Università degli Studi di Roma "La Sapienza"
Dottorato di Ricerca in Ingegneria Informatica

Text Generation from MUC templates

Nicola Cancedda

XI-99-2

Supervisore:
Prof. Luigia Carlucci Aiello

Coordinatore:
Prof. Giorgio Ausiello

Nicola Cancedda

Text Generation from MUC templates

Dottorato di Ricerca in Ingegneria Informatica

XI-99-2

Università degli Studi di Roma "La Sapienza"

Author's address:
Nicola Cancedda
Dipartimento di Informatica e Sistemistica
Università degli Studi di Roma "La Sapienza"
Via Salaria 113, I-00198 Roma, Italy
e-mail: cancedda@dis.uniroma1.it
www: http://www.dis.uniroma1.it/~cancedda

Contents

Acknowledgments

I wish to thank all the people who helped me during my PhD. First of all, I'm indebted to Luigia Carlucci Aiello, my advisor, who constantly encouraged and supported me by giving me all the autonomy I needed and at the same time making sure that I didn't forget my main goal.

I would like to thank all the people at the Dipartimento di Informatica e Sistemistica of the University of Rome "La Sapienza", and in particular the members of the Artificial Intelligence group. They all contributed in creating an environment relaxing and stimulating in the right proportion.

I carried out a significant part of my work when I was a guest of the Artificial Intelligence Group at SRI International: I am deeply indebted to Jerry Hobbs and Ray Perrault for making such an experience possible, and to all the people of the Natural Language group, who inspired me through many stimulating discussions. A special thanks goes to David Martin, without whom experimental evaluation would not have been possible.

I express my gratitude to the members of the PhD Committee of the Dipartimento di Informatica e Sistemistica, who helped me in the accomplishment of this work. To John Bear and Emanuele Pianta as external referees, to Maurizio Lenzerini and Oliviero Stock as internal referees, and to Giorgio Ausiello as president.

Much good advice came by members of the Natural Language group of the "Istituto per la Ricerca Scientifica e Tecnologica" of Trento, which I would also like to thank.

Last but not least, special thanks to my family, who encouraged me in difficult moments.

Chapter 1

Introduction

A series of conferences, called "Message Understanding Conferences" (MUC) was held in the last years as a forum for discussing issues related to Information Extraction, a discipline born from Natural Language Processing. The core events of these conferences were formal evaluations aimed at comparing performances of systems developed by different groups. The output of these systems was a data structure called a "MUC template".

This thesis is about generating text from MUC templates.

MUC templates are a special way of representing part of the semantic content of a text, designed to be the output of a special class of computer programs. At a first glance, such a statement of intentions could sound minimalist, in that it looks geared towards a very specific class of applications. This first glance would be wrong, however, because under the clothes of a limited task an ambitious challenge is disguised: that of showing that it is possible, and can be convenient, to approach the problem of automatic text summarization from a knowledge-based perspective and under realistic practical constraints. This challenge is an ambitious one, because on one hand knowledge-based approaches are often synonym of limited scopes and expensive resource development, and, on the other, realistic practical constraints tend to be translated into a renunciation of any really "intelligent" processing.

The main contribution of this thesis is a new architecture for text generation designed with the purpose of fleeing both these specters. It is a knowledge-based architecture, because its input is in a symbolic form that can be interrogated using appropriate rules, and it is transformed into a final text by means of rules aware of the semantic content of the objects they operate on. Nevertheless, it achieves a degree of portability to different domains that is acceptable in many applicative scenarios by sharing domain-dependent resources with the Information Extraction system that produces its input. This balance results from an accurate decomposition of the generation process, both in terms of elaboration tasks and in terms of declarative resources to support them. To approach all the tasks in the generation process, competences in quite a few areas of Natural Language Processing had to be acquired and deployed, including, beyond Natural Language Generation, automatic text summarization, Information Extraction, and, to some extent, parsing and Machine Translation.

The proposed architecture has been implemented by the author in a proof-of-concept system, called GeM (for "**Ge**neration from **M**UC templates"), that produces short texts about management succession events from the templates that were adopted for the MUC-6 evaluation conference. GeM is written in ANSI Common Lisp and, though developed using Franz's Allegro Common Lisp and Harlequin's Liquid Common Lisp on Sun workstations, it is easily portable to other hardware and software platforms. GeM was a valuable testbed for new ideas and allowed the architecture to be validated through extensive experiments.

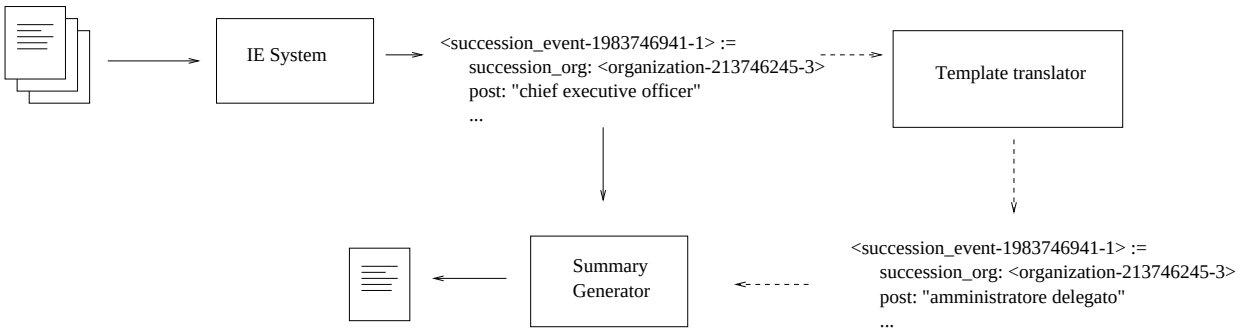


Figure 1.1: A schematic view of the overall context for summarization based on Information Extraction and text generation.

1.1 Motivations and objectives

Information Extraction and automatic text summarization belong to a class of applications that received a considerable impulse in the last years, mainly due to technological factors. Until recently, whenever a large quantity of information was stored in a computer memory, it was reduced to small, labeled units, given a rigid structure, and called a database. A steep drop in the cost of mass storage, however, made possible the transfer of whole *textual* archives from their original paper form into an electronic, machine readable form. Once this mass of texts was made available on a computer, the need for support tools to manage it grew: as traditional database technology was ill suited for dealing with unstructured, textual information, new approaches had to be devised. Information Retrieval, the task of returning to a user the subset of documents relevant to him, stepped from working on bibliographic data only — author, title, year, etc. — to full-text indexing. When the need for more and more sophisticated techniques for text management met Artificial Intelligence — and Natural Language Processing in particular — new trails were opened. One of them is Information Extraction, that consists in searching within a text for the answer to some fixed, predefined questions: "Does this text tell about a company merger?", "If so, which companies merged? When did the merger take place? What is the stock conversion ratio?" and so on. Another is automatic summarization: given a text, produce another text expressing the more important information in it. Faced to these new needs, Natural Language Processing underwent a methodological shift that brought researchers from studying toy situations to managing real-world scenarios.

As far as summarization is concerned, the prevalent approach so far is based on passage extraction: the source text is segmented into passages, each passage is scored according to some salience measure, and the best scoring passages are selected. This can be seen as a transposition of IR techniques from the text-base level to the single-text level: there is a set of passages instead of a set of documents, but, as in the IR case, the problem is that of returning to the user the subset made of the most relevant items. The "IR" approach is not the only possible one, though. An alternative, more in the tradition of Artificial Intelligence, consists in interpreting the source text, produce a representation of its content, prune what is superfluous and then re-generate a summary. This dissertation approaches summarization from this second, "AI" perspective, with the difference that —instead of completely interpreting the text and then filtering its content at the semantic level— an Information Extraction system is used as a front-end that filters out irrelevant information as soon as the text is analysed.

The overall scenario is thus the one depicted in Figure 1.1. A source text is first passed through an Information Extraction system that produces a representation of the relevant portion of its content in a so-called *MUC-template*. The present dissertation concerns what happens to

this template once it has been generated: it is fed to a Text Generation system (solid line) that produces the final summary.

This procedure for generating summaries has advantages and disadvantages over passage-extraction. One of the advantages, as will be shown in this thesis, is that a template translation module can be interposed between Information Extraction and text generation (dashed line), so that the final summary can be in a language different from that of the source text. This feature, as well as others that will be presented here, clears the path to a full range of applications unconceivable within the "IR" paradigm.

The scenario presented above assumes that suitable language generation techniques are available. Natural Language Generation, however, lags somewhat behind with respect to the methodological shift in language-related technologies mentioned above. On one hand, the vast majority of interaction in natural language between computer programs and human users still relies, as a matter of fact, on canned text: frozen strings in dialog windows, above all. On the other hand, most research was conducted on generation at the sentence level, and approached it as such a knowledge intensive activity that resource development costs allowed only for very limited applications. This thesis is also an attempt to narrow this gap. True text generation is concerned, as canned text would not suffice for our goal, but an eye is always kept on keeping required resources as cheap as possible, or on reusing possibly costly resources developed for other purposes, and making the best out of them.

1.1.1 Structure of the dissertation

This thesis begins with an introduction to two topics of interest for the rest of the discussion: Information Extraction (Chapter 2) and Automatic Text Summarization (Chapter 3). Information Extraction is relevant because it provides the input to the Natural Language Generation component that is the focus of this dissertation. Specific aspects differentiating IE from Natural Language Understanding are discussed, and the shape given to the field by Message Understanding Conferences is described. Automatic Text Summarization is relevant as well, because this is indeed the overall operation carried out by the concatenation of Information Extraction and Natural Language Generation. The most successful among methods for text summarization based on passage extraction are described, and advantages and disadvantages of the two approaches are contrasted. Chapter 4 expresses the requirements for a generation module to be coupled to an Information Extraction system, and discusses the properties of MUC templates as semantic representation for Natural Language Generation. The following Chapter 5 is the core of this dissertation: it presents in detail an architecture for text generation from MUC templates that follows from clearly stated design principles. The architecture results from balancing flexibility on one side and the need to reduce the effort of resource development on the other. An implementation of the architecture, called GeM, is also described. The architecture has been validated by means of experiments on GeM: these experiments and their results are presented and discussed in Chapter 6. The last chapter draws some conclusions and presents possible developments.

Chapter 2

Information Extraction

This thesis is about text generation from MUC templates. MUC templates are a special way of representing the specific part of the content of a text considered relevant by a user, explicitly conceived to be filled by an appropriate class of automatic systems: *Information Extraction* systems. Such systems were originally motivated by the needs of a few professionals daily dealing with large amounts of textual information in machine readable form. Low-cost storage media and the Internet, however, enabled a broader class of users to access huge amounts of unstructured, textual information and stressed the need for support tools. Text generation from MUC Templates is interesting because, combined with Information Extraction, makes for complete text summarization, a task with many potential applications. It is important to understand, however, that Information Extraction is a task relevant *per se*, and that in this case generation is mostly a way to make extraction results more readily accessible. Some knowledge of Information Extraction is thus necessary to build a clear framework and understand the scenario in which the generation system proposed in this thesis will fit. This chapter is devoted to an introduction to Information Extraction and to the methods developed for it.

2.1 Introduction

Now more than ever, people risk to be overwhelmed by an unmanageable quantity of information in textual form. As a consequence Information Retrieval —the study of the organization of large collections of documents aimed at an efficient access— received new impulse and found applications outside the context in which it was born: library science. Moreover, new research areas became active. Information Extraction (IE hereafter) is one such area.

IE is the task of recovering from a source text some specific pieces of information, whose nature is known in advance, and structuring them in a format suitable for populating a traditional database.

Computer programs capable of completing such a task have many possible applications. Work on IE was originally sponsored by intelligence agencies, whose analysts must skim every day through piles of paper in search of news on the specific field they specialize on. Many experiments, for example, concerned the extraction of information related to terrorist attacks in Central and South America, one of the first subjects to be chosen as a testbed [Sundheim, 1991]. From texts concerning such attacks, computer programs extracted, when mentioned, the time and the place of the event, the number of casualties, the way the attack was perpetrated, the perpetrator and other specific information.

Also the work of financial operators relies heavily on fast acquisition of information. Continuous streams of news from economic newspapers and press agencies must be checked, and this can require a good deal of effort. Given this pressing need, it is no surprise that many exper-

imental systems focus on financial texts. News stories can be searched, for instance, for information related to changes in management positions [DARPA, 1995] —to figure out what company is involved, what positions, which managers and so on— or to mergers and acquisitions [Sundheim, 1992]. Similar tasks are performed by systems operating in military [Marsh, 1986], medical [Friedman, 1986, Cavazza and Zweigenbaum, 1994] and industrial [Ciravegna, 1995] environments.

An Information Extraction module is also present in a system for distributing information about road traffic in the United Kingdom [Mellish *et al.*, 1992]: police reports about accidents with a potential impact on circulation are selected and analyzed, and useful information is distributed to car drivers using appropriate channels such as local radios and electronic boards.

