Lexical Choice for Complex Noun Phrases

Michael Elhadad
Mathematics and Computer Science Dept.
Ben Gurion University in the Negev
Beer Sheva, 84105 Israel

Abstract.

This paper presents a lexical choice component for complex noun phrases. We first explain why lexical choice for NPs deserves special attention within the standard pipeline architecture for a generator. The task of the lexical chooser for NPs is more complex than for clauses because the syntax of NPs is less understood than for clauses, and therefore, syntactic realization components, while they accept a predicate-argument structure as input for clauses, require a purely syntactic tree as input for NPs. The task of mapping conceptual relations to different syntactic modifiers is therefore left to the lexical chooser for NPs.

The paper focuses on the syntagmatic aspect of lexical choice, identifying a process called "NP planning". It focuses on a set of communicative goals that NPs can satisfy and specifies an interface between the different components of the generator and the lexical chooser.

The technique presented for NP planning encapsulates a rich lexical knowledge and allows for the generation of a wide variety of syntactic constructions. It also allows for a large paraphrasing power because it dynamically maps conceptual information to various syntactic slots.

1. Introduction

This paper addresses the issue of lexical choice for complex noun phrases (NPs). Complex NPs are characterized by the presence of various syntactic modifiers - adjectives, nouns, PPs and relative clauses - of conjunctions and of appositions. Mapping the conceptual representation of a discrete set of individuals to such complex NPs can be performed in many different ways. Consider the following example in the context of ADVISOR II, a system providing advice to university students preparing their course schedule. In this domain, numerous NPs can be formed to refer to the set of three topics which are covered in the AI class and which are interesting to the hearer (according to a user-model):

1. *Vision, Expert Systems and NLP...*
2. *Interesting topics, Vision, Expert Systems and NLP...*
3. *Three interesting topics...*
4. *Some interesting AI topics...*

5. *Many/Most AI topics are interesting.*

6. *Few of the topics that interest you are in AI.*

Analyzing this variety of realizations for the same conceptual description, one can observe:

- Different syntactic structures: in (1) a "proper nouns" conjunction, in (2) an apposition of a common noun and a conjunction and in the other cases, a common noun with various pre and post modifiers.

- Different syntactic modifiers realizing the same conceptual relation: the relation between the AI course and the topics it covers can be expressed by a noun-noun modification (*AI topics*), a PP post-modifier (*topics in AI*) or a relative clause (*topics that are covered in AI*).

- Some of the conceptual information conveyed outside the NP: While in (4) the NP conveys both properties that the topics are covered in AI and interesting, in (5) and (6) only one property is in the NP, the other one being conveyed in the rest of the clause.

- Different realizations of the set cardinality: implicitly, through an enumeration in (1) and (2), as a precise number (3), an indefinite (4), and a judgment determiner in (5) and (6).

The combination of all these choices results in many possible paraphrases. Within a standard pipeline architecture for the generator, one must first explain why the lexical chooser is the component responsible for most of the paraphrasing power observed in these examples.

In the standard pipeline architecture assumed here, the lexical chooser sits between a content planner and a syntactic realization component. The content planner passes conceptual information to the lexical chooser, which is responsible for "phrase planning" and the selection of open-class lexical items. The output of the lexical chooser is then passed to the syntactic realization component which is responsible for mapping thematic structures (agent, theme...) to syntactic roles (subject, object...); providing linear precedence constraints among constituents; deal with agreement and morphological inflections; and control the application of syntactic alternations (cleft, dislocation, causative, passive, dative move, middle construction etc).[1] This standard architecture is summarized in Fig.1.

[1] cf. [36] for a discussion of the overall architecture of a generator, [16] for a general discussion of the role of a lexical chooser and [17] for a discussion of the role of a syntactic realizer.

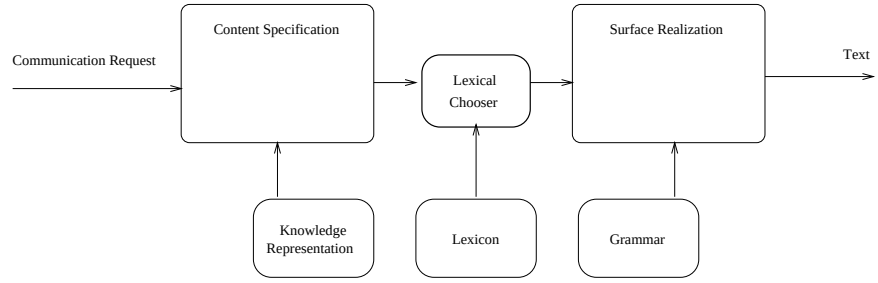

Figure 1. Standard architecture - placement of lexical choice within a generator

Within this architecture, there are several reasons why lexical choice for NPs as a component within a complete generation system deserves increased attention:

- There is an asymmetry between clauses and NPs: while there is a clear consensus on the form of input a syntactic realization component can expect for clauses (a predicate argument structure also known as thematic or transitivity structure) the situation is less clear for NPs. NPs are nevertheless as complex and versatile a syntactic category as clauses. Except for limited cases (cf. [41], [28], [18]), no linguistic semantic classification of nominals has been provided. As a consequence, while for clauses input can be provided in thematic form, for nominals it must be provided directly in terms of syntactic roles (head, pre-modifier or post-modifier). The task of mapping domain-specific thematic relations to the syntactic slots in an NP is therefore left to the lexical choice module.

- An analysis of template-based generation systems shows that many of the problems faced by a template generation system are due to the handling of NPs (deciding on pronominalization and binding, avoiding repetitions, combining several predicates on the same entity are all difficult to achieve using template techniques). To address these limitations but still keep the productivity and practicality of templates, hybrid architectures appear that handle paragraphs and clauses as templates to be filled with NPs that are built using knowledge-based techniques (cf. for example [7] and FLOWDOC [33] for examples of hybrid generators, and [37] for a discussion of the respective merits of NLG and templates). In such systems, a separate NP generator is built. A similar architecture is used in the KNIGHT explanation generation system [27], where a separate NP

generator is used to insert well-formed NP descriptions into clause patterns.

- There is a close inter-relation between the capability of the syntactic realization component and the task the lexical chooser must fulfill.[2] Because the syntactic realizer does less work for NPs than for clauses, the lexical chooser must compensate. In general, the lexical chooser is capable of mapping conceptual attributes of the input to syntactic modifiers of NPs because it can rely on domain-specific knowledge. For example, in [10], in the domain of terrorist news stories, the attribute `bomb-type` is always mapped to an adjective (*a remote-controlled bomb*) and `bomb-location` to relatives (*a bomb which was hidden in the sewers*).

Relying only on domain knowledge to map conceptual attributes to syntactic modifiers has three limitations:

1. It is not portable from domain to domain.
2. It limits compositionality. When several modifiers must be attached to a single NP, there are difficult syntactic co-occurrence restrictions that only the syntactic realizer should handle.
3. It limits the paraphrasing power of the system, as in general the mapping attribute/syntactic slot is specified one-to-one.[3]

Key to the development of a partially portable lexical chooser is the problem of determining general constraints on the mapping attributes/syntactic slots and on the combination of such mappings when several modifiers are used.

