of Objects and Processes.* MIT Press, Cambridge, MA.

Henriette de Swart. 1998. *Introduction to Natural Language Semantics.* CSLI Publications.

Michael Elhadad and Jacques Robin. 1992. Controlling content realization with functional unification grammars. In *Aspects of Automated Natural Language Generation*, Lecture Notes in Artificial Intelligence, 587, pages 89–104. Springer-Verlag, Berlin, April.

Pierre-Joseph Gailly. 1988. Expressing quantifier scope in French generation. In *Proceedings of the 12th International Conference on Computational Linguistics (COLING-88)*, volume 1, pages 182–184, Budapest, August 22-27,.

Barbara J. Grosz, Douglas E. Appelt, Paul A. Martin, and Fernando C. N. Pereira. 1987. TEAM: An experiment in the design of transportable natural-language interfaces. *Artificial Intelligence*, 32(2):173–243, May.

Jerry Hobbs and Stuart Shieber. 1987. An algorithm for generating quantifier scopings. *Computational Linguistics*, 13(1-2):47–63, January-June.

Helmut Horacek. 1997. An algorithm for generating referential descriptions with flexible interfaces. In *Proc. of the 35th ACL and 8th EACL*, pages 206–213.

Ronald M. Kaplan and Joan Bresnan. 1982. Lexical-functional grammar: A formal system for grammatical representation. In Joan Bresnan, editor, *The Mental Representation of Grammatical Relations*, chapter 4. MIT Press.

Martin Kay. 1979. Functional grammar. In *Proceedings of the 5th Annual Meeting of the Berkeley Linguistic Society*, pages 142–158, Berkeley, CA, February 17-19,.

Alistair Knott, Mick O'Donnell, Jon Oberlander, and Chris Mellish. 1997. Defeasible rules in content selection and text structuring. In *Proc. of the 6th European Workshop on Natural Language Generation*, Duisburg, Germany.

James D. McCawley. 1981. *Everything that linguists have always wanted to know about logic (but were ashamed to ask).* University of Chicago Press.

Kathleen McKeown, Shimei Pan, James Shaw, Desmond Jordan, and Barry Allen. 1997. Language generation for multimedia healthcare briefings. In *Proc. of the Fifth ACL Conf. on ANLP*, pages 277–282.

Kathleen R. McKeown. 1985. *Text Generation: Using Discourse Strategies and Focus Constraints to Generate Natural Language Text.* Cambridge University Press, Cambridge.

Chris Mellish, Mick O'Donnell, Jon Oberlander, and Alistair Knott. 1998. An architecture for opportunistic text generation. In *Proc. of the 9th International Workshop on Natural Language Generation.*, pages 28–37.

George Miller, Richard Beckwith, Christiane Fellbaum, Derek Gross, and Katherine Miller. 1990. Five papers on WordNet. CSL Report 43, Cognitive Science Laboratory, Princeton University.

Douglas B. Moran and Fernando C. N. Pereira. 1992. Quantifier scoping. In Hiyan Alshawi, editor, *The Core Language Engine*, pages 149–172. MIT Press, Cambridge, MA.

Jong C. Park. 1995. Quantifier scope and constituency. In *Proc. of the 33rd ACL*, pages 205–212.

Rebecca Passonneau, Karen Kukich, Vasileios Hatzivassiloglou, Larry Lefkowitz, and Hongyan Jing. 1996. Generating summaries of work flow diagrams. In *Proc. of the International Conference on Natural Language Processing and Industrial Applications*, pages 204–210, New Brunswick, Canada. University of Moncton.

Fernando C. N. Pereira. 1990. Categorial semantics and scoping. *Computational Linguistics*, 16(1):1–10.

Owen Rambow and Tanya Korelsky. 1992. Applied text generation. In *Proceedings of the Third ACL Conference on Applied Natural Language Processing*, pages 40–47, Trento, Italy.

Ehud Reiter and Robert Dale. 1992. A fast algorithm for the generation of referring expressions. In *Proceedings of the 14th International Conference on Computational Linguistics (COLING-92)*, pages 232–238, Nantes, France.

Jacques Robin. 1995. *Revision-Based Generation of Natural Language Summaries Providing Historical Background.* Ph.D. thesis, Columbia University.

James Shaw. 1998. Segregatory coordination and ellipsis in text generation. In *Proc. of the 17th COLING and the 36th Annual Meeting of the ACL.*, pages 1220–1226.

Jan van Eijck and Hiyan Alshawi. 1992. Logical forms. In Hiyan Alshawi, editor, *The Core Language Engine*, pages 11–38. MIT Press, Cambridge, MA.

Zeno Vendler. 1967. Each and every, any and all. In *Linguistics in Philosophy*, pages 70–96. Cornell University Press, Ithaca and London.

Leo Wanner and Eduard Hovy. 1996. The HealthDoc sentence planner. In *Proc. of the 8th International Workshop on Natural Language Generation*, pages 1–10, Sussex, UK.

William A. Woods. 1978. Semantics and quantification in natural language question answering. In *Advances in Computers*, volume 17, pages 1–87. Academic Press.

Frans Zwarts. 1983. Determiners: a relational perspective. In A. ter Meulen, editor, *Studies in model-theoretic semantics*, pages 37–62. Dordrecht: Foris.