Information Extraction, finally, can be used as a support to text categorization in an Information Retrieval application: the names of persons, organizations, geographic areas and so on can be used to better specify the topic of a document, thus enhancing retrieval performances [Hayes *et al.*, 1988, Jacobs, 1992, Jacobs, 1993, Gilardoni *et al.*, 1994, Bear *et al.*, 1997].

Tasks of this kind, on one hand, are too difficult for approaches based on statistic models typical of classical Information Retrieval, and on the other hand require capabilities that were not traditionally the main concern of researchers approaching natural language processing from the point of view of Artificial Intelligence. Whereas for many years the emphasis in Natural Language Processing has been on deep analysis methods, and most research was conducted on few selected sentences to show the appropriateness of a model for dealing with particular phenomena, with IE the focus shifted on large amounts of text naturally occurring in real world applications. As a consequence, accuracy had to be traded for robustness. IE differs from traditional Natural Language Understanding in that:

- the kind of information searched for in a text is known in advance: there is no longer the ambition of producing a semantic representation of all nuances of meaning in a text;
- as a consequence, only a (usually small) fraction of a source text is relevant and needs be analyzed;
- approaches must allow for wide coverage in significant, real-world domains: deep inference capabilities on a knowledge base modeling the domain at hand cannot be taken for granted, as such knowledge bases are usually too complex to be managed with current technologies, and anyway too expensive to create and maintain. More effort is oriented towards finding ways to compensate for parsing deficiencies and less on more complete parsing strategies;
- evaluation methods tend to shift from qualitative (the behavior on selected examples of certain linguistic phenomena) to quantitative (what portion of the information that should have been extracted from a text was actually recovered, and how much of the extracted information is correct and relevant).

These new goals put in evidence the necessity of a closer integration of linguistic and statistical techniques, the former providing usually a better accuracy and the latter wide coverage and superior robustness.

2.1.1 Text properties

A characteristic of IE techniques is that they must be tailored on the specific properties of the texts to be analyzed. The impossibility of managing a knowledge comparable in size to that available to a human reader leads to two possible approaches. The first consists in limiting the attention to texts from a small, well defined domain. In this case it is possible to model the domain through

entities and relationships, and a sufficiently accurate understanding of a good portion of the text is feasible. Otherwise, if it is not possible to impose such constraints on the domain, one must either give up the idea of performing a full analysis or find a way to detect relevant portions and analyse only them. Delimiting a domain not only leads to a manageable domain model: it also reduces the number of linguistic phenomena to deal with. If the task consists in extracting informations on company performances from Reuter reports, then it is not necessary to deal with problems specific to dialog understanding. Some important aspects to take into account are:

- in many cases the texts dealt with are not formed of complete, grammatical sentences. In many situations, for instance when some kind of diagnosis is involved, they are made of complex nominal phrases as in:

  > Right shoulder with calcium tendonitis.[Friedman, 1986]

- in some situation a sublanguage contains abbreviations and terms difficult to understand. A casual reader probably will not understand that:

  > gge rqd owner name n/k req ford gge if poss fst ford escort

  means

  > A garage is required. The owner (name not known) requests a Ford garage if possible. A front suspended tow for a Ford Escort is required [Mellish *et al.*, 1992].

- Another important property is the average length of sentences. Computational effort grows with sentence length, and this growth is steeper if ambiguities are present. If sentences tend to be long, as it may happen in newspaper articles, then special care should be devoted to adopting efficient techniques and keeping computation time reasonable.

An accurate examination of a collection of texts, usually referred to as a *corpus*, similar to those that will be dealt with during operations, is generally the first step in any effort aimed to designing an IE application.

2.1.2 Specific problems and techniques

The depth to which linguistic analysis is carried depends on the relative importance accorded to accuracy and robustness. Statistical techniques typical of IR are combined with NLP methods to obtain a satisfactory balance.

McDonald [McDonald, 1992] points out two problems, related to each other, that must be overcome by a system working on unrestricted text:

1. it should detect portions of text more likely to contain interesting information, and conduct a more accurate analysis of those portions only;

2. it should react to unknown words without loosing all the content of the text spans in which they occur.

The first point is motivated by the lack of appropriate techniques for managing the world-knowledge required for analysing unrestricted text to any depth, while the second stems from the unavailability (even in principle) of full-coverage dictionaries.

Most research on real-world applications is oriented towards extracting information only in well-delimited domains by skipping over all the text dealing with extraneous issues. A necessary

feature of such *partial parsers* is the ability to check that ignored portions of the text do not contain elements that significantly alter the content of analysed portions. In other words, the result of the analysis of a given text span should not change if additional spans are considered. As an example, consider the sentence:

> The yen plunged, also because of a prediction made by the treasurer for an unexpectedly high deficit.

A program designed for extracting informations about currency rates should be capable of giving to "plunged" the meaning of "decreased in value compared to other currencies", even if probably the available domain representation would not be detailed enough to allow a thorough understanding of the explanation. For the extraction to be effective, anyway, the parser must be reasonably sure that the content of the rest of the sentence does not invalidate the interpretation of the main clause, and this can happen, for instance, if it can be recognized as an explanation.

Most recent research [DARPA, 1995] pays a special attention to the issue of cross-domain portability. Most systems developed before 1995 required the dedicated effort of the system designers when a new subject was to be taken into consideration, or even when different pieces of information were to be searched for within an already studied domain. This has been an impediment to the diffusion of IE systems probably even more significant than performance. Ideally, the end user should be given the possibility to port an IE system to a new domain with a reasonable effort and, above all, without specific linguistic competences. Together with issues of efficiency and robustness, these considerations gained a definite advantage to approaches based on pattern-matching [Weischedel, 1995, Fisher *et al.*, 1995] and finite-state transducers [Appelt *et al.*, 1995, Grishman, 1995, Childs *et al.*, 1995, Lee, 1995], with significant effort being devoted to the application of machine learning techniques [Fisher *et al.*, 1995, Aberdeen *et al.*, 1995, Krupka, 1995, Gaizauskas *et al.*, 1995].

2.2 MUC conferences and templates

It is probably not incautious to say that most of the effort devoted to research on IE has been motivated by the TIPSTER program sponsored by the US Department of Defense. Starting from 1987, a series of evaluation conferences named Message Understanding Conferences (MUCs), now at their seventh edition, has been arranged to compare performances of systems developed by many different groups on a same task (or set of tasks).

In the last editions of these conferences, the overall procedure was as follows:

1. at conference announcement time, a description of the tasks is given to participants, together with representative texts and answer keys for one or more *scenarios*[1] similar to, but not equal to, that of the actual evaluation;

2. one month prior to the evaluation the actual scenario is disclosed, and participants receive a training set of texts and answer keys;

3. at the conference itself participants are given a test set of texts. The results from each system are compared with hand-crafted answer keys by means of an automatic scoring program. Participants do not have the chance of modifying their systems after being given the test set and before running the scoring program.

[1] Domains of interest and information to be extracted.

A number of issues about this evaluation procedure are object of discussion, but they fall outside the scope of this work. The general idea behind MUCs is to set up quantitative evaluations as fair as possible, trying to reach a good balance between achieving the best possible results and discouraging too ad hoc solutions. MUC conferences are also a privileged forum for discussing issues related to IE in general.

To give a better understanding of the tasks set up at MUCs a brief description of those proposed for the two last editions may be helpful. Four different tasks were proposed at MUC-6 [DARPA, 1995]:

- **Named Entity recognition (NE)**;
- **Template Element recognition (TE)**;
- **Co-Reference resolution (CoRef)**;
- **Scenario Template (ST)**.

Each of them builds on the results gathered at the previous stage.

NE is the task of detecting occurrences of names in a text. Together with the string representing the name, the system must specify if it is the name of a person, an organization or a location (city, country, etc.). Date and time expressions and numerical expressions must be correctly identified as well.

To complete the **CoRef** task, participants must identify which occurrences of references to entities actually co-refer to the same real-world entity. This encompasses recognizing different names for a same entity and resolving pronouns and definite descriptions as well as metonymical references.

For the **TE** task, relevant properties of the mentioned entities must be collected. For instance, for an organization the type (government, company, etc.) and the location may be required.

The first three tasks are designed to be as domain-independent as possible: in virtually every application of interest there will be entities to be identified and put in relation to some properties, and the categories adopted for such entities are generic enough to be easily portable to new situations. The **ST** task, instead, is the IE task as we previously introduced it: there is a specific type of information of interest in a particular domain, usually centered on a specific type of event. Descriptions of events of this type must be recognized, along with participants together with the role they play. So, for instance, as we will see in more detail in section 2.3, if we are interested in management succession events, relevant entities will be organizations and persons, and important pieces of information will be the involved position within the organization, who is the person leaving it and/or who is acceding to it, when this succession is taking place and so on.

In MUC-7 [muc7proc, 1998] an additional task was added: **Template Relation (TR)**. This task was motivated by the observation that the ST task is not monolithic in nature, because it requires the identification of several different relations among entities, each of which can be scored separately. Moreover, ST compels the system to encode a relation between template objects by means of either an object of a specific, domain-dependent type or as an attribute, whereas it is sometimes possible to recognize that a generic relation between entities holds, without being able to specify its domain-dependent type. The TR task consists in instantiating domain-independent relational objects —of type LOCATION_OF, EMPLOYEE_OF and PRODUCT_OF— with slots pointing to template objects.

While there is a clear increase in complexity going from NE to ST, reflected by decreasing performances, and the first tasks may be considered somehow intermediate stages in the completion of the last, it is interesting to observe that they can be very useful in themselves. A module

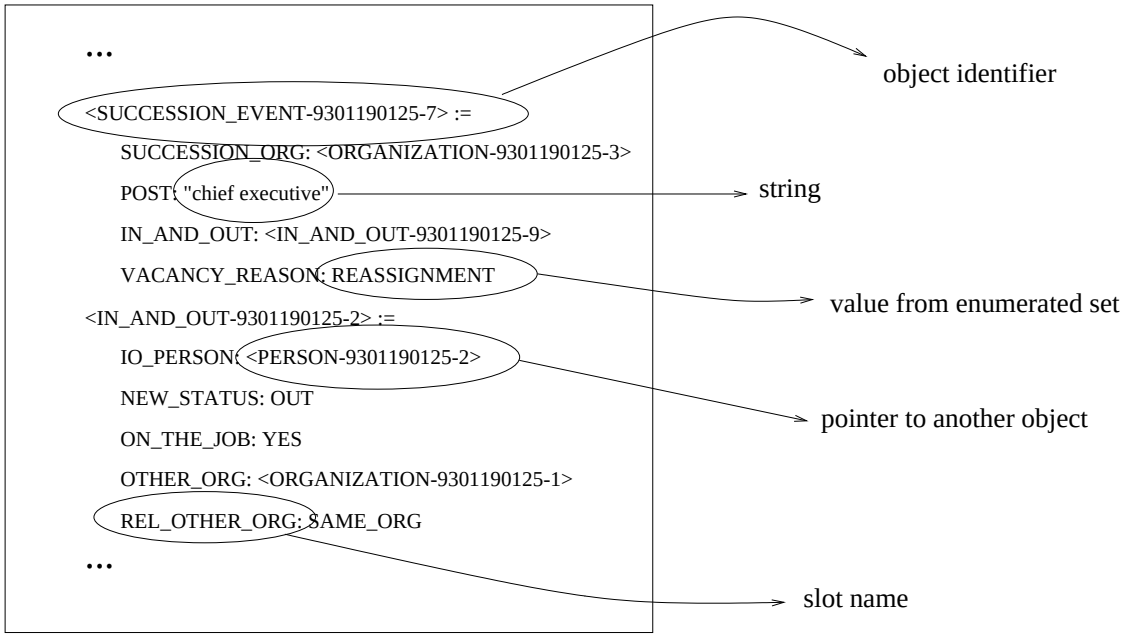


Figure 2.1: An example of a template object.

performing NE only can be valuable for indexing a text with the entities it mentions for retrieval, classification or filtering purposes. As entity names are usually formed by more than one word and display some flexibility (the same company could occur as "Coca Cola Company", "Coca Cola", "Coke", ...), IR traditional statistical models are not well suited. The information gathered performing TE and TR may be of use in the same kind of setting.

Most of the work presented in this thesis, although not strictly related to a domain, is based on experiments on texts and templates from the "Management Succession" scenario used in MUC-6 official evaluation. The following section is dedicated to a more accurate description of the ST task for this domain, especially of the template structure adopted as task target.

2.3 Definition of the management succession scenario

A *scenario* in MUC terms is a description as accurate as possible of the target structure of the extraction process together with the criteria governing the instantiation of such structure.

The structure to be constructed in a ST task is named a *template*. A template is a collection of *template objects*, or more simply *objects*, each characterized by a type. Every object in a template has its own unique identifier and, according to its type, a set of *slots*. Each slot is identified by a name, and may or may not receive a *value* by means of the extraction process. An example of a template object is shown in Figure 2.1. There are four kinds of values:

- a string of text as it appears in the source text;
- a normalized string;
- an element of a predefined, finite set;
- a pointer to another template object.