This paper addresses the issue of building a lexical chooser for complex noun phrases (LCNP). It presents a system that can generate alternative realizations like (1)-(6) and control their selection through a small set of pragmatic factors. In this task, we focus on two primary aspects:

- We identify a stage in the generation of complex NPs called NP planning which consists in mapping semantic modifiers onto syntactic slots of an appropriate type and controlling their co-occurrence.
- We focus on the generation of NPs that can participate to the expression of the argumentative intent of the speaker and show how

[2] This is an instance of what Meteer has called the expressibility problem [32].

[3] A more flexible mapping is in principle possible but it is difficult to abstract away from the syntactic constraints when negotiating to which syntactic slot a given conceptual attribute is to be mapped.

this pragmatic goal can affect the structure of the generated NP (cf. [15] for a discussion of the use of argumentation in generation in general.)

The LCNP therefore acts as a black-box that receives as input a conceptual description for sets and individuals and is controlled by discursive and evaluative parameters to produce a syntactic tree for a complete NP. The syntactic tree is then passed to the SURGE syntactic realization component to produce text (cf. [17] for an overview of SURGE).

To a large extent the techniques presented here are portable. The described LCNP has been developed within the context of the ADVISOR II system [14]. It is currently being re-used in the context of a hybrid generator for automatic business letter composition. A similar strategy has been used in the development of the PLANDOC system [26] which generates documentation for telephone network extensions.

In the following sections, we first present previous work on NP generation and precise the scope of our work (which types of NPs are handled). We then present the input accepted by the LCNP, the output it produces and passes down to the syntactic realization component, and the procedure used to map from input to output.

2. Previous Work on NP Generation

2.1. Communicative Roles of NPs Covered

While NPs can be used to satisfy many different communicative goals, we focus in this paper on lexical choice for NPs satisfying the following criteria:

- The NPs denote sets of countable entities.
- The NPs are used to fulfill the communicative goals of referring and evaluating (expressing an argumentative intent).
- We only consider restrictive modifiers.

2.1.1. *Referring and Describing*

The distinction between descriptive and referential NPs is discussed in [11], [24] and [40]. In a referential usage, an NP is used when the speaker wants the hearer to identify some object. In this case, modifiers are used to contrast the target object from other potential referents. The proper modifiers are chosen based on their discriminatory power. For example, in a background containing blocks of different forms and colors, the

generator will pick a combination of form and color that can be used to uniquely identify the referent and differentiate it from all other blocks in the background. Several algorithms for modifier selection are presented in [8] pp.249-262, [2], [35] [38] and [9]. In particular, these algorithms ensure that selected modifiers do not trigger unwanted implicatures.

In attributive usages, the goal of the speaker is to inform the hearer of some property of an object. In [31] and [2] for example, modifiers are introduced to perform `inform` speech-acts. In this case, the generator must map from the information in the knowledge-base describing the object to a modifier denoting the property being attributed. Note that in KAMP [2], the notion of *action subsumption* was introduced to account for cases where a particular noun-phrase simultaneously served as a referring and attributive expression and the modifiers are selected both because of their contrastive value and of their informative value.

Related to the distinction attributive/referential, the distinction between *initial* and *subsequent* reference within a discourse is important in generation (cf [30] p.216, [10] pp.176-179, [8] pp.190-192 for a discussion of the differences).

Also related to the distinction attributive/referential, is the distinction between *restrictive* and *non-restrictive* modifiers. The linguistic test to identify non-restrictive modifiers is that they can be paraphrased by a conjunction of two independent clauses ([41] Chap.1):

1. **Non-restrictive***: The chair, which is red, is new.*
2. **Restrictive***: The red chair is new.*
3. *The chair is new. The chair is red.*

(1) is semantically equivalent to (3), (2) is not. Restrictive modifiers contribute to the goal of referring, non-restrictive modifiers to the goal of describing or informing.

In this paper, we only address the case of referring NPs and restrictive modifiers.

2.1.2. *Evaluating*

There are few examples of the generation of usages of NPs that are neither attributive nor referential, but evaluative. In PAULINE [21], the generator could produce a sentence like *poor John was severely beaten by the police* where *poor* does not denote any information about John but rather expresses the orientation of the speaker. In this work, Hovy covered many different linguistic devices satisfying pragmatic constraints and as a result provides only a very superficial treatment of modifiers selection (he devotes a single paragraph to its discussion). In [5] and [6], a model for describing the argumentative potential of lexical

items is introduced which is derived from the theory of Ducrot and Anscombre [1]. This model aims at explaining how for example adjectives like *courageous* express both a property of the modified object and an argumentative orientation of the speaker (a favorable evaluation of the object), whereas adjectives like *intrepid* or *bold* while conveying roughly the same information also convey a different orientation. The reported work is oriented towards interpretation. We use here many concepts derived from this work in the context of generation. We also discuss in this paper the use of judgment determiners to evaluate the quantity or number of elements in a set to satisfy an argumentative intention of the speaker.

2.2. Stages in the Generation of NPs

In general, in the conceptual input to a generator objects are represented as frames and a subset of the object slots are used to derive a lexical entry which serves as the head-noun of the NP. The other slots are mapped to NP modifiers. When dealing with sets, the determiner sequence must be generated to specify the quantity or number of elements in the sets.

NP generation therefore consists of three subtasks:

1. *head determination*
2. *modifier generation*
3. *determiner generation*

2.2.1. *Head determination*

Head determination is generally performed using a discrimination net associating lexical entries (nouns) to different levels of the is-a hierarchy of the underlying knowledge-base.

Following [39], Reiter used the notion of *basic-level* to define a lexical preference relation and guide head determination [35], by using predefined nodes in the discrimination net as preferred head-nouns. Reiter also used a user model specifying which nodes are known to the user. If a preferred node is unknown, an alternative description is dynamically chosen (e.g., *a shark* vs. *a dangerous fish*)

In [10], the subset of the slots which can produce the head of NPs is statically identified for a given domain. For example, in Danlos's domain, the slots `name` and `profession` of a person can be used to generate NPs such as *Bill Clinton* or *the President of the USA* ([10], pp.175-182).

In some cases, a complex entity is in fact encoded in the knowledge representation language as a primitive entity and the whole NP is used as a "macro proper noun". For example, in all existing systems, noun compounds are represented as single lexicon entries, chosen all at once (*e.g.*, in ANA [25] *Wall Street securities markets*). In contrast, we present a technique based on Levi's Recoverably Deletable Predicates (RDP) [28] to compositionally determines when certain noun compounds can be used. Besides this contribution, the presented LCNP combines several existing techniques for head determination.

2.2.2. *Modifier Generation*

Modifier generation includes two tasks: determination of which modifiers should be included in the NP and determination of the syntactic type of the modifiers. We have discussed above how modifiers are selected to satisfy the goals of either referring or describing the denoted entity.

For mapping to syntactic modifiers, all existing systems rely on a static mapping from conceptual modifier type to syntactic modifier. For example, as mentioned in the introduction, in [10], `bomb-type` is always mapped to an adjective (*a remote-controlled bomb*), `bomb-location` to relative clauses (*a bomb which was hidden in the sewers*), `bomb-charge` to additional sentences (*Anarchists exploded a bomb. The bomb contained 4kg of dynamite*).