The scenario description provides, for each slot, a *definition* in natural language, some *minimum instantiation conditions* that must be matched in order for the slot to be filled, and possibly some

special usage notes giving more detailed instructions to follow in cases that could turn out to be ambiguous. The template must be dumped onto a file, according to a syntax specified in BNF.

Among all object types, two have a distinguished role. The first is the so-called *top level* template object. Each such object is univocally associated to a source text and gets instantiated anyway, even if the text does not contain anything relevant. Its `CONTENT` slot contains a possibly empty sequence of pointers to objects of the second distinguished type, that is to *scenario* template objects. Such objects capture the essential relation or event of interest in the task and have slots for pointers to other template objects.

The description of all template object types considered domain-independent or domain-neutral by the conference organizing committee are given to participants well in advance, at announcement time. In the case of MUC-6 and MUC-7 such object types were TEMPLATE, ORGANIZATION, PERSON and ARTIFACT. BNF definitions for these objects are reported in Figure 2.3. Note that in the `TEMPLATE` object definition the actual type of the fillers for the `CONTENT` role is left unspecified, as it depends on the scenario. Note also that the appropriate type for fillers of the ART_TYPE slot of the ARTIFACT object type is left unspecified as well, for the same reason as above.

Let's review briefly the definitions in Figure 2.3. First, a few comments on the BNF notation:

- items in angled brackets, such as <PERSON>, are object type identifiers. They appear both on the left hand side of BNF rules as object to be defined and on the right hand side, where they represent appropriate types of pointer slot fillers;
- items on the left of a colon, such as PER_NAME, are slot names;
- items on the right of a colon are descriptions of appropriate filler types for the slots. More precisely:
  - items in double-quotes stand for strings of text;
  - a list of items enclosed in braces enumerates a set of potential fillers for the slot;
  - items with neither double-quotes nor braces stand for normalized strings, that is, for strings coming from the source text and transformed according to rules given in the scenario definition;
  - the operator "$\hat{}$" indicates that exactly one filler of the preceding type must be present;
  - the "*" operator indicates that zero or more fillers of the appropriate type can be present;
  - a "+" indicates that one or more fillers may be present;
  - the "-" operator declares a filler as optional: there may be zero or one fillers for that slot;
  - the "|" operator, finally, indicates alternative fillers.

Two slots have a special meaning: the COMMENT slot and the OBJ_STATUS slot. The COMMENT slot is used only by the human expert who compiles the answer key to add comments on why he or she took certain decisions on some fillers. Also the OBJ_STATUS is used only by the human expert, to state that the instantiation of a certain specific object from a specific source text is optional, that is, that it is not an error neither to instantiate it nor to fail to do so.

Concerning the other slots:

- for the TEMPLATE objects:
  - DOC_NR: must be filled by the identification number of the source text contained in an appropriate SGML tag;

```
<TEMPLATE> :=
        DOC_NR:             "NUMBER"^
        CONTENT:            <scenario-specific-object>*
        COMMENT:            " "-

<ORGANIZATION> :=
        ORG_NAME:           "NAME"-
        ORG_ALIAS:          "ALIAS"*
        ORG_DESCRIPTOR:     " "-
        ORG_TYPE:           GOVERNMENT, COMPANY, OTHER^
        ORG_LOCALE:         LOCALE-STRING LOC_TYPE*
        ORG_COUNTRY:        NORMALIZED-COUNTRY-or-REGION
                            | COUNTRY-or-REGION-STRING *
        OBJ_STATUS:         OPTIONAL-
        COMMENT:            " "-

<PERSON> :=
        PER_NAME:           "NAME"^
        PER_ALIAS:          "ALIAS"*
        PER_TITLE:          "TITLE"*
        OBJ_STATUS:         OPTIONAL-
        COMMENT:            " "-

<ARTIFACT> :=
        ART_ID:             "ID"-
        ART_DESCRIPTOR:     "DESCRIPTOR"-
        ART_TYPE:           scenario-specific-set-fill
        OBJ_STATUS:         OPTIONAL-
        COMMENT:            " "-
```

Figure 2.2: Template object BNF definition for domain-independent objects as given to MUC-6 participants.

  - CONTENT: contains a list of pointers to scenario objects;

- for the ORGANIZATION objects:
  - ORG_NAME: is the name of the organization;
  - ORG_ALIAS: is a list of possible variants, or nicknames, for the organization;
  - ORG_DESCRIPTOR: is a noun phrase describing or referring to the organization without naming it;
  - ORG_TYPE: categorizes the organization as being a government entity, a company or some other kind of organization;
  - ORG_LOCALE: the (most) specific place where the organization is located, as it is explicitly asserted in the source text. Some location types from a predefined set (such as CITY, PROVINCE,COUNTRY etc.) may be specified as well;
  - ORG_COUNTRY: the country or region in which ORG_LOCALE is located. It is normalized in the sense that it must appear in exactly the same form as it appears in a list provided by the MUC organizing committee to all participants, and may be determined from knowledge sources other than the source text;

- for the PERSON objects:
  - PER_NAME: the proper name of the person;
  - PER_ALIAS: variants of the proper name, possibly more than one;
  - PER_TITLE: a title such as "Dr." or "Ms.";

- for ARTIFACT objects:
  - ART_ID: a unique identifier for the artifact;
  - ART_TYPE: a categorization of the artifact, with the inventory of categories depending on scenario definition.

As stated by the narrative accompanying the Management Succession scenario definition for MUC-6:

> This scenario concerns events that would be of interest to an analyst who tracks changes in company management. The event object captures the management post, the company, the current manager, and the reason why the post is or will be vacant. The relational and low-level objects capture information on who's "in" and who's "out", where the new manager came from, and where the old manager is going. A relevant article refers to assuming or vacating a post in a company and must minimally identify the post and either the person assuming the post or the person vacating the post.

The narrative is reflected in the BNF definition for the domain-dependent object types (Fig.2.3).

SUCCESSION_EVENT objects describe the management succession event:

- SUCCESSION_ORG is the organization where the post has been vacated and/or filled;
- POST is the position itself;
- IN_AND_OUT is filled with one ore more (usually no more than two) pointers to the object(s) capturing the relational information about the person assuming the post and/or the person vacating the post.

```
<SUCCESSION_EVENT> :=
        SUCCESSION_ORG:         <ORGANIZATION> ^
        POST:                   "POSITION TITLE" | "no title" ^
        IN_AND_OUT:             <IN_AND_OUT> +
        VACANCY_REASON:         {DEPART_WORKFORCE, REASSIGNMENT,
                                NEW_POST_CREATED, OTH_UNK} ^
        COMMENT:                "COMMENT" -

<IN_AND_OUT> :=
        IO_PERSON:              <PERSON> ^
        NEW_STATUS:             {IN, IN_ACTING, OUT, OUT_ACTING} ^
        ON_THE_JOB:             {YES, NO, UNCLEAR} ^
        OTHER_ORG:              <ORGANIZATION> -
        REL_OTHER_ORG:          {SAME_ORG, RELATED_ORG, OUTSIDE_ORG} -
        COMMENT:                "COMMENT" -
```

Figure 2.3: The template object BNF definition for domain-dependent objects in the management succession domain as given to MUC-6 participants.

- VACANCY_REASON is a general categorization for the reason why the post was, is, or will be vacated.

IN_AND_OUT objects maintain information about the person assuming or vacating a post:

- IO_PERSON: pointer to an object contributing information on the person assuming or leaving a post;
- NEW_STATUS: identifies the new status of the person with respect to the post. Basically IN or OUT, with less frequent variants for positions being held "ad interim", temporarily and so on;
- ON_THE_JOB: specifies if the status depicted by the object was effective at the time the article was written;
- OTHER_ORG: pointer to an ORGANIZATION object for the organization where the person was previously employed, in case of a person accessing a post, or for the organization that will employ the person, in case of a vacated post;
- REL_OTHER_ORG: specifies if the old or new organization employing the person accessing or vacating the post respectively is the same organization, a related organization, or a completely independent organization.

2.4 Summary of the chapter

In this chapter I illustrated the concept of Information Extraction, highlighting the motivations behind its development. Some applications, both real and potential, were outlined, as well as specific problems accompanying the task and possible approaches adopted to solve them. IE was

shown to be significantly different from Natural Language Understanding as it is traditionally intended, although of course related to it.

MUC conferences had a very important role in promoting research in the field: their motivation and structure were presented, with a special attention to the preparation stages, the tasks prepared and the definition of the target structure.

Most of the experimental work presented in this dissertation was performed on templates filled, either manually or automatically, as a consequence of performing the scenario-template task in the management succession domain. The structure of this template has been presented in some detail, as it will represent the input to the natural language summary generation prototype. A discussion of the properties of such template as input to a text generation system will be given in Section 4.3.

Chapter 3

Automatic text summarization

In Chapter 2 we introduced Information Extraction: what is usually intended with it, what motivates it, the specific problems it poses and the techniques that can be adopted to overcome them. Information Extraction is relevant to this thesis because it produces the semantic representation taken as input by the proposed generator. In the present chapter we introduce a somewhat related research field: *Automatic Summarization*. Automatic summarization is relevant to this thesis as well, because the combination of an IE system with the proposed generation architecture makes for a complete text summarization system.

The motivations behind automatic summarization are quite the same as those behind IE: to enable the user to consult large amounts of text.

Automatic summarization has been firstly investigated in the late 1950's and in the early 1960's, but it has been somewhat quiescent until advances in language processing technologies and broad diffusion of text in electronic format in this last decade have brought a new interest on it.

Automatic text summarization can mean many different things. Adopting a terminology that is becoming a standard [Hand, 1997]:

- a summary is *query dependent* if it addresses a need of the information seeker, *generic* otherwise;
- it may represent a *single document* or *multiple documents*;
- it may be made of sentences extracted from the source text (*extract summary*) or it may not use any of the wording from the source document (*generated text summary*);
- it may provide a general overview of the content (*indicative summary*) or it may act as a substitute for the actual document (*informative summary*).

In this chapter I present Text Summarization as it is generally intended by most researchers working on it and as it may be intended in relation with Information Extraction. Section 3.1 presents the approach based on extracting passages from the source text, whereas Section 3.2 illustrates summarization as the concatenation of Information Extraction and Natural Language Generation. Section 3.3 is devoted to comparing the two approaches.

3.1 Summarization by sentence extraction

Many authors (see [Barzilay and Elhadad, 1997, Aone *et al.*, 1997, Boguraev and Kennedy, 1997] among others) agree that there are two main possible approaches to text summarization, namely:

1. extracting sentences or other textual units from the source text according to some salience measure;

2. performing information extraction on a text and then regenerating from the obtained template.

Yet, a vast majority of all groups working on summarization adopted the first approach, so that a definite bias in favor of it has emerged. A significant evidence of this bias is perceivable in the way DARPA TIPSTER evaluation of summarization systems has been conceived (see paragraph 3.1.5 below, [Hand, 1997] and TIPSTER web page: `http://www.tipster.org`).

A comparison of the two approaches is the object of section 3.3. In the present section I will introduce summarization techniques based on sentence extraction.

3.1.1 Sentence extraction based on word frequency and indicator phrases

Early work on summarization [Luhn, 1968] started from the observation that usually in a text words that are more indicative of the content tend to be repeated more frequently. Sentences containing more such words are thus more likely to be central in the text and are thus suitable candidates to be inserted in a summary. Several heuristics may be adopted to improve results, such as giving different weights to sentences occurring in different portions of the text (such as the title, the abstract, the lead paragraph and so on), or to words occurring in different positions in a sentence. These approaches are much in the tradition of Information Retrieval in that they tend to consider a text simply as a sequence of words, without any special consideration for its meaning. In this way a good deal of domain independence may be achieved, but at the expenses of output quality. Techniques typical of IR, such as word stemming — to collapse different morphological variants of a same lexical item on the same term — word sense disambiguation — for discriminating among different senses of a same word — and the use of thesauri — to group together occurrences of synonyms — may be fruitfully adopted.

A different idea, that may be used by itself or may be combined with the word frequency approach, consists in observing that in some cases important passages are explicitly marked by means of special words or stereotypical phrases. Single words that have been considered as salience indicators are, for instance, "greatest" and "significant". Words like "impossible" or "hardly", on the other hand, may be given a negative salience score. *Indicator-phrases*, such as:

- "The main aim of the present paper is to describe..."
- "The purpose of this article is to review..."
- "Our investigation has shown that..."

are in general a more reliable clue for salience than individual words. Of course, the problem here is that they need to be detected automatically, and simple string matching techniques tend not to be adequate. Investigations found out, however, that there are many regularities in these phrases, and that most of them may be covered by a relatively simple grammar [Paice, 1990].

3.1.2 Sentence extraction based on rhetorical structure

Usually texts are not sequences of disconnected sentences: they tend to be organized according to a rhetorical structure that reflects the communicative goals of the author. The way texts are structured has been studied by many authors, and several theories of discourse structure have emerged. According to several of them [Hobbs, 1985, Mann and Thompson, 1988] it is possible to identify a set of *rhetorical relations* that may hold between segments of a well-formed discourse (see

[Hovy and Meier, 1994] for a discussion of different proposed set of relations). In most theories, rhetorical relations may be applied recursively, so that it is in principle possible to determine the *rhetorical tree* of a text: each node is labeled with a relation, and leaves correspond to actual text spans (figure 3.2 shows a rhetorical tree taken from [Marcu, 1998] for the text in figure 3.1).

Theories formal enough to lead to a computer implementation postulate two kinds of relations: *hypotactic* relations and *paratactic* relations. Hypotactic relations identify a span that is essential for the writer's purpose, called the *nucleus*, and a *satellite* that adds information but is not essential to the writer's purpose, whereas paratactic relations hold between spans of equal importance (so, they have two *nuclei*).

This asymmetry suggested a new possible approach for constructing summaries [Hobbs, 1993, Ono *et al.*, 1994, Marcu, 1997a]:

1. parse rhetorically the text to find its rhetorical tree;
2. score text spans according to their depth in the tree, privileging spans occurring at the end of paths with more *nucleus* labels;
3. select the best scoring spans, either comparing their score with a threshold or taking a fixed fraction of the source text;
4. perform some cosmetics to end up with a readable summary (usually this involves introducing some segments around the best scoring ones to present complete sentences).

This seems a very reasonable way of constructing a summary, and there is indeed some evidence of psychological plausibility for it [Marcu, 1997a], yet, it hides quite a few trapdoors. First of all, there is no general agreement on what set of rhetorical relations should be used. Apart from that, rhetorical relations are usually defined in a semi-formalized way, so that the same text may be given many different rhetorical trees. The analogy with traditional phrase structure grammars that drives rhetorical structure theories breaks down at the leaves level: whereas it is usually sufficient to store in a lexicon the possible parts-of-speech of each word to assign pre-terminals to the units in a sentence, there is no trivial way of assigning a rhetorical relation to a pair of text spans. Actually, there isn't even a trivial way to segment the text into a sequence of spans. If this is a problem when rhetorical analysis is to be done by a human expert, it is even more so when a computer program is supposed to complete it, as it should in an automatic summarization system [Marcu, 1997c, Corston-Oliver, 1998]. A discussion of these and some other problems with this approach to summarization is given in [Marcu, 1998].

3.1.3 Sentence extraction based on text cohesion

The property of a text of being structured according to rhetorical relations is usually indicated as *coherence*. Coherence is not the only property that distinguishes a text from a casual sequence of sentences. Sentences from a text usually tend, to some extent, to speak about the same entities and the same activities. This regularity makes for the so called *cohesion* of a text [Halliday and Hasan, 1976]. An important means to ensure cohesion is the use of *anaphora*, that is, of expressions referring to entities that have already been introduces in the discourse. Anaphora can be of several different kinds:

- anaphora by *word repetition*: occurs when the same word is used several times in a text. Of course, we are interested in the repetition of *content words*. Content words are those words actually contributing to the meaning of a sentence, as opposed to *function words* — articles, prepositions etc. — whose purpose is to mark relations among content words and make for the grammatical structure of the sentence. Obviously the knowledge that the article "the" has many occurrences in a text is of little help when constructing a summary;

[Smart cards are becoming more attractive[1]][as the price of micro-computing power and storage continues to drop.[2]][They have two main advantages over magnetic-stripe cards.[3]][First, they carry 10 or even 100 times as much information[4]][— and hold it much more robustly.[5]][Second, they can execute complex tasks in conjunction with a terminal.[6]][For example, a smart card can engage in a sequence of questions and answers that verifies the validity of the information stored on the card and the identity of the card-reading terminal.[7]][A card using such an algorithm might be able to convince a local terminal that its owner had enough money to pay for a transaction[8]][without revealing the actual balance or the account number.[9]][Depending on the importance of the information involved,[10]][security might rely on a personal identification number[11]][such as those used with automated teller machines,[12]][a midrange encipherment system,[13]][such as the Data Encryption Standard (DES),[14]][or a highly secure public-key scheme.[15]]

[Smart cards are not a new phenomenon.[16]][They have been in development since the late 1970s[17]][and have found major applications in Europe,[18]][with more than a quarter of a billion cards made so far.[19]][The vast majority of chips have gone into prepaid, disposable telephone cards,[20]][but even so the experience gained has reduced manufacturing costs,[21]][improved reliability[22]][and proved the viability of smart cards.[23]][International and national standards for smart cards are well under development[24]][to ensure that cards, readers and the software for the many different applications that may reside on them can work together seamlessly and securely.[25]][Standards set by the International Organization for Standardization (ISO), for example, govern the placement of contacts on the face of a smart card[26]][so that any card reader will be able to connect.[27]]

Figure 3.1: A segmented text. Text segments are numbered and enclosed in squared brackets.

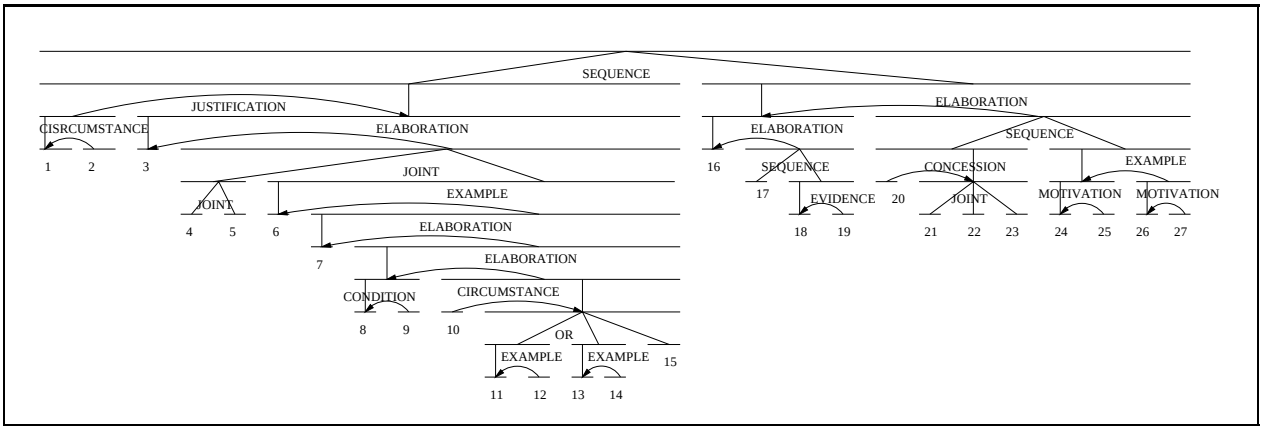


Figure 3.2: An example of a rhetorical tree. Lines go from a parent node to the nucleus-child (possibly more than one). Arrows connect satellites to their nuclei. The numbers on the leaves indicate the corresponding text segments.

- *pronominal* anaphora: an entity is referred to again by means of a pronoun (as in: "An anaphora$_1$ is an expression referring... *It*$_1$ can be...");
- anaphora by *definite reference*: a noun preceded by a definite article (as in "There was a queen$_1$ in the castle. *The queen*$_1$ was very sad.")
- *elliptic* anaphora: occurs when an entity fully introduced is referred to again in an abbreviated, incomplete form (as in "The famous writer Ernest Hemingway$_1$ was born in Oak Park, near Chicago, on July 21, 1898. *Ernest*$_1$ was soon introduced by his father to hunting and fishing...") or is omitted because it is obviously understood;
- anaphora by *synonymy*: an entity is referred to using a word with the same meaning as one used previously (as in: "The ship$_1$ was hit by a German shell. After a few hours, *the vessel*$_1$ sunk.");
- anaphora by *hypernymy*: a reference by means of a more generic expression than one previously used (as in "The cruiser$_1$ was hit by a German shell. After a few hours, *the ship*$_1$ sunk.")

[Mani *et al.*, 1998] presents a method for assessing sentence relevance based on text cohesion, and compares its results with the method based on text coherence. The idea is to build a graph in which there is a node for each word in the text, and edges are labeled with cohesion relations. Some "initial nodes" are then computed using a simple statistical technique, called *tf*idf*, that privileges words occurring more often in the text and penalizes words occurring frequently in many texts in a reference corpus. Initial nodes are given a score, and this score is propagated along graph edges according to a spreading activation algorithm. The salience of a sentence is thus computed as the sum of the scores of the words in the sentence. This work is somewhat preliminary, and methods were compared on a small text sample, but the authors conclude that although the method based on text coherence gives better results, the method based on text cohesion has its merits being much less knowledge intensive[1].

A second approach based on text cohesion is presented in [Barzilay and Elhadad, 1997]. Observing that of all cohesion relations lexical relations, such as identity, synonymy and hyperonymy,

[1]Only a few cohesion relations were actually used by the summarization system. Some cohesion relations, such as hyperonymy, are definitely knowledge intensive.

are most suitable to automatic identification, the authors rely on *lexical chains*. Lexical chains are subsets of the occurrences of nouns in a text that are related to one another by some lexical relation. The linguistic resource adopted to identify lexical relations is WordNet [Miller *et al.*, 1990]. Once lexical chains have been determined, and their strength scored according to some criteria, they may be used to select the most salient sentences. Three heuristics are presented to this purpose.

1. for each lexical chain whose strength exceeds a given threshold, select the first sentence in the text in which an element of the chain occurs;

2. score the words in a lexical chain, and find *representative words*, that is, words with significantly more occurrences than the average in the chain. Then select, for each chain, the first sentence in the text containing an occurrence of one of its representative words;

3. for each lexical chain, find a subsequence of contiguous text spans in which words in the chain are particularly dense: this subsequence is probably the part of the text dealing with the concept associated with the chain. Select the first sentence from every such sequence.

According to the authors, the first two heuristics tend to produce the same output, with the second being preferable in cases where discrepancies were found. Quite surprisingly, the most sophisticated heuristic, namely the third, was the least satisfactory.

There are mainly three problems with the lexical chain approach:

1. long sentences have a high probability of being inserted into the summary, even if they contain many irrelevant passages;

2. extracted sentences contain anaphora links to the rest of the text;

3. as exactly one sentence is extracted for each lexical chain, and there is no control on the number of lexical chains, there is no control on the length of the summary.

To overcome these problems, a more sophisticated, and costly, syntactic and rhetorical analysis of the source text would be necessary.

3.1.4 Other work based on sentence extraction

Summarization methods based on sentence extraction suffer from a general lack of cohesion and, for methods not based on the rhetorical structure, from a lack of coherence. [Strzalkowski *et al.*, 1998] approaches this problem by extracting whole paragraphs instead of sentences, following the intuition that paragraphs are usually self-contained units.

[Grefenstette, 1998] presents an interesting alternative to the dichotomy between sentence extraction and Information Extraction followed by Generation (IE+G hereafter). His purpose is to show the feasibility and the usefulness of text compression into a *telegraphic* style summary. Text undergoes some shallow parsing in order to detect superficial syntactic roles, such as main clauses and subclauses, sentence subjects, verb objects etc. Once the text has been tagged accordingly, it may be filtered pruning some class of elements. Grefenstette proposes eight styles of compression, each corresponding to a selection of roles as shown in Figure 3.3. The application proposed for this processing style is text skimming for the blind: instead of feeding the full text in a text-to-speech synthesizer, only the telegraphic version is given in input. Figure 3.4 shows an example source text together with the eight versions obtained with the corresponding compression styles.

Another somewhat eccentric approach is pursued by Endres-Niggemeyer [Endres-Niggemeyer, 1998a, Endres-Niggemeyer, 1998b]. Declared purpose of her research is to construct a cognitively plausible model of human expert summarization. Six expert summarizers took thinking-aloud protocols of nine summarizing processes each. The analysis of these

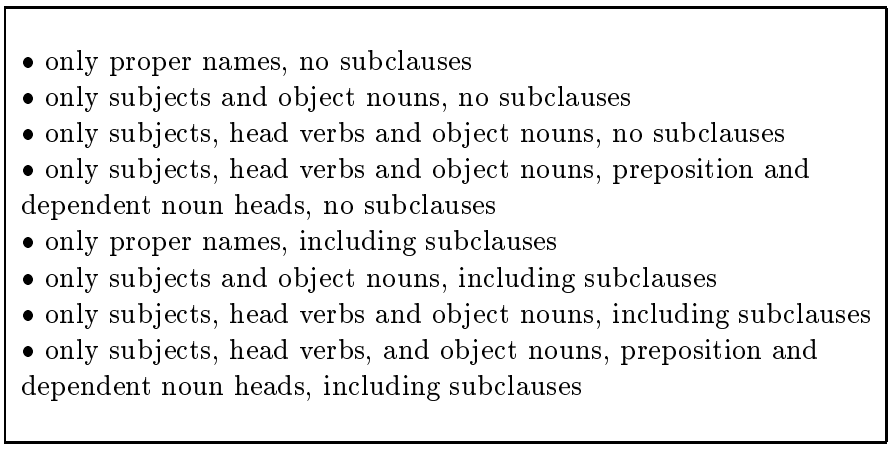


Figure 3.3: Eight possible filters on syntactic roles corresponding to eight styles of compression.

protocols, integrated by observation on data, yielded a set of 552 summarization strategies, such as:

- *no-truism*: Leave out what is trivial.
- *no-void*: Leave out what lacks content.
- *own-only*: No cited results, own results of the authors only.
- *relevant-contrast*: Important is what stands out from the rest.
- *relevant-new*: Important is what is new and original.