The syntactic structure of the generated NPs is thus hard-wired in the conceptual input. In contrast, the described LCNP dynamically determines the syntactic type of each modifiers based on its *value* plus pragmatic factors. Also, the variety of modifiers generated by previous system was quite limited (*e.g.*, only relative modifier in Danlos's system, only adjective premodifiers and relatives in EPICURE). The LCNP presented here produces a greater variety of NP modifiers.

2.2.3. *Determiner Generation*

Most of the previous work in generation has focused on the difficult decision definite/indefinite; relatively little work has dealt with non-singular entities. Notable exceptions are [19] and [4]. Gailly [19] investigates how ambiguous scopings can be avoided by signalling scope relations using markers like "each". In the WISBER system described in [4], the selection of determiners is constrained by an analysis of the scope of the NP. For example, in *"three persons bought a car*, a singular reference to *car* can be selected because the cars are within the scope of the plural subject. (A disambiguating "each" could be added, but would lead to a less natural sentence.) If the same sentence is followed

by *I sold the three cars*, because the cars are not within the wider scope of a plural subject, a plural determiner must be used.

In this paper, we present techniques for the generation of judgment determiners which fulfill an important pragmatic function (evaluation of the number of elements in a countable set according to an argumentative intent).

3. Input to the LCNP

In our architecture, the lexical chooser is positioned between the content planner and the syntactic realization module of a generator. The input is, therefore, a conceptual description and the output a well-formed syntactic description with all open class lexical items selected, and realization features for closed-class items (determiner sequence) properly set. We first describe and motivate the kind of input defined as an interface between the content planner and the lexical chooser for the specification of NPs. In the next section, we present the interface between the lexical chooser and the syntactic realization module.

3.1. Conceptual Descriptions

Following ([3], Sect.1.3), NPs *as a whole* are viewed as the expression of *generalized quantifiers*. The input to the LCNP is accordingly a set specification. Sets are characterized in intension by a domain and the properties that must be satisfied by all elements.

The description is given in FUF, our extension of the functional unification formalism of Kay [22] used for the implementation. The input specification contains objects of three types:

- **Individuals** correspond to instances in the reference model:

```
((name <name-of-instance>)
 (cat <name-of-concept>)
 (index <unique-id>))
```

  Concept names are organized into a FUF feature hierarchy [13] that corresponds to the conceptual is-a hierarchy.

- **Relations** correspond to attributes in a frame representation.

```
((cat relation)
 (name <name-of-attribute>)
 (1 <argument1>)
 (2 <argument2>))
```

The arguments of a relation can be sets, in which case the semantics is distributive. (We do not deal with the possibilities of collective or mixed readings.) Relations are also organized as a hierarchy.

– **Sets** of instances, described by the following features (all are optional except `cat` and `index`):

```
((cat set)
 (index <unique-id>)
 (kind <prototype>)
 (cardinality <n>)
 (extension <list-of-individuals>)
 (intension <a-relation>)
 (reference <a-set>))
```

`kind` is used for sets of objects of the same type. `extension` is the explicit list of the set elements. `intension` and `reference` are described below.

3.1.1. *Why two Conceptual Modifiers are Used*

Usually, a set can be specified mathematically by a domain and a restriction predicate, as in the following, describing the set of topics that interest the student and that are in the area of AI:

$S = \{x \in \text{TOPICS} \mid \textit{Interest(x,student)} \wedge \textit{Area(x,AI)}\}$

In our input, however, we distinguish between two types of predicates to restrict the domain: *reference* and *intension*. This distinction is necessary to account for the semantic behavior of evaluative NPs, which are the focus of this work.

The key motivation is that evaluative NPs do not satisfy a semantic condition called the *intersection condition* in [3]. The linguistic test corresponding to the formal definition of the *intersection condition* is the following: let P_1 and P_2 be two properties; then if a determiner D satisfies the intersection condition, the sentences *there are* D P_1 P_2 N and D P_1 N *are* P_2 are semantically equivalent. For example:

There are exactly 3 interesting AI topics.
Exactly 3 interesting topics are in AI.
Exactly 3 AI topics are interesting.

These three forms are equivalent, indicating that the quantifier *exactly n* satisfies the intersection condition. In contrast, consider:

(1) There are many interesting topics which are in AI.
(2) There are many AI topics which are interesting.

These NPs are not equivalent, as shown, for example, by considering the following situation: a person has interest in 100 topics; AI covers 10 topics; the intersection between the interesting topics and the AI topics contains 7 elements. Then (1) is probably not valid (7 topics out

of 100 is not many) while (2) is valid (7 out of 10 is many). Note that the "classical" quantifiers, corresponding to the mathematical $\exists$ and $\forall$, both satisfy the intersection condition, but evaluative quantifiers, *e.g.*, *many, few, most*, do not satisfy it.

Consider now the fact that in both (1) and (2) the NPs with the *many* determiner denote the same set of individuals (the 7 topics of the intersection). The validity of the sentences, however, is different when the scope of the *many* changes from one modifier to the other. This indicates that the input conceptual description of sets must attribute a different status to the two modifiers. These two modifiers correspond to different perspectives that can be held on the set: when the *interest* property is the intension and the *area* property is the reference, the definition can be written as follows:

S1 = $\{x \in$ AI-TOPICS $\mid$ *Interest(x,student)*$\}$

And, under normal circumstances this representation leads to the English realization: *Most AI topics are interesting.*

If in contrast the perspective is switched, and *interest* becomes the reference and *area* the intension, then the definition and realization become:

S2 = $\{x \in$ INTERESTING-TOPICS $\mid$ *Area(x,AI)*$\}$

Few of the topics that interest you are in AI.

The same conceptual information of a set of topics satisfying two properties can lead to the generation of two contradictory argumentative evaluations. This indicates that, because evaluative quantifiers do not satisfy the intersection condition, the structuring of properties in a set specification between *reference* and *intension* must be present in the input to the generator (as in S1 and S2), and that a neutral representation for sets such as S would not be appropriate.

A FUF input for the following set is shown below:

$S1 = \{x \in$ TOPICS $\mid Interest(x, student) \wedge Area(x, AI)\}$

```
((cat set)
 (kind ((cat topic)))
 (cardinality 3)
 (extension ~( ((semr ((cat topic) (name nlp))))
               ((semr ((cat topic) (name vision))))
               ((semr ((cat topic) (name robotics)))) ))
 (reference ((cat set)
             (kind ((cat topic)))
             (cardinality 10)
             (intension ((cat class-relation)
                         (name area)
                         (1 {^ argument})
                         (2 ((cat field) (name AI))))))))
 (intension ((cat user-relation)
             (name interest)
```

```
(1 {^ argument})
(2 ((cat student))))))
```

Intuitively, this set contains the 3 topics that are of interest to the user among the 10 topics which are covered in AI.