These strategies were included in an *intellectual toolbox* and animated in the SimSum system.

3.1.5 Summary evaluation for sentence-extraction-based summarizers

In the preceding discussion we left apart a thorny issue: how should summaries be evaluated? What distinguishes a good summary from a bad summary? These turn out to be much debated questions.

Summaries may be evaluated according to two criteria: their inherent quality as for informative content and readability and their usefulness in performing other tasks. Evaluations aimed at assessing the inherent quality are called *intrinsic*, while those oriented on their usefulness are called *extrinsic*, or *task oriented*.

Intrinsic evaluations

For summarization methods based on sentence extraction it is possible to import evaluation methodologies from Information Retrieval [Paice, 1990, Marcu, 1997a, Salton *et al.*, 1997]. A human expert selects the sentences that should be extracted, thus determining "the" correct summary, or *ground truth*. An automatic system can then be evaluated on the basis of how close it gets to this correct summary. Measures generally used are based on the so called *confusion matrix* (see fig.3.5):

- *precision* is the ratio between the number of sentences correctly extracted by the system and the total number of sentences extracted by the system (included those that were not selected by the human expert). In terms of the confusion matrix:

Source text:

Former Democratic National Committee finance director Richard Sullivan faced more pointed questioning from Republicans during his second day on the witness stand in the Senate's fund-raising investigation. While he grew flustered at times, the 33-year-old Sullivan stayed tight-lipped, downplaying concerns about former fund-raiser John Huang.

Telegraphic reductions:

• Richard Sullivan Republicans Senate Sullivan John Huang.
• Richard Sullivan pointed questioning. Sullivan tight-lipped concerns.
• Richard Sullivan faced pointed questioning. Sullivan stayed tight-lipped concerns.
• Richard Sullivan faced pointed questioning from Republicans during day on stand in Senate fund-raising investigation. Sullivan stayed tight-lipped concerns about John Huang.
• Richard Sullivan Republicans Senate Sullivan John Huang.
• Richard Sullivan pointed questioning. Sullivan tight-lipped concerns.
• Richard Sullivan faced pointed questioning from Republicans during day on stand in Senate fund-raising investigation. he flustered, Sullivan stayed tight-lipped concerns about John Huang.
• Richard Sullivan faced pointed questioning from Republicans during day on stand in Senate fund-raising investigation. he flustered at times Sullivan stayed tight-lipped concerns about John Huang.

Figure 3.4: A source text and eight telegraphic reductions. Emission time from a same text-to-speech synthesizer drops from 36 seconds for the original text to 20 seconds for the level 8 compression, that retains much of the source text.

	Sentences that should have been extracted	Sentences that should not have been extracted
Sentences actually extracted	A	B
Sentences actually not extracted	C	D

Figure 3.5: Confusion matrix correlating sentences actually extracted with those marked as to be extracted by a human judge.

$$precision = \frac{A}{A+B} \tag{3.1}$$

- *recall* is the ratio between the number of sentences correctly extracted by the system and those that should be extracted according to the human expert (including those that were not selected by the system). In terms of the confusion matrix:

$$recall = \frac{A}{A+C} \tag{3.2}$$

The obvious problem with such evaluations is that there isn't a single correct summary: there may be interchangeable sentences in a source text, and a summarizer should not be penalized only for choosing a sentence different from that selected by the human expert in such cases. A few proposals have been set forth for dealing with this problem. [Jing *et al.*, 1998] proposes to modify the usual way of basing precision and recall on a binary decision. If more than one expert is available to summarize each text, a weight corresponding to the fraction of experts who chose each sentence may be computed. Recall and precision can then be assessed computing the total weight of the sentences that were selected. This solution is not without problems, however. First of all, a summarizer that chose more than one member from a set of equivalent sentences would score higher than a summarizer that chose only one. Secondly, the correct selection of a sentence belonging to a set of equivalent sentences would weigh less than the selection of a sentence that does not have an equivalent in the source text, and this does not seem justified. In the same paper the authors propose also an alternative: identifying the sets of equivalent sentences in a text and considering each of them as a single unit. In other words, this would amount to considering a full "hit" the selection of any sentence from the set. This proposal does not suffer from the problems mentioned above. However, it still relies on the assumption that sentences are equivalent *pairwise*, while it may well happen that equivalence holds between subsets of sentences. For example, it could turn out that selecting both sentences 1 and 2 could be as appropriate as selecting sentences 4, 6 and 7 together, and this cannot be modeled without setting more complex constraints. Furthermore, it is not obvious that the process of detecting these equivalence classes would not introduce more noise (due to the difficulty of the task itself) than that induced by the presence of such classes, and this both if the classification is to be done manually or by means of some statistical correlation measure. [Johnson *et al.*, 1993] propose to generate manually a template of key concepts in a text and then measuring the matching according to such concepts but, again, there isn't a single

template of key concepts for a given text, and concept matching is a fuzzy operation. Another interesting observation is made in [Jing *et al.*, 1998]. Presenting an accurate and quite extensive experimental study on evaluation methods, it points out that precision and recall figures tend to be very sensitive to summary length: the very same system may perform differently when required to produce a summary 10% the size of the original text or 20%.

An intrinsic evaluation protocol not based on precision and recall is presented in [Minel *et al.*, 1997]. The proposed evaluation method consists in checking summaries according to four criteria:

1. *Number of Anaphora deprived of Referents*: the number of cases in which an anaphoric reference (such as those induced by a pronoun) is present in the summary and the referent is not available;

2. *Rupture of textual segments organized by Linear Integration Marker*: some textual connectives, such as "on the one hand" and "on the other hand", or "firstly" and "secondly", must occur in pairs. The number of cases in which such a connective occurs without the corresponding member of the pair are counted;

3. *Presence of "tautological" sentences*: a sentence is considered tautological if the information it provides is completely independent of the source text. The example reported is "Predicting the future is a difficult and uncertain exercise". The idea is to detect and penalize overly general abstracts;

4. *Legibility of the abstract*: a human subject is required to assess the legibility of the abstract and assign a "mark" from {*Very Bad, Mediocre, Good, Very Good*}.

The third criterion turned out to be very dependent on the degree of the reader's competence within the domain of the text: what is obvious to an expert may well be new to a beginner. Interestingly, no correlation was found between legibility and the first two criteria: according to interviewed readers, overall reading of the abstract allows them to overcome any localized lack of understanding caused by the absence of anaphoric referents.

Extrinsic evaluations

An extrinsic, or task oriented, evaluation assesses the quality of a summary by verifying how it increases a user's productivity in performing some other tasks. Typically, the task considered has been one of Information Retrieval: a user is given a collection of summaries and must decide if the text it comes from is relevant to fulfill some predefined information need (stated as a query). Her performance, in terms of precision, recall and execution time, is compared to that of a subject performing the same task using the source text. Ideally, by using summaries, precision and recall should not decrease, while execution time should drop.

A comparison among the performances on such a task by using summaries 10% and 20% the length of the source text, both automatically produced and handwritten, is presented in [Jing *et al.*, 1998]. Results are somewhat difficult to interpret. By far the best performances are obtained on 10% long, hand-crafted summaries. However, whereas automatic systems receive better scores on 20% summaries, there is a steep drop going from 10% to 20% hand-crafted summaries. Another interesting result is that execution time is far from proportional to summary length. This is explained by the fact that the quality of some automatic summarizer was not good enough to produce readable summaries, so adding some sentences significantly improved readability, and reduced reading time. Anyway, even with human-generated summaries, the time required when using 10% summaries was more than half that of using 20% summaries, and more than one tenth of that

necessary with the full source text. The authors conclude that, in extrinsic evaluations, no cutoff length should be imposed, as different systems may achieve their best results, as support tool, at very different summary lengths.

The same experiment showed also that caution should be posed in the selection of the query and of the texts to use in tests. Some queries are much easier than others, so all systems tend to score roughly the same. Furthermore, the relevance of some texts can be assessed very easily by simply looking at occurring words, so that almost any set of extracted sentences would do. Easy texts may be detected by checking if the results using a simple keyword method are high, while there is no obvious way to identify an easy query prior to the evaluation itself.

Another extrinsic evaluation protocol is described in [Minel *et al.*, 1997]. The way summaries affect text categorization and preparation of a synthesis by a human subject is assessed. Some numeric data are given for the system SERAFINE, apparently collected on a pretty small sample and not statistically very meaningful.

Extending the experience acquired with Information Retrieval and Information Extraction, DARPA is organizing a series of evaluation conferences on summarization under the umbrella of the TIPSTER program[2] [Hand, 1997]. Three tasks will be proposed to participants:

1. **indicative summary for text categorization**: summaries will be evaluated according to the way they help a human operator in categorizing a document into a predefined set of classes. Texts belonging to one of a predefined set of categories will be summarized by participant systems. Summaries will be generic, in the sense that the summarization system will not be told the category of each text. Output summaries will be given to human operators who will decide what category they fall in. Classification based on summaries will be scored against classification based on the full text as performed by a control group of human experts.

2. **ad-hoc indicative summary**: user-directed summaries will be evaluated to determine if they capture the information sought by the user. User needs will be stated as a query, and systems developed by participants will build summaries using the query as an indication of the user's interest. The assessor will review a query and judge whether or not each summary is relevant.

3. **informative summaries Q&A test**: summaries evaluated in this task are imagined as an intermediate stage in a typical report writing process, and will be assessed on the basis of the number of correct answers they provide to a set of questions reflecting mandatory aspects of the topic (i.e.: those that must be satisfied for a document to be judged relevant to that topic). As the same set of questions is used for all the documents, the challenge for the system is to understand the topic of a document and produce as short a summary as possible covering all obligatory aspects. Human judges will then mark on each text the sentences providing answers to these fundamental questions, and query-biased summaries will be scored according to their overlap with the sentences extracted by the judges.

More tasks are planned for the future:

- when a text is inserted in a document base it usually undergoes an indexing procedure to ease subsequent retrieval operations. There is the possibility that indexing could be more accurate if done on a summary of the text, instead that on the full text. Summarizers will be compared according to the effect they have on the quality of the indexing of a set of documents;

- the time saved by using a summary can be multiplied if it is possible to build a single summary from multiple texts on the same subject, factoring common information. A task to assess the quality of multi-document summaries will be designed.

[2] See http://www.tipster.org for up to date details.

For all these tasks, both precision and recall scores and execution time will be measured.

Conclusions

If there is one point the whole summarization community agrees upon it is that evaluating summaries is difficult. There is no single correct summary for a text, and readability measures tend to be highly subjective. Moreover, more accurate evaluations must be carefully designed and require a big effort in terms of time and resources, so that they can hardly be set up by individual research groups. It is predictable that evaluations announced under the TIPSTER umbrella will increase the interest, and the pressure, in the field, as it happened with TREC and MUC conferences in IR and IE respectively.

3.2 Summarization by information extraction and regeneration

Even if most efforts focused on summarization as sentence extraction, this is not the only possible solution. Looking at human activities, summary generation by sentence extraction seems more similar to skimming through a text marking the more significant passages. The activity of deliberately writing a summary is different: it goes through a deep understanding of the source text, the identification of the most salient *concepts* therein and the careful planning of a new text to express them concisely.

Reproducing the human way of writing a summary requires resolving a good portion of all problems studied in Artificial Intelligence, mainly because the complete understanding of a text is so difficult, and is thus far from being feasible in the foreseeable future.

Information Extraction is, however, a somewhat more feasible undertaking than full language understanding, and the representation of text content it produces can be adequate to the purpose of writing summaries.

This approach has been pursued by the Natural Language group at the Columbia University [McKeown and Radev, 1995, Radev, 1997], with the explicit motivation of generating a single summary from multiple texts, possibly coming from different sources, and of integrating background information from resources external from the source text.

The SUMMONS system takes as input MUC templates in the terrorist attack domain used for MUC-4.[3] More precisely, it assumes that an external retrieval system has recognized a set of texts reporting on a same event, and an IE system of the kind developed for MUC evaluations has generated the corresponding templates. Templates are analysed by means of high level patterns in order to discover some interesting configuration of events. Apart from discovering the very event itself, high level patterns are mainly intended to perform a contrastive evaluation of how the same event was reported by different sources. A set of linguistic devices generally used to render this patterns was determined through the analysis of a corpus of summaries in the terrorist attack domain. Firing of patterns instantiate semantic representations of sentences, that are in turn sent to a sentence generator based on the FUF engine and on the SURGE general grammar of English [Elhadad and Robin,]. The text planner has access to a database of descriptions of entities, such as persons and organizations. For example, such a database can contain, for the entity named "Tony Blair", the description "The British Prime Minister". Actually descriptions are not stored as strings, but in a semantic representation that can be used to generate the corresponding strings using the FUF/SURGE sentence generator: in this way descriptions may be integrated homogeneously within the sentence representations produced by the text planner. The database of descriptions is populated from the same newswires by a separated extraction module. An example

[3] Templates used in MUC-4 were significantly simpler than those used in the following editions. There was only a single object for each text and no links among objects.

MESSAGE: ID	TST-REU-0001
SECSOURCE: SOURCE	Reuters
SECSOURCE: DATE	March 3, 1996 11:30
PRIMSOURCE: SOURCE	
INCIDENT: DATE	March 3, 1996
INCIDENT: LOCATION	Jerusalem
INCIDENT: TYPE	Bombing
HUM TGT: NUMBER	"killed: 18"
	"wounded: 10"
PERP: ORGANIZATION ID	

Figure 3.6: Template for the first message included in the summary.

MESSAGE: ID	TST-REU-0002
SECSOURCE: SOURCE	Reuters
SECSOURCE: DATE	March 4, 1996 07:20
PRIMSOURCE: SOURCE	Israel Radio
INCIDENT: DATE	March 4, 1996
INCIDENT: LOCATION	Tel Aviv
INCIDENT: TYPE	Bombing
HUM TGT: NUMBER	"killed: at least 10"
	"wounded: 30"
PERP: ORGANIZATION ID	

Figure 3.7: Template for the second message included in the summary.

output, taken from [Radev, 1997], is shown in figure 3.10. This text is obtained from the templates in Figures 3.6 to 3.9.

3.3 A comparison between the two approaches

Summaries can be produced by extracting passages from the source text or by performing information extraction and then generating a new text. Which way is better?

As the reader probably guessed, there is no sharp answer to this question as both approaches have their own advantages and disadvantages.

First of all, the IE+G approach is inherently oriented towards informative summaries, whereas passage extraction can also be adopted to produce indicative summaries. Moreover, as some resources necessary to IE systems are constrained by the structure of the output template, the former approach requires the user to state her interests in advance and in a quite static way.

In the most simple instantiations of the sentence extraction approach, sentence salience is assessed relying on statistics only, or on superficial linguistic clues. Both information extraction and text generation are instead knowledge intensive, as they require quite sophisticated linguistic resources. This difference makes IE+G more costly, and gives sentence extraction an edge whenever the texts to be summarized do not come from a limited domain. This point must be taken with some caution, however. Among sentence extraction techniques, the most satisfactory, at least for the moment, seem to be those based on rhetorical parsing. Rhetorical parsing requires:

- the ability to segment the input text into text spans, usually at the clause level;

MESSAGE: ID	TST-REU-0003
SECSOURCE: SOURCE	Reuters
SECSOURCE: DATE	March 4, 1996 14:20
PRIMSOURCE: SOURCE	
INCIDENT: DATE	March 4, 1996
INCIDENT: LOCATION	Tel Aviv
INCIDENT: TYPE	Bombing
HUM TGT: NUMBER	"killed: at least 13" "wounded: more than 100"
PERP: ORGANIZATION ID	"Hamas"

Figure 3.8: Template for the third message included in the summary.

MESSAGE: ID	TST-REU-0004
SECSOURCE: SOURCE	Reuters
SECSOURCE: DATE	March 4, 1996, 14:30
PRIMSOURCE: SOURCE	
INCIDENT: DATE	March 4, 1996
INCIDENT: LOCATION	Tel Aviv
INCIDENT: TYPE	Bombing
HUM TGT: NUMBER	"killed: at least 12" "wounded: 105"
PERP: ORGANIZATION ID	

Figure 3.9: Template for the fourth message included in the summary.

Reuters reported that 18 people were killed in a Jerusalem bombing *Sunday. The next day,* a bomb in Tel Aviv killed at least 10 people and wounded 30 according to Israel radio. Reuters reported that *at least 18 people* were killed and *105* wounded. *Later the same day*, Reuters reported that *the radical Muslim group* Hamas has claimed responsibility for the act.

Figure 3.10: An example summary output by the SUMMONS summarizer.

- the ability to determine accurately the rhetorical relation between two text spans, not only at the leaf level but also at higher level in the parse tree.

The first subtask has already been done quite successfully in the past, both following a stochastic [Church, 1988, Brill, 1993] and a knowledge-based approach [Ejerhed, 1988, Abney, 1996]. However, all solutions rely on the assumption that the most appropriate units to parse rhetorically a text are spans of *contiguous* text, but this could turn out not to be always the case. Sentences containing embedded clauses, like the one that comes hereafter, for example, can be problematic:

> Sentences containing embedded clauses, like the one you are reading now, for example, can be problematic.

because one could want to separate the two and end up with only the principal sentence ("sentences containing embedded clauses can be problematic") included in the summary.

The second subtask, the automatic determination of rhetorical relations, is much harder. [Marcu, 1997c] presents an algorithm that, given a text segmented into text spans *and a description of the rhetorical relations occurring between each pair of spans*, computes all the possible rhetorical trees for the text, weighs them and selects the best. The problem, then, is that of determining the relations holding between atomic textual units. The way this is attempted is by relying on *discourse markers* and *lexico-grammatical constructs*. Relevant lexico-grammatical constructs include some uses of verb tense and aspect, certain patterns of anaphoric references and pronominalization, while discourse markers are usually cue phrases. For automatic relation determination purposes, most interesting constructs and markers are those that can be detected accurately by a shallow linguistic analysis. The idea, then, is to examine a corpus of texts rhetorically parsed by hand and find a correlation between occurrences of markers and constructs on one side and rhetorical relations on the other.

This approach suffers from the fact that the very same constructs and markers are used sometimes as discourse structure indicators, and sometimes with a *sentential* function only, that is, only to specify the meaning of a single sentence. Furthermore, most markers and constructs are ambiguous, in the sense that they are almost equally correlated with more than a single relation. The first difficulty is reduced by the observation that in most cases a shallow analysis of the context in which constructs and markers occur is sufficient to tell sentential uses from rhetorical uses. Marker and construct ambiguity, on the other side, is less critical, because relations may be attributed somewhat profligately, as the following constraint based process that leads to the determination of all possible rhetorical trees tends to keep only some, and prune all others.

Besides all these problems, [Marcu, 1997c] reports a 66% recall and a 68% precision score from a test on five texts manually parsed by 13 judges.

Turning to resource development cost issues, it appears now that also the sentence extraction approach, at least in its best performing version at the moment, is not "for free": a corpus of text rhetorically parsed by hand is required to train the parser. Experiences on this manual task are very preliminary, but it is not unlikely that there could be differences in marker usage in very different domains, so that a new training set could be necessary each time.

Moreover, if better scores were to be attained, it could turn out to be necessary to consider even more domain-dependent resources. [Marcu, 1998] concludes that attributing relevance to sentences only considering nuclearity in rhetorical relations is not sufficient to go beyond present performance figures: the semantic properties of each specific relation should be taken into account, and in some cases it could be necessary to exploit features not covered by rhetorical tree theory.

A problem that was immediately clear to people working on sentence extraction is that summaries tend to be neither coherent nor cohesive. This is definitely an advantage for the IE+G approach, as output text structure is in this case completely under control. This does not mean,

of course, that generating a text coherent and cohesive enough to sound natural is easy, but that at least problems such as anaphora with missing antecedents, inappropriate verb tense selection and others that could make the summary difficult to read are not an issue. Indeed, successful text planning algorithms such as those presented in [McKeown, 1985] and [Hovy, 1993] are inherently based on textual coherence, that ends up being the informative principle of the output.

Informative summaries are useful because they retain (almost) the same content as the source text in much less space. This compression is obtained by pruning marginal information and eliminating *intra-textual* redundancies. In many situations, however, one has to deal with many texts on the same subject. The availability of a summary for each text can be of great help, but what if also *inter-textual* redundancies could be eliminated? In other terms: wouldn't it be useful if *a single* summary could be generated from a collection of related documents? This problem has hardly been approached (see [Radev, 1997]), but IE+G seems in a definitely better position than sentence extraction, in this case. It is hard to imagine that sentence selection could go much further than presenting some kind of alignment of the passages extracted from different source texts, and any attempt to merge passages from different sources would greatly amplify traditional problems such as coherence and cohesion. IE+G systems, on the other side, could exploit the intermediate semantic representation provided by the templates to perform a merge at the content level, and then produce a single text. Though not straightforward, this merge is quite possible in principle. The hardest problem would probably be that of recognizing that two entities or events mentioned in different texts are indeed the same, and it is not much different from the co-reference problem encountered in standard information extraction, where two different references to the same entity *in the same text* must be conflated.[4] Moreover, merging entities from different texts is likely to be easier than detecting co-referring expressions in a single text, because it could rely on a much more complete representation of the entities themselves. Particular care should be posed in cases where the IE template does not contain explicit temporal and causal information about reported events, because there is the risk of producing misleading text. In the management succession domain, for example, the order in which succession events take place can be definitely informative.

Summarization efforts so far were oriented to produce summaries in the same language as the source text. Now, consider a case in which there is a huge amount of text potentially containing critical information that is written in a language completely unknown to the user. This is a quite realistic scenario for many of the applications for which IE was conceived. Informative summaries from these documents *written in the language of the user* could be of enormous utility. Informative or indicative summaries in the language of the user would be useful even in cases where translator were available: professional translators are costly resources, so a filter on the texts to be sent to them could save a great effort. Again, if *cross-linguistic summarization* is the goal, sentence extraction seems at odds. As there is no constraint at all on the sentences that could be extracted, translating automatically the summary is not qualitatively easier than translating the full source text. IE, on the other hand, would greatly simplify the task [Kameyama, 1997]: what should actually need a translation is not full sentences, but simpler strings extracted as such during the IE process. This happens because all the other information stored in the template is language-neutral, so that it is quite indifferent to perform generation into a language or another. Strings occurring in IE templates usually correspond to relatively simple noun phrases. Moreover, their variability is somehow limited and known in advance and, above all, some information on their structure could already have been extracted in the parsing process and could be reused. In the management succession domain, for instance, only position names definitely require a translation, whereas organization names can often be left in the source language without much detriment. The problem of translating position names is not trivial, but is definitely simpler than unconstrained

[4] It should be noted, however, that coreference resolution is one of the hardest problems in IE.

machine translation of naturally occurring text.

IE+G is better positioned than sentence extraction also when *contrastive and incremental* summary generation is considered. An interesting application of this kind is news service monitoring [Radev, 1997]. Sometimes several different messages, from the same or from different sources, relate to the same event. In such cases it could be helpful to have summaries in which conflicting information is highlighted. More precisely, one could assume that the reader has read previous messages on the same event, so that only additional or conflicting information would have to be expressed. This kind of processing is out of reach for sentence extraction techniques, whereas it can be attempted by IE+G because some inferencing on the semantic representation is possible.

A last option available to IE+G and not to sentence extraction is the inclusion of information coming from sources external from the input text. If some external repository containing background knowledge about mentioned entities is available, it can be useful to insert this information in the summary [Radev, 1997]. This can be done quite naturally within the IE+G approach: once involved entities are determined the external repository may be queried, and the resulting answer can be stored in a format suitable for access by the text planner. Summarizers based on sentence extraction cannot exploit this background knowledge because they do not have an explicit representations of the entities involved, and even if they had it, then they could not introduce the relevant background information into the extracted sentences.

In conclusion, the IE+G approach has many advantages over sentence extraction in terms of more sophisticated elaborations. These advantages depend mostly on the availability of an explicit representation of text content. This approach, however, is also somewhat more limited in that it needs domain dependent resources and is worse suited for generic or indicative summaries.

Chapter 4

Requirements

Natural Language Generation approaches presented in literature are very heterogeneous as for the task they accomplish and the input representation they start from. The present chapter introduces some terminology and illustrates the way text generation has been approached in the past. This will give the reader keys for interpreting the architecture description in the chapter that follows. Furthermore, the requirements that an architecture for generating text from MUC templates should meet are illustrated. A discussion of the properties of MUC templates as semantic input representation for text generation closes the chapter.

4.1 Natural Language Generation

Natural Language Generation (NLG) is the subfield of Artificial Intelligence concerned with the automatic production of text from data internally maintained in a computer application. NLG systems perform this task by combining knowledge about the language and knowledge about the underlying application.

Depending on the variety in the text to be produced and on the richness of the input representation, different approaches can be adopted, more or less demanding in terms of linguistic and computational resources.

NLG systems have been used, for instance, to:

- generate weather forecasts from graphical representations of weather maps ([Goldberg *et al.*, 1994]);
- summarize data extracted from a database or a spreadsheet ([Iordanskaja *et al.*, 1992]);
- explain medical information in a patient-friendly way ([Buchanan *et al.*, 1995, Cawsey *et al.*, 1995]);
- describe a chain of reasoning carried out by an expert system ([Swartout, 1983]);
- produce answers to questions about an object described in a knowledge base ([Reiter *et al.*, 1995]);
- produce reports summarizing the activity of telephone planning engineers ([McKeown *et al.*, 1994]);
- generate paraphrases of a conceptual models designed by a system analyst, to have them validated by domain experts ([Cancedda *et al.*, 1997]).