3.2. Pragmatic Annotations

In addition to the conceptual input, the interface to the discourse model and evaluative model is encoded as follows:

- **Discourse model**: specifies the features `definite` (yes or no); selects the appropriate modifiers to be included in the input from all the properties present in the conceptual database, using a modifier selection algorithm as presented in Section **??**; and selects some of the realization switches identified below (`realize-intension`, `realize-extension`, `realize-reference`).
- **Evaluative model**: specifies the argumentative intent of the speaker as the evaluation of the entity along a certain scale and with a given orientation. This is encoded using the following features: `orientation` (+ or -); `scale` which can be `cardinality` or one of the scales identified in the domain; if possible, the `degree` feature is also specified (+ or -) to indicate the strength of the evaluation. It also sets the value of the realization switch `realize-quantity`. The following example encodes the argumentative evaluation that a set of topics has many elements:

```
((topics
  ((cat set) (kind ((cat topic)))
   (cardinality 7)
   (reference
    ((cat set) (kind ((cat topic)))
     (cardinality 10)
     (intension
      ((cat class-relation) (name area)
       (1 {^ argument})
       (2 ((cat field) (name AI)))))))
   (intension
    ((cat user-relation) (name interest)
     (1 {^ argument})
     (2 ((cat student)))))))
 (argumentation
   ((cat evaluation) (evaluated {topics})
    (scale ((name cardinality)))
           (orientation +))))
```

3.2.1. *Why the Evaluative Model is Required*

Evaluative quantifiers (like "many topics") are *non extensional*: they do not have a definite truth value (cf. for example a discussion in [23]

pp.257–8). As a consequence, the LCNP must rely on the speaker's argumentative intention to be able to generate such quantifiers.

Consider for example the fact that in both (1) and (2) the NPs with the *many* determiner denote the same set of individuals (the 7 topics of the intersection). The validity of the sentences, however, depends depending on the argumentative intention of the speaker. This indicates that a conceptual input only, encoding only the denotation of a set cannot be sufficient to account for the generation of vague quantifiers, and justifies the addition of pragmatic annotations within the input to the LCNP.

The particular model we have developed for the evaluative model is based on the theory of Argumentation Within Language (AWL) of Anscombre and Ducrot [1] and is described in detail in [15].

4. Output of the LCNP

The output of the LCNP is an input to the syntactic realization component SURGE. In this section, we describe the main features of the NP descriptions expected by SURGE.

4.1. Syntactic Structure of NPs

Following [34] and [20], the syntactic description of NPs contains four types of modifiers in addition to the determiner sequence:

- Determiner Sequence: *many of the same topics.*
- Describer: *an interesting topic.*
- Classifier: *a programming class.*
- PP Qualifier: *a class in AI.*
- Clause Qualifier: *a class which AI covers.*

Describers and classifiers are pre-modifiers and qualifiers are post-modifiers. The distinction between describers and classifiers is that, for a describer, the construct *(the D N)* is semantically equivalent to *(the N is D)*:
The red chair vs. *The chair is red.*
This does not hold for classifiers:
The programming assignment vs. * *The assignment is programming.*

In general, describers are restrictive modifiers and therefore participate to the satisfaction of the goal of referring. Classifiers are used to create names for entities.

Qualifiers can be either restrictive or non-restrictive. We restrict our attention to restrictive ones.

pre-det	(of)	det	deictic2	ord	card	quant	NP-head
all	*of*	*the*	*famous*	*first*	*ten*		*boxes*
half	*of*	*my*				*many*	*properties*
twice as						*much*	*work*

Figure 2. Syntactic structure of the determiner sequence

In addition, complex NPs can include conjunctions and appositions. Conjunctions are used to enumerate elements of a set as in: *AI, Expert Systems and NLP.* Appositions are the juxtaposition of two full NPs - they provide a supplementary way to refer to the denotation, as in: *2 topics - AI and NLP.* We restrict our attention to only the simplest forms of conjunctions and apposition.

4.2. Syntactic Structure of the Determiner Sequence

The determiner sequence is in itself a complex syntactic constituent. It has the specificity that it is mainly a *closed system - i.e.*, the lexical elements are part of a small set of words which are determined completely by a small set of syntactic features. We describe here the syntax of the determiner sequence as we implemented it in the SURGE realization component.

When implementing SURGE, the issue was to identify a minimal set of features accounting for the variety of determiner sequences observable in English. The syntactic description implemented in SURGE is an augmented version of that presented in ([20], pp.159-176), with additions derived from observations in ([34], pp.136-165). A set of 24 features controlling the realization of the determiner sequence was thus identified, which is presented in detail in ([14], Sect.5.4). We only present here a brief overview of the grammar for determiners, and focus on the features relevant to the satisfaction of the evaluative communicative goal.

The structure of the determiner sequence is shown in Fig.2. Predeterminers can be one of the following elements: *all, both* or *half*, multipliers (*twice, three times*) or fractions (*one fourth*). Complex cooccurrence restrictions exist between the different pre-determiners and different classes of nouns (mass, count nouns denoting a number or an amount) and between pre-determiners, cardinals and quantifiers. There are also special cases of noun classes that take zero articles, including seasons, institutions, transport means, illnesses (Quirk et al

[34], pp.156-159). The implementation of such co-occurrence restrictions explains the complexity of SURGE's determiner grammar.

To control the selection of the various elements of the determiner sequence, we rely on Halliday's distinction between three functions of the determiner sequence:

1. **Deictic**: to identify whether a subset of the thing is denoted, and if yes, which subset. The relevant decisions are depicted below, in Fig.3, in the form of a systemic network. The toplevel distinction is between specific and non-specific determination. A specific deictic denotes a known, well identified subset of the thing. A non-specific deictic denotes a subset identified by quantity. The deictic function is mapped onto the `det` constituent of the determiner sequence.
2. **Deictic2**: to specify the subset of the thing by "referring to its fame, familiarity or its status in the text" ([20], p.162). The deictic2 element is an adjective such as *same*, *usual*, *given*. Such adjectives are part of the determiner sequence because they systematically occur before the cardinal element of the determiner (in the `deictic2` position), in contrast to any other describing adjective, which must occur after the cardinal.
3. **Numerative**: to specify the quantity or the place of the thing. The numerative specification can be either quantitative (expressing a quantity, *three*) or ordinative (expressing a relative position, *third*). In both cases, the expression can be either exact (*one, two...*, *first, second...*) or inexact (*a lot, the next*). The source of the inexactness can be an approximation device (*about three, roughly third, approximately ten*) or a range expression (*between six and ten*). Alternatively, it can be a context dependent expression like *the next, many, few, more*, and an evaluative expression like *enough, too many, too much*. Figure 4 summarizes the relevant decisions. The numerative function is mapped onto the slots `pre-det`, `ord`, `card`, and `quant` of the determiner sequence.

The features that control the syntactic realization of the communicative goal of evaluating a quantity are located in the non-specific region of the deictic network and in the inexact region of the numerative network. The subset of SURGE features which trigger the selection of argumentative determiners is `total, orientation` and `degree`.

5. NP Planning

After having specified the input and the output of the lexical chooser for NPs, we now proceed to the description of the mapping process.