The rest of this section describes the tasks involved in the generation process and presents a typical architecture that performs them.

4.1.1 NLG Tasks

Despite the fact that NLG systems tend to differ from one another depending on the application, some consensus is emerging on identifying six basic activities to be performed in going from the input representation to the actual text [Reiter and Dale, 1997]. These activities do not necessarily correspond to separate modules, and indeed most systems perform different tasks within the same processing stage. They are:

1. Content determination
2. Discourse planning
3. Sentence aggregation
4. Lexicalization
5. Referring expression generation
6. Linguistic realization

Each task will be described in turn.

Content determination

Content determination is the process of deciding what information should be communicated in the text, and results in the creation of a set of *messages*. The message creation process and the way messages are represented depend on the specific application. Some formal representation is generally used, with labels representing —intuitively— entities and relations in the domain. In many applications content determination is also affected by *background knowledge* available to the system and by a *user model*, that can specify, for instance, what goal the user is pursuing, his or her level of expertise and, in the case of dialogue systems, what previous interactions took place with the NLG system.

Discourse planning

Discourse planning is the process of imposing ordering and structure over the set of messages to be conveyed. A good text is structured in a way that makes its content more easily available to the reader. The result of discourse planning is usually a tree, whose leaves are messages and whose nodes describe how messages are grouped together. Sometimes nodes are labeled with "rhetorical relations", specifying what relation —among a limited set of possibilities— holds among two or more groups of sentences.

Sentence aggregation

Sentence aggregation is the process of grouping messages together into sentences: it takes a tree structured text plan as input and produces a new plan whose leaves are combinations of messages that will be realized as individual sentences. Aggregation is not always necessary, as each message can be expressed as a separate sentence, but it can significantly contribute to fluency and readability. The aggregation system must decide both what messages to aggregate and what syntactic mechanism should be used to combine the messages. Message aggregation can be used, for instance:

- to introduce *simple conjunctions* between messages, so that two or more messages will be presented as a single sentence in the final text;
- if two or more messages have a common constituent, to combine them and *elide* the repeated constituent;
- for *set formation*: if the messages to be aggregated differ in only one constituent, it may be possible to replace them with a single sentence plan that contains a conjunctive constituent;
- to *embed* one clause as a constituent of another.

Cue words such as *also* can be added to increase readability in cases where aggregation was possible but was not performed.

Lexicalization

Lexicalization is the process of deciding which specific words and phrases should be chosen to express the domain concepts and relations that appear in the message. In many cases lexicalization can be done trivially by specifying a specific lexical item for each concept or relation in the domain, but in some cases fluency can be improved if the system can vary the words used.

Referring expression generation

Referring expression generation is the task of selecting words or phrases to identify domain entities. Most commonly, it is viewed as the task of including enough information in the description to enable the reader to unambiguously identify the referred entity. A distinction is generally made among three variants of the problem: the generation of the initial reference to an object, the generation of pronouns and the generation of definite descriptions.

Linguistic realization

Linguistic realization is the process of applying the rules of a grammar to produce a text which is syntactically, morphologically and orthographically correct. Operations to be carried out in solving this task include the formation of verb groups, the enforcement of agreement constraints, and phonological, morphological and orthographic adjustments.

4.1.2 The typical NLG architecture

There are many ways to distribute the tasks described in the previous Section among modules of an NLG system, ranging from having a pipeline of six modules, each one for a single task, to having each task coded as a set of constraints and using a single constraint-solver to deal with all of them at once. The most common architecture, however, is a pipeline with the following stages:

- *Text planning*, encompassing content determination and discourse planning;
- *Sentence planning*, that combines sentence aggregation, lexicalization and referring expression generation;
- *Linguistic realization*, that performs the task with the same name.

The intermediate representation at the interface between sentence planning and text planning is usually referred to as the *text plan*, whereas the representation used as input to the linguistic realizer is usually called a *sentence plan*. As we mentioned in describing discourse planning, a text

plan is usually a tree with individual messages on the leaves. There are many ways of representing messages, but usually the representation adopted is as similar as possible to the one selected for sentence plans, so as to make the work of the sentence planner easier. A wide variety of notations have been used for sentence plans, among which the most common are templates and *Abstract Sentential Representations* [Reiter and Dale, 1997]. In their simpler version, templates are strings of canned text with gaps, and sentence planning is limited to filling these gaps without any further manipulation. This mechanism, however, is too rigid for most applications, so that more flexible templates have been proposed [Geldof and Van de Velde, 1997, Cancedda *et al.*, 1997] that allow for morphosyntactic adjustments both in the "frozen" portion and in the gap fillers. Among all *Abstract Sentential Representations*, the best known is probably the *Sentence Planning Language* (SPL) [Kasper, 1989]. An SPL expression for a sentence is a feature structure with named attributes for controlling specific syntactic properties; for instance, there can be one for declaring the type of the sentence (declarative, interrogative, imperative), one for polarity (affirmative or negative), one for verb tense and so on.

Text Planning

Text planning, in the traditional architecture, covers the tasks of content determination and discourse planning.

Content determination is the activity of forming a set of *messages* from the knowledge sources available to the system. It is a very application-dependent activity, about which it is difficult to state general rules. Proposed approaches range from the use of plan-recognition techniques to infer the goals of the user and present her with the information she needs, to domain dependent rules and heuristics elicited from a domain expert.

While content selection draws from knowledge about what information should be communicated, discourse planning uses knowledge about how messages should be organized to produce a coherent text. The output of the discourse planner is the text plan: a tree whose leaves are messages and whose internal nodes state the discourse relations holding between different portions of the text. There is no general agreement on what set of relations should be used. The most common is probably the one provided by the *Rhetorical Structure Theory* (RST) [Mann and Thompson, 1988].

A very general way for performing text planning consists in "operationalizing" discourse relations, and use them as planning operators in the AI sense [Hovy, 1993, Moore and Paris, 1993]. Such operationalization amounts to specifying, for each relation, which preconditions must hold for the relation to be applied between two or more messages and what effect the use of such relation has on the reader's mental state. Though promising, such an approach is knowledge intensive and computationally expensive, so that only a few NLG systems actually adopt it.

It is often the case that the texts required for a given application conform in structure to a relatively small number of patterns. These patterns can be determined by a careful analysis of a corpus and by talking to domain experts, and can be coded into *Schemas* [McKeown, 1985]. Each schema is a pattern that specifies how a particular text plan should be constructed using smaller schemas or atomic messages, and also the discourse relation that holds between these constituents. The schema expansion mechanism can be started after the completion of content determination, but it is more common for the content determination system to operate 'on demand' whenever the text planner requires a message of a given kind. In this way content determination is interleaved with discourse planning, with the discourse planning process in overall control. Most schema-based systems allow general programming constructs, such as local variables and conditional tests, to be included in the schema.

Sentence Planning

Sentence planning encompasses three tasks: sentence aggregation, lexicalization and referring expression generation.

Sentence aggregation consists of combining together messages that can be better expressed by a single sentence. Rules to decide whether aggregation is appropriate, and of what type, can be derived from a corpus or, in some cases, from psycholinguistic knowledge on reading comprehension and from writing handbooks.

Lexicalization is the task of choosing words to express a concept or a relation, and is commonly seen as the task of converting an input graph, whose primitives are concepts and relations, into an output graph, whose primitives are words and syntactic relations. Powerful graph-rewriting algorithms have been developed that use "dictionaries" to relate domain primitives and linguistic primitives. Such dictionaries allow in general a single domain primitive to be mapped into a linguistic structure that includes multiple words, and vice versa. A simpler approach, suitable in many cases, consists in using decision trees to encode simple choice rules that are used to vary how concepts are expressed according to various factors. Decision trees can be used, for instance, to select different words to realize a concept in different contexts, to conform with different stylistic parameters or simply to add variety.

Referring expression generation is the task of selecting words to unambiguously identify domain entities. The amount of information needed to do this will depend on the current discourse context and on the type of the expression. Relatively little research has been carried out on the generation of initial references to objects. Two common strategies are either to give the name of the object (if it has one) or to include in the description its physical location. The generation of pronouns is also a somewhat neglected issue. A simple algorithm that works well in many cases is to use a pronoun to refer to an entity if the entity was mentioned in the previous clause, and there is no other entity in the previous clause that the pronoun could possibly refer to. Definite descriptions, finally, can be generated by including a base noun describing the object and then, if necessary, adding adjectives or other modifiers to distinguish the target object from all other objects mentioned in the discourse.

Linguistic realization

Linguistic realization is the task of generating grammatically correct sentences to communicate messages.

One of the most popular approaches consists in seeing realization as the inverse of the parsing process found in natural language analysis systems. In a parsing system a grammar is used to map from a surface sentence to a representation of the semantic content of that sentence. Under this model the realizer takes as input a representation of the semantic content of a sentence that is similar to the representations produced by parsers and produces a surface sentence that expresses this content. A number of algorithms have been proposed for this task, of which the best known is the semantic-head-driven algorithm [Shieber *et al.*, 1990]. A problem with the inverse parsing approach, however, is that the representations that are naturally produced by text planning and sentence planning are indeed quite different from the representations currently produced by parsers.

Another successful approach to linguistic realization is motivated by *Systemic Functional Linguistics*. Systemic grammar emphasize choice-making: the central task is not viewed as the finding of a chain of grammar rules which convert an input structure into a sentence, but rather that of making a series of increasingly fine-grained choices which determine the syntactic characteristics of the sentence under construction. These choices are generally implemented as queries posed to the intended semantic content and to the wider environment in order to determine what function words should be added, how words should be ordered, and so on.

Some NLG systems do not perform syntactic realization at all, because the content determination process directly specifies messages as text templates. This approach have an impact on the tasks described so far. In template-based systems, the text plan has templates as its leaves, possibly including linguistic annotations to allow for some flexibility [Cancedda *et al.*, 1997]. Sentence aggregation is usually not performed, or is very reduced in its scope, because templates do not specify the sentential content at a sufficiently abstract level to permit the appropriate manipulations. Lexicalization is subsumed by the content determination system, that chooses templates which contain appropriate words to describe domain concepts. Referring expression generation generates slot fillers for the template; this can be difficult because the NLG system does not know what entities are mentioned in the canned portion of the template. Realization, finally, may take care of agreement and other aspects of morphology and orthography, but it does not perform any syntactic processing.

The template approach often makes sense when only limited syntactic variability is required in the output texts. Another advantage of this approach is that templates are usually easier to understand for domain experts than mechanisms that manipulate complex structures, and this can be of help in the knowledge acquisition task.

4.2 Desiderata for a generation system coupled to an IE system

Text generation from MUC templates is an approach to summarization with advantages and disadvantages over traditional sentence extraction methods. The very same processing may be considered also from a different perspective, however: it can be simply a convenient way to present information extracted by an IE system. MUC template format was designed for populating a database: it is not well-suited to be read by a human operator. Actually, when humans need to figure out what the propositional content of a template is, they often end up literally scratching short sentences beside the most relevant entities.

Graphics may be very effective in presenting the content of the template to a human, especially when the human is informed of the template structure and exact content of the slots must be made available. Any graphic interface, anyway, must be learned and might not be the best choice for a casual user. It is probably not too ambitious to state that in many occasion the best suited way to present template content is by generating simple and concise sentences from it.

The present section is devoted to an analysis of the desiderata for a natural language generation system coupled to an IE system such as those evaluated in MUC conferences. The principal such desiderata we identified are:

1. **Fidelity**
2. **Completeness**
3. **Correctness**
4. **Conciseness**
5. **Readability**
6. **Portability to new applications**
7. **Efficiency**

The first five are properties of the text to be produced, whereas the last two are desired features of the text generator. In the following these properties will be illustrated in turn.

4.2.1 Fidelity

Fidelity is the property of expressing correctly the content of the template, that is, without leading the reader to believe that such content is different from what it actually is. It is probably the most obvious feature to require to a the summary generator as a presentation device, and indeed to any NLG system, but nonetheless it cannot be taken for granted. Natural language can be a powerful way for expressing concepts, but it is highly ambiguous, and words could easily be used in a misleading way.

4.2.2 Completeness

A second property that should be guaranteed is completeness: everything in the template should be presented somehow. Consequences of this are pretty obvious if one considers entities or events: if the template mentions a person stating that she no longer holds a given management position, this fact should be expressed in the text. Things are less obvious when side information is the concern, however. The template for management succession events, for instance, contains a slot named ON_THE_JOB (see Section 2.3). Its value must be one among YES, NO and UNCLEAR, and its purpose is to state whether the state of facts depicted was actually effective at the time the original text was written. Of course, one could always put sentences like:

> Bill Gates resigned as president and CEO of Microsoft. *It is unclear whether this transition is actually effective.*

Resulting summaries would tend to be repetitive and very unnatural, though. Moreover, they would not be *concise*, and conciseness is, as it will be soon discussed, a very desirable feature. The alternative, then, is to put this information into sentences in a more subtle way, by choosing appropriate terms and verbal tenses, for example.