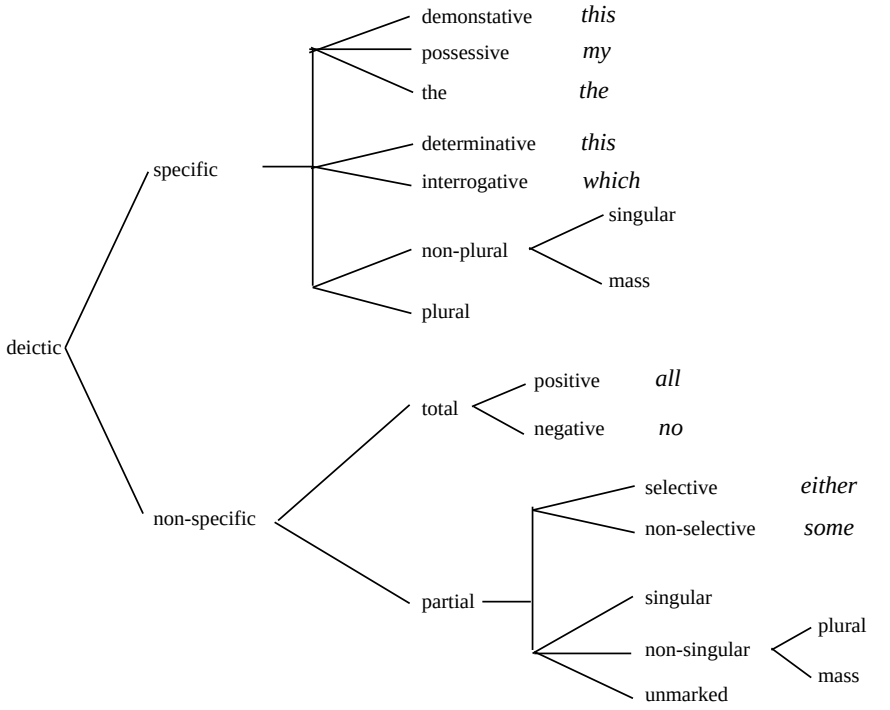

Figure 3. The deictic network

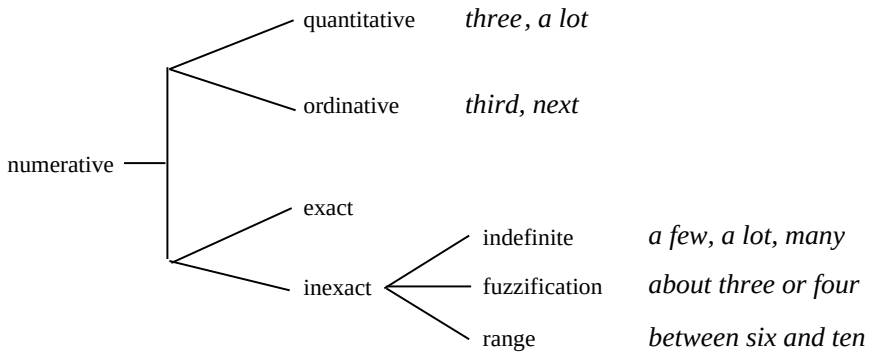

Figure 4. The numerative network

We distinguish between three subtasks to prepare a complete syntactic tree for an NP given an input conceptual structure:

- First, the high level structure of the NP is determined. This includes selecting a head for the NP, determining which conceptual relations will serve as modifiers, and deciding whether an apposition will be used.
- Second, each conceptual modifier is mapped to one of the four types of syntactic modifiers identified above.
- Last, the features controlling the generation of the determiner sequence are computed from the input features.

5.1. Two Dimensions of Lexical Choice

An important aspect of the method presented here is that the lexical chooser is in charge of constructing the high-level syntactic structure of the NPs. In general, we discuss in [16] the two tasks a lexical chooser must fulfill:

- *Syntagmatic decisions*: choosing among the many possible mappings from the flat conceptual network the lexical chooser receives as input onto the thematic tree structure it must produce as output.
- *Paradigmatic decisions*: choosing among possible alternative possibilities inside a particular thematic structure, *e.g.,* the choice of the noun "fish" as opposed to "shark".[4]

While syntagmatic decisions may seem to be more syntactic in nature (and therefore belong to the syntactic realization component), they are directly intertwined with various lexical choices and in the case of NPs, they cannot be performed by a domain-independent syntactic realizer.

The examples provided in the introduction illustrate how a syntagmatic decision (using a list in extension of the elements of a set as in *"Vision, Expert Systems and NLP"* vs. using a description in intension of the elements as in *"three interesting topics"*) completely modifies which lexical items are selected.

The general process used by the LCNP is therefore to first map the input conceptual structure to a syntactic structure, and only then to

[4] We refer here to the notion of paradigm accepted in modern linguistic and derived from the notion of "groupes associatifs" used by Saussure in the *Cours de Linguistique Générale*. Ducrot gives the following definition for paradigm in the "syntagm and paradigm" article of [12]: "two unit u and u' belong to the same paradigm if and only if they can be exchanged within the same syntagm, that is, if there exist two syntagms vuw and $vu'w$." This definition is at the basis of the classical picture of the syntagmatic axis as an horizontal axis and the paradigmatic axis as the vertical line representing all the potential elements that can appear within a given linguistic environment.

select the specific lexical items for each subconstituent of the NP (head, determiner, pre and post modifiers).

Because in the case of NPs the syntactic structure is not specified in thematic terms (as is the case for clauses), the LC must also insure that the combination of syntactic modifiers selected during the syntagmatic stage can indeed be realized by the grammar. In this case, the LC must compensate for the lack of syntactic generalizations in the syntactic realization component. We call this aspect of the lexical choice process structure planning.

5.2. Structure Planning

The representation of sets contains up to five major features: extension, intension, kind, reference and cardinality. Each of these features, when it is present, can be realized in different ways. For example, the extension can be realized by a conjunction listing all the elements. The cardinality can be expressed by an exact value, as in *six assignments*, by an evaluation, as in *many assignments*, or not at all, as in *assignments*.

When several of these features are realized together, the structure of the NP becomes complex. For example, if both extension and intension need to be realized together, an apposition must be selected, as in *interesting topics, Vision and NLP.* If the cardinality must be expressed, then a head noun must be introduced, to carry the determiner, as in *two things, Mathematics and essays.*

To enumerate the different NP patterns which can be generated from this description, four realization switches are defined, which control how each aspect of the set representation must be realized. The switches with their possible values are:

- **realize-extension**: yes/no
- **realize-intension**: yes/no/clause
- **realize-reference**: yes/no/clause
- **realize-quantity**: evaluation/cardinal/no

The value of these features determine which element of the set description must be realized in the NP, and how. A value of `no` indicates that the element should not be realized, while `yes` indicates that it should be, in any possible way. The following values have a special meaning:

- Both intension and reference can be realized by the position of the NP in a matrix clause. For example, consider the set of topics being covered in AI. The property *covered in AI* is the intension of the set description. If this set is realized in the context of the clause *AI*

covers many topics, then the intension is realized by the fact that the NP appears in object position of the clause, not by any of the elements of the NP itself. In this case, the realization is of mode `clause`.

- Quantity can be realized in two modes: `cardinal`, *i.e.*, by stating the precise number of elements in the set, or `evaluation`, *i.e.*, by expressing a judgment on the number of elements.

These four features can be combined into $2 \times 3 \times 3 \times 3 = 54$ ways. Of these 54 ways, however, certain combinations are invalid, because they do not express any information at all about the set, or because they require the co-occurrence of two incompatible features, leaving 27 valid combinations. To illustrate this enumeration, consider the set of topics defined by:

- Kind: topics
- Extension: *Vision*, *NLP* and *Robotics*.
- Intension: the topics are interesting.
- Reference: the set of topics covered in AI (cardinality 10).
- Cardinality: 3

Examples of realizations for this set are shown below:

```
Intension:yes
Extension:yes
Reference:yes
Quantity:evaluation
```

John wants to learn about many interesting topics covered in AI - Vision, NLP and Robotics.

```
Intension:yes
Extension:no
Reference:clause
Quantity:no
```

Some interesting topics are covered in AI.

```
Intension:clause
Extension:no
Reference:yes
Quantity:cardinal
```

John is interested in 3 AI topics.

Certain combinations of features are impossible because they require the NP to be a participant in two clauses simultaneously. Others are incomplete, because they do not express enough information to properly identify the set. For example, the combination `(Intension:yes Extension:No Reference:No)` yields *many interesting topics* but does not specify that the topics are those covered in AI. A systematic enumeration identifies 27 valid combinations of these features, corresponding to 27 different NP patterns.

When all elements are realized, the overall pattern for the realization of the set description is `(head mod1 mod2) conj`, that is, up to two modifiers attached to a head noun realizing the kind, in apposition to a conjunction of nouns realizing each individual element of the extension.

5.3. Choice of Syntactic Modifier Type

Once the top level structure of the NP is planned, the system maps each modifier to a syntactic modifier - either classifier, describer or qualifier. The strategy followed is to map in priority a modifier to a classifier if possible, else to a describer, and in last resort to a qualifier. These priorities, however, can be changed under certain pragmatic situations.

We first determine when each type of modifier can be used. In the following discussion, the modifier relation is represented by the expression `x R y`. One of the arguments, `x` or `y` is the head of the NP. We call the second argument the *external argument*, following [42] as used in [29].

5.3.1. *When can a classifier be used?*

Classifiers correspond to pre-modifiers where a transformation *X N* into *N is X* is not semantically valid. Because the meaning of a classifier modification can be so diverse, there is a tendency to view classifier-head groups as frozen collocations in the domain, that are idiomatic and need to be stored as a group in the lexicon. There are, however, as demonstrated in [28], *productive* processes that generate classifier modification. Levi advances that classifiers are all derived by just one of two syntactic processes: the *deletion* or the *nominalization* of the predicate in the underlying sentence (p.50). In this work, we focus on predicate deletion. For the classifiers resulting from such deletion, a small set of *Recoverably Deletable Predicates* (RDPs) can be identified, such that "only its members and no other predicates can be deleted in the formation of complex nominals; the members of this set are: Cause, Have, Make, Be, Use, For, In, About and From" ([28] p.50).

The claim is, therefore, that classifiers can be produced systematically from a conceptual form `x R y` by using the operation of predicate deletion when `R` is an instance of RDP, and, that a closed set of 9 generic predicates (RDPs) can explain the production of a large class of classifier usages.

Classifiers are preferred over qualifiers because they make up for more compact expression (*e.g.*, compare *an AI topic* with *a topic (which is) about AI*). A compositional production of classifiers is also necessary when several modifiers are present. Consider:

- *There are four programming assignments in AI.*
- *Four AI assignments require programming.*

Depending on the structure of the clause, each one of the modifiers (`assignment-type` and `assignment-field`) can be realized as a classifier. Only when classifiers are compositionally generated can these two sentences be produced from the same conceptual description.

In the implementation, certain domain-specific conceptual relations are classified as a specializations of one of the 9 RDPs, using FUF type hierarchies over relation names. For example, because the domain relation `topics` is a specialization of `IN-rdp`, *an Operating Systems topic* is generated instead of *a topic which is covered in Operating Systems*. When a modifier relation is an instance of RDP, the external argument is mapped to the classifier.

There are some problems in applying this technique that deserve further study: first, Levi's predicates are not defined formally; second, there remains to establish clear criteria to determine when a modifier which is an instance of RDP is nonetheless realized by a qualifier. As it stands, Levi's list of RDPs provide a useful guideline to perform a classification of domain relations, and in our experience, the method does not over-generate classifiers.

5.3.2. *When can a describer be generated?*
To determine if a describer can be generated, the following conditions are checked:

- The relation can be lexicalized by an ascriptive clause (of the form *N is A*, *e.g.*, *a topic is theoretical*)
- The relation can be lexicalized into a clause whose main verb can be transformed into an *-ing* or *-ed* pre-modifier, *e.g.*, *an interesting topic*, *a required class*.

The derivation of *-ed* pre-modifiers is studied in depth in [29] by Levin and Rappaport (L&R). *-ed* pre-modifiers are known as "objective modifiers" in traditional grammars, because the head appears to

correspond to the object of the clause: *a well taught class* corresponds to *someone teaches the class well*, where *class* is the "object" of *teach*. The problem is that not all pairs *verb-object* can give rise to a form *verb-ed object* where *verb-ed* is an adjective. For example, * *a helped man* or * *a thanked man* are not valid modifiers. An alternative analysis based on the thematic role of the head in the underlying clause looks, therefore, more appealing: an *-ed* adjective is usable if the head of the NP corresponds to the affected (or theme) role. Levin and Rappaport argue against the thematic condition based on an analysis of dative verbs, for example, *to feed* as in *feed some cereal to the baby* which can produce *unfed baby* but not *unfed cereal*, even though *cereal* fills the theme (or affected) role. Another class of verbs which do not fit with the thematic condition is the *spray/load* family, as illustrated by (from [29] p.635):

- *load the truck; the recently loaded truck.*
- *load the hay; recently loaded hay.*

where both the affected and the location argument can become the head of the NP. For certain verbs, only the location argument can become the head (*e.g.*, *stuff*, *cram*). L&R, therefore, propose instead an account which seems to rely mainly on the syntactic constraint that the direct object can become head with an *-ed* modifier.

The second issue to determine whether both *-ed* and *-ing* modifiers are appropriate is to check whether the external argument can be omitted. For example, from the relation *a course interests the hearer* can the NP *an interesting course* be derived, even though there is no mention of *the hearer* in the NP? The alternative is *a course interesting for you* in the post-modifier position.

To simplify, the implementation requires each verb entry in the lexicon that can be used as a describer to specify it using the features `(ing-adj yes)` or `(ed-adj yes)`. This feature indicates that the *-ing* or *-ed* form of the verb can be used as an adjective and that the external argument does not need to be realized. To check whether the head matches the appropriate argument of the verb in the underlying clause we use the subcategorization frame of the verb. For example, the lexical entry for the verb *to interest* is shown below:[5]

[5] Using the subcat frame to check the argument compatibility is similar to using the thematic condition criticized by L&R. But as L&R themselves point out, the thematic condition is only challenged by the "marginal data from the exceptional dative verbs and from the relatively small class of *spray/load* verbs" [29] p.657. So the simplification seems warranted by the generality of its application. Note also that the "worst that can happen" is that a relative clause will get generated when an *-ed* describer could have been used, so the cost of the simplification is really negligible.

```
;; Lexical entry for the verb ``interest''
((name interest)
 (type lexical)
 (lex "interest")
 (ing-adj yes)
 ;; This is the subcategorization frame for the verb
 ;; it links syntactic roles 1 (subject) and 2 (object)
 ;; with the semantic roles of the denoted relation.
 (subcat ((1 {semr roles interesting-topic})
          (2 {semr roles interested})))
 ;; This points to the conceptual arguments of the
 ;; relation which must be an instance of ``interest''
 (semr ((cat relation)
        (name interest))))
```

Finally, a describer can be realized to express an argumentative evaluation when the following conditions are met:

- The evaluated element is coindexed with the head of the NP.
- The scale and orientation of the evaluation can be lexicalized by a scalar adjective.

5.3.3. *When can a PP qualifier be generated?*
The last option to realize an NP modifier is to use a qualifier. Qualifiers can be PPs or relative clauses. The lexical chooser tries to generate a PP over a relative clause when possible because it is more compact. The choice between PP and relative is based on the lexical entry of the conceptual relation in the modifier. A relation is mapped in general to a verb. Some relations can also be mapped to a preposition. For example, the relation *topics* which holds between a course and a topic can be realized by the preposition *about*, or by the verb *cover*. So if the *topics* relation is to be realized as a qualifier, the system chooses the realization *a class about AI* over *a class which covers AI*.

5.4. Computing the Syntactic Features for the Determiner Sequence

The third task of the LCNP is to compute the syntactic features of the determiner sequence from the conceptual and pragmatic features provided in input. The following mapping procedure is used.

When mapping from a conceptual description to the features controlling the determiner selection, the first decision is whether the speaker's argumentative intent is to be realized through the use of an evaluative quantifier or with other linguistic devices (such as connotative verbs, scalar adjectives or connectives). This decision can interact with

most lexical choice decisions and is discussed at length in [16] together with the computational implications this has.

When argumentation *is* to be expressed in a determiner site, the following mapping rules are applied:

- **Total**: when the set is the object of a positive evaluation, its cardinality is known and equal to that of the reference set, then total is set to +. If the evaluation is negative and the cardinality is known to be 0, total is set to -. In all other cases, total is set to none.
- **Orientation**: when the set is the object of an argumentative evaluation, orientation records whether the evaluation is high or low. Otherwise, it is set to none.
- **Superlative**: set to yes when the reference set is given, its cardinality is known, the cardinality of the set is larger than half that of the reference set, and the set is the object of a positive argumentative evaluation.

The general heuristics behind these rules is to use the pragmatically strongest determiner possible to realize the speaker's argumentative intent. For example, if *all AI topics are interesting* can be produced, it will be preferred to *some AI topics*.

For **Degree**, the determination of a value is more difficult. Degree determines the selection among *a few, some, many, a (large, great, incredible...) number* if orientation is +, and among *few, a (small, tiny, ridiculous...) number* if orientation is -. In the implementation, degree is limited to have values +, - or none. A finer account of the degree of determiners is needed, but it creates many problems which are discussed in [14].

5.5. Summary: Mapping Conceptual Modifiers to Syntactic NP Modifiers

In summary, NP planning process determines how conceptual modifiers of an NP are mapped onto syntactic NP modifiers of four different types: classifiers, describers, PP qualifiers and relative qualifiers. These four types are tried in this order, giving preference to the most compact forms when they can be used. Figure 5 summarizes the procedure. The following criteria are used to determine when each type of syntactic modifier can be used when the input modifier is a conceptual relation `xRy`:

- A classifier can be used when the relation `R` is an instance of one the nine recoverably deletable predicates identified in [28].
- A describer can be used:

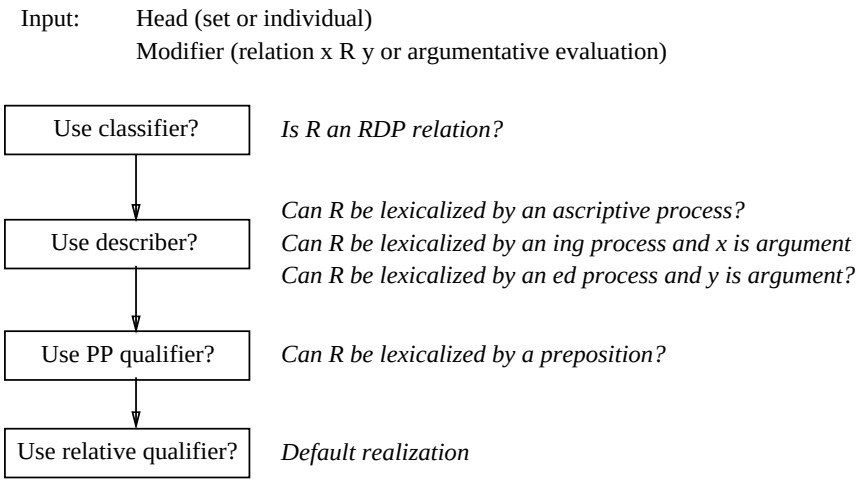

Figure 5. Flowchart of the LCNP procedure

  - When the relation `R` can be lexicalized by an ascriptive relation.
  - When `R` can be realized by a process which carries either the `ing-adj` or `ed-adj` feature, the head argument corresponds to the appropriate thematic role of the relation and the external argument does not need to be expressed.
  - When an argumentative evaluation must be realized, the head of the NP realizes the evaluated element of the evaluation and the scale and orientation of the evaluation can be realized by a scalar adjective.

- A PP-qualifier can be used when `R` can be lexicalized by a preposition.

- In all other cases, a relative-qualifier is generated.

In addition, the argumentative evaluations specified in the input can be mapped to describers or to features of the determiner sequence.

This mapping procedure relies on the fact that in the conceptual representation, all modifiers are represented uniformly as relations - as opposed to a distinction between attributes and relations in the conceptual encoding. This illustrates the general principle that the less the conceptual representation is committed to a linguistic perspective, the more paraphrasing power the lexical chooser can implement.

6. Conclusion

This paper has presented a lexical chooser for complex NPs and highlights the richness of a stage of processing called here *NP planning*. In the lexical processing of NPs, three stages are identified: structure planning, modifier mapping and determiner sequence specification.

The output of the first stage is a generic linguistic structure of the form `(apposition (head mod1 mod2) list)`. The ability to map a conceptual input onto this level of NP structure is a major source of paraphrasing power. For a fully specified set, 27 valid patterns to realize the set structure have been identified. In addition, each one of the two modifiers can be realized in up to four ways. This level of variety ($4 \times 4 \times 27$) illustrates the power of a compositional approach to text generation as opposed to phrasal lexica or templates. Above all, the selection between the many realizations of a set as an NP is sensitive to the pragmatic situation, and therefore produces more appropriate text.

The method described relies on previous work for head determination and modifier selection, and significantly extends the flexibility of lexical choosers by:

- representing all entity modifiers in a uniform manner, as binary relations, therefore relieving the content planner from the task of determining which modifiers must be represented how.
- allowing for a dynamic mapping from modifier to syntactic types.
- describing a compositional technique to produce classifiers.
- considering a large variety of syntactic modifiers (including *ed* and *ing* verbal adjectives).

The method is implemented and has been ported to several domains (the domain of ADVISOR II, of business letters and of telephone maintenance). Future work will focus on the consideration of nominalizations to generate other forms of complex NPs (*e.g.*, *the destruction of the city by the barbarians*).

References

1. J. Anscombre and O. Ducrot. *L'argumentation dans la langue*. Philosophie et langage. Pierre Mardaga, Bruxelles, 1983.
2. D. Appelt. *Planning Natural Language Utterances*. Studies in Natural Language Processing. Cambridge University Press, 1985.
3. J. Barwise and R. Cooper. Generalized quantifiers in english. *Linguistics and Philosophy*, 4:159–219, 1981.

4. R. Block and H. Horacek. Generating referring expressions using multiple knowledge sources. In Karlgren, editor, *Proceedings of COLING 90*, pages Vol 2. 24–29, Helsinki, 1990.
5. D. Bruxelles, S. Carcagno and C. Fournier. Vers une construction automatique des topoi a partir du lexique. *CC AI - Journal for the integrated study of Artificial Intelligence cognitive science and applied epistemology*, 6(4):309–328, 1989.
6. S. Bruxelles and P. Raccah. *Argumentation et Semantique: le parti-pris du lexique*. 1991.
7. J. Coch and R. David. Une application de generation de textes. In *Actes de TALN-94*, pages 37–45, Marseille, France, 1994.
8. R. Dale. *Generating referring expressions in a domain of objects and processes*. PhD thesis, University of Edinburgh, Scotland, 1988.
9. R. Dale and E. Reiter. Computational interpretations of the gricean maxims in the generation of referring expressions. *Cognitive Science*, pages 233–263, 1995.
10. L. Danlos. *The linguistic basis of text generation*. Studies in Natural Language Processing. Cambridge University Press, 1986.
11. K. Donnellan. Reference and definite description. *Philosophical Review*, 75:281–304, 1966.
12. O. Ducrot and T. Todorov. *Encyclopedic Dictionary of the Sciences of Language*. The John Hopkins Universty Press, 1979. Translated from French by Catherine Porter.
13. M. Elhadad. Types in functional unification grammars. In *Proceedings of the 28th Annual Meeting of the Association for Computational Linguistics (ACL'90)*, Pittsburgh, PA, 1990. ACL.
14. M. Elhadad. *Using argumentation to control lexical choice: a unification-based implementation*. PhD thesis, Computer Science Department, Columbia University, 1992.
15. M. Elhadad. Using argumentation in text generation. *Journal of Pragmatics*, 24:189–220, 1995.
16. M. Elhadad, K. McKeown, and J. Robin. Floatings constraints in lexical choice. *Computational Linguistics*, 1996. To appear.
17. M. Elhadad and J. Robin. An overview of surge: a reusable comprehensive syntactic realization component. Technical Report 96-03, Ben Gurion University, Dept of Computer Science, Beer Sheva, Israel, April 1996.
18. P. Fries. *Tagmeme sequences in the English noun phrase*. Number 36 in Summer institute of linguistics publications in linguistics and related fields. Benjamin F. Elson for The Church Press Inc., Glendale, CA, 1970.
19. P.-J. Gailly. Expressing quantifier scope in french generation. In *Proceedings of COLING 88*, Budapest, 1988.
20. M. Halliday. *An introduction to functional grammar*. Edward Arnold, London, 1985.
21. E. Hovy. *Generating natural language under pragmatic constraints*. L. Erlbaum Associates, Hillsdale, N.J., 1988.
22. M. Kay. Functional grammar. In *Proceedings of the 5th Annual Meeting of the Berkeley Linguistic Society*, 1979.
23. E. Keenan and Y. Stavi. A semantic characterization of natural language determiners. *Linguistics and Philosophy*, 9:253–326, 1986.
24. A. Kronfeld. *The Referential-Attributive distinction and the Conceptual-Descriptive Theory of Reference*. PhD thesis, University of California, Berkeley, 1981.
25. K. Kukich. *Knowledge-based report generation: a knowledge engineering approach to natural language report generation*. PhD thesis, University of Pittsburgh, 1983.

26. K. Kukich, K. McKeown, J. Shaw, J. Robin, N. Morgan, and J. Phillips. User-needs analysis and design methodology for an automated document generator. In A. Zampolli, N. Calzolari, and M. Palmer, editors, *Current Issues in Computational Linguistics: In Honour of Don Walker*. Kluwer Academic Press, Boston, 1994.
27. J. Lester. *Generating natural language explanations from large-scale knowledge bases*. PhD thesis, Computer Science Department, Universtity of Texas at Austin, 1994.
28. J. N. Levi. *The Syntax and Semantics of Complex Nominals*. Academic Press, New York, 1978.
29. B. Levin and M. Rappaport. The formation of adjectival passives. *Linguistic Inquiry*, 17(4):623–661, 1986.
30. D. McDonald. *Natural Language Generation as a Process of Decision-Making under Constraints*. PhD thesis, MIT, 1980.
31. K. McKeown. *Using Discourse Strategies and Focus Constraints to Generate Natural Language Text*. Studies in Natural Language Processing. Cambridge University Press, 1985.
32. M. Meteer. *The generation gap: the problem of expressibility in text planning*. PhD thesis, University of Massachussets at Ahmerst, 1990. Also available as BBN technical report No. 7347.
33. R. Passoneau, K. Kukich, J. Robin, and L. Lefkowitz. Generating executive summaries of workflow diagrams, 1996.
34. R. Quirk, S. Greenbaum, G. Leech, and J. Svartvik. *A comprehensive grammar of the English language*. Longman, 1985.
35. E. Reiter. *Generating appropriate natural language object description*. PhD thesis, Center for research in computing technology, Harvard University, Cambridge, MA, 1990.
36. E. Reiter. Has a consensus natural language generation architecture appeared and is it psycholinguistically plausible? In *Proceedings of the 7th International Workshop on Natural Language Generation*, pages 163–170, June 1994.
37. E. Reiter. Nlg vs templates. In *Proceedings of the 5th European Workshop on Natural-Language Generation (ENLGW-95)*, Leiden, The Netherlands, 1995.
38. E. Reiter and R. Dale. 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, August 1992. COLING.
39. E. Rosch. Principles of categorization. In E. Rosch and B. Lloyd, editors, *Cognition and Categorization*, pages 27–48. Lawrence Erlbaum, Hillsdale, NJ, 1978.
40. J. Searle. Referential and attributive. In *Expression and Meaning: Studies in the Theory of Speech-Acts*. Cambridge University Press, Cambridge, 1979.
41. Z. Vendler. *Adjectives and nominalizations*. Mouton, The Hague, 1968.
42. E. Williams. Predication. *Linguistic Inquiry*, 11:203–238, 1980.

Address for correspondence: Mathematics and Computer Science Dept.
Ben Gurion University in the Negev
Beer Sheva, 84105 Israel
elhadad@cs.bgu.ac.il