The notion of completeness presented here applies to all NLG systems whose purpose is presenting the content of some underlying repository. Other classes of NLG systems, such as those aimed at helping the user in performing some activity, would need a different definition, possibly based on goal satisfaction.

Fidelity and completeness, together, can be viewed as follows: an operator (or a "perfect" IE system) should be able to fill an empty template by reading the summary, and this template should be identical to the original one.

4.2.3 Correctness

Sentences used for presenting template content should be grammatical sentences of the English (or of any other) language, whatever this means. Note that this requirement is not as strong as it could seem at a first glance: humans have a well developed capability of making up for syntactic errors, and they are helped, in doing so, by the redundancy of natural language. If sentences are strongly ungrammatical, however, their semantic content can be unrecoverable, and the summary would fail to be a valid presentation method. Correct sentences, moreover, tend to require less effort to read than ungrammatical sentences, even if the information content that can be recovered from both is identical.

4.2.4 Conciseness

The purpose of an informative summary is to convey the same information content as the original text *in much less space*, following the assumption that the reader can thus read and understand it in much less time. A MUC template is already a summary, in the sense that it contains all and only

Type	Defects	Skills
Semantic	Poor message choice Missing message class Redundancy due to lack of temporal recognition Redundancy due to lack of causal knowledge	Interesting message choice Appropriate message combination Appropriate detail filtering
Discourse	Sentences too long Missing ellipsis Overuse of pronouns Lack of hyperonymy Lack of parallelism	Appropriate use of pronouns, ellipsis and hypernyms Sentence length variety Consistent verb tense use
Grammar	Repetition of locative Repetition of temporal	Effective clause combining Appropriate use of conjunctions
Syntactic	Overuse of syntactic form	Appropriate syntactic choice
Lexical	Repetition of terms	Lexical variety Metaphorical usage

Figure 4.1: Fluency defects and skills to avoid them grouped by type of linguistic level.

the relevant information by hypothesis. Differently from what happens with sentence extraction, then, conciseness is not a matter of content selection, because what should be said is known in advance. But while in sentence extraction systems the way things are expressed is given, because the sentences composing the summary were already written by the original author, in the IE+G approach to summarization conciseness becomes a matter of how things are expressed. Caution must be devoted to figuring out how to factor information and avoid useless repetitions. As humans tend naturally to adopt a kind of "principle of economy" when they write about facts, a side effect of pursuing conciseness will be more natural-sounding summaries.

4.2.5 Readability

In introducing conciseness we said that an assumption underlying summarization is that it takes less time to recover the same information from a summary than from the original text. As we mentioned earlier, however (see Section 3.1.5) experimental studies [Jing *et al.*, 1998] show that the time required for reading a summary is not proportional to its length with respect to the source text. An hypothesized cause for this phenomenon is that some automatically generated summaries tended to be of poor quality, especially when their length dropped below a certain level, and the major effort required by poor readability more than compensated for the reduction in length.

In order for concise summaries to be read quickly, thus, they must also be highly readable. This means, for instance, that they should not contain overly complex and long sentences, even if this were the best way to pack information. Furthermore, the generated text should be made as coherent and cohesive as possible. The table in Figure 4.1, modified from one in [Kukich, 1991], lists and classifies fluency defects, together with skills a generation system should possess in order to avoid them.

Defects and skills are grouped according to the linguistic aspect of the text they are related to.

Semantic skills and defects are related to content determination. Selected *messages*, units of content, must be interesting, properly combined and at the right level of detail. Redundancies

caused by a poor domain model, causing the expression of information that could be easily (and even unconsciously) inferred by the reader, should be avoided.

Discourse defects and skills concern organization at the level of the whole text. Sentences should be of varied length and, in order to be understood, they should not be too long. Pronominal and elliptic referring expressions, as well as references by means of hypernyms, are common in human-written text, so they would be a desirable feature for automatically generated summaries. In using them, however, one must be sure that the actual referent can be inferred by the reader, and no misleading inference is induced.

At the grammar level, when semantic grammaticalization is performed, sentences must be organized by effectively combining clauses. Repetitions may be avoided by correctly factoring clauses and introducing conjunctions. The component responsible for semantic grammaticalization, furthermore, must ensure a good variety in the choice of syntactic forms.

Lexicalization must avoid repeating too often the same term and, ideally, be capable of using metaphorical language.

This list of skills and defects will be considered again in Section 6.3.2, where the readability of the summaries generated by an implementation of the proposed architecture will be assessed.

4.2.6 Portability to new tasks and domains

One of the main reasons for the little diffusion of sophisticated information extraction systems in real-world scenarios is probably the time and skill required to have them work on a domain different from that for which they were originally designed, or even for changing the information sought in texts within the same domain. At present, the cost of porting these systems to a new domain is such that an organization must carefully assess economical convenience before doing it.

Given this state of affairs, it is very important that the effort required for building resources for generation from the template be small.

4.2.7 Efficiency

Information extraction systems as those involved in the last MUC evaluations take a relatively short time to produce a template. The order of magnitude of this time is minutes. If text generation is intended as a means of displaying IE result, the time required by the generator itself should at least be comparable to that used up by the IE system, and should possibly be much shorter.

For most applicative scenario it would probably be preferable to produce summaries of a somewhat lower quality in a reasonable time than an extremely good summary in half an hour. This is surely true if summaries have to be delivered real-time, but it is also important for batch processing if throughput is an issue.

4.3 Characteristics of MUC templates as semantic representation for text generation

Although some authors claim generality for the solutions they propose, "the" general architecture for text generation is unlikely to exist, at least if one is not satisfied with a very generic task decomposition. To assess the appropriateness of a proposed solution it is fundamental that a discussion of desired properties of the output is paired with an evaluation of properties of the input. This section presents the characteristics of MUC templates as semantic representation for a natural language generation component, and compares them with the representation used by other systems.

Applications discussed in literature differ significantly on the properties of their input. McKeown's TEXT [McKeown, 1985], a natural language interface allowing a user to query the structure (as opposed to the content) of a database, accepts a query expressed in a very narrow fragment of English and a description of the database structure itself in a KL-ONE-style knowledge representation language. The natural language query is directly mapped onto a communicative schema according to the interrogative pronoun in it and some easily detectable structural properties. The algorithm for filling communicative schemata queries the knowledge base and instantiates messages.

PAULINE, the generator developed by Eduard Hovy [Hovy, 1993], assumes as a starting point a formal representation of the communicative goals of the writer. These goals concern expressing specific parts of the content of a frame-based knowledge base.

Several experiments have been carried on on generation from data in numeric format. Kukich's ANA [Kukich, 1991] produces stock market reports based on market statistics for a single day, FOG [Bourbeau *et al.*, 1990] generates daily marine local weather bulletins from meteorological data, and STREAK [Robin, 1994] writes single paragraph long reports on basketball matches.

PHRED, the system described in [Jacobs, 1991], is, together with the PHRAN analyzer, part of a natural language interface to a knowledge base concerning the UNIX operating systems, and especially information on how users can perform different operations.

The author himself participated in the realization of the SAX system [Cancedda *et al.*, 1997] for automatically generating a hypertextual description of a SADT conceptual model. Models of this kind are produced by system analysts as part of the process of designing large information systems. The input to the SAX system proper was a representation of the SADT model as a prolog knowledge base. The knowledge base was in turn derived from the representation used internally by a CASE tool.

MUC templates characteristics must be considered carefully in designing a generation system for expressing them in natural language.

Recall from Section 2.3 that there is a (logical) distinction, within a template, between *scenario* template objects, which capture the essential event or relation of interest in the scenario, and other objects devoted to the description of the entities involved. This distinction makes scenario template objects well suited to be used as focal elements around which to organize *messages*, that is, content units to be expressed by individual sentences. As a first approximation, we might say that the information related to a scenario template object may be sufficient to determine the main verb of a clause, whereas other objects contribute in specifying the subject and possibly the object and other complements. A template can thus be viewed as a collection of scenario template objects, where information about individual entities is factored into other objects and is referred to from within scenario template objects by means of pointers. This suggests that:

1. the template is too flexible for a "canned text" approach to be feasible: the number of possible sentences that should be stored is enormous and would contain many redundancies;

2. message variability is relatively little, however, if compared to that considered by other text generation systems: as messages are centered around scenario template objects, and these are of a very few types for each scenario, template content can be expressed using a relatively small number of possible sentence structures. This consideration is strengthened by the fact that sentences to be generated are all *expository* in nature: no interrogative or imperative structures needs be considered.

A very important feature of MUC templates is that they contain also, as slot fillers, textual strings. This is an important difference with frame-based knowledge bases such that the value of a slot is either a pointer to another object or an element in a predefined set, known in advance, because it makes the input *unpredictable*. As textual strings in MUC templates usually contain information

that must absolutely be present in the final text, a way to introduce these strings into the output must be conceived. Input structures without textual strings can be directly mapped onto sentence structures by specifying, in a lexicon and in a grammar, the relations holding between lexical items on one side and slot and fillers on the other. In principle one could consider textual strings as individual lexical items by guessing their grammatical properties on the base of the semantics of the slot they fill. In the management succession domain, for instance, textual strings occur where the proper name of persons and organizations must be specified, so that one can treat the string as a lexical item belonging to the *proper_noun* category. The information one can obtain from the semantics of slots, however, is not sufficient in general to constraint adequately the linguistic usage of the string. In the previous example, for instance, the fact that a filler is the proper name of a person does not say anything about its grammatical gender, so that, should the generator decide to pronominalize it, it would be impossible to decide whether to use "she" or "he". Other slots in a template can give even less clues on the grammatical properties of their fillers (consider, for instance, the slot named *description* in the management-succession domain). In some cases one can make available to the generator patterns for sentences generic enough to allow strings to be plugged in without much risk of ungrammaticality. In the case of template objects describing organizations, for instance, one can insert the name of the organization in the text most of the time, and possibly choose an underspecified hypernym for a definite reference (such as "The company", or "The government organization") based on the value of the ORG_TYPE slot. In other cases, however, this is not possible, because agreement features of the entity described in the string may need to be changed in order to ensure sentence grammaticality. A case of special interest in the management succession domain is that of the strings naming positions in organizations. Figure 4.2 contains some of the strings for position names taken from the answer keys handwritten for MUC evaluations. Now, if from a template it may be concluded that person A was appointed vice-president of General Motors and person B was appointed vice-president of General Motors, one could want to produce the sentence "A and B were appointed *vice presidents* of General Motors", whereas in the template the position is indicated in both cases in the singular form: "vice president". In this particular case it would be possible, of course, to treat "vice president" as a single lexical item and form the plural as for other common nouns, by appending a final "s". This solution, however, is not general enough because it would not work with positions such as "vice chairman of the operations". In such a case, in order to correctly utter the plural "vice chairmen of the operations" the only solution is to find the syntactic head of the phrase ("chairman" in the example) and replace it with the corresponding plural form.

To achieve this, some kind of parsing is necessary. As we will see later in discussing the proposed architecture, this can lead to different design decisions according to the degree of independence that one chooses to obtain between the IE system and the text generation system.

Position names taken from answer keys can be definitely complex noun phrases. Moreover, the lexicon necessary to cover all words that could occur in position names is virtually unlimited (one could be vice president for anything): a complete parse of position names would require extensive linguistic resources and would slow down the system significantly. This obstacle appears somewhat less serious, however, when one considers position names as they are extracted by an actual high quality IE systems, such as FASTUS [Appelt *et al.*, 1995, Hobbs *et al.*, 1997]. Some of these are shown in Figure 4.3.

Position names extracted by actual systems lack some of the variability found in those coming from the handwritten answer keys. A reason for this is of course that the IE system itself has necessarily a limited coverage. Note, however, that while sometimes this limitation may end in the IE system not recognizing a position name, in many others it just cuts off part of it. Whereas the human responsible for answer keys wrote:

"vice president of merchandise and marketing"
"director of marketing"
"vice president of trade relations"
"deputy general manager"
"controller"
"co-chairman"
"vice president, planning"
"co-chief executive officer"
"co-chief executive"
"deputy chairman"
"finance director"
"finance chief"
"nonexecutive chairman"
"non-executive chairman"
"group president, information services group"
"one of two vice chairmen"
"vice chairmen"
"head of GM's European purchasing operations"
"senior vice president of communications"
"no title"
"vice-chairman"
"executive vice president of AT&T in charge of a newly formed seven-member Global Operations Team"
"group executive"
"senior vice president for marketing and strategic planning"
"vice president of manufacturing"
"co-chief executives"
"a managing director"
"a vice chairman"
"senior vice president, operations"
"senior vice president, special projects"
"a senior vice president"
"senior vice president of human resources"
"executive"
"communications director"
"human-resources vice president"
"director of human resources"

Figure 4.2: Some position names extracted from handcoded answer keys for MUC templates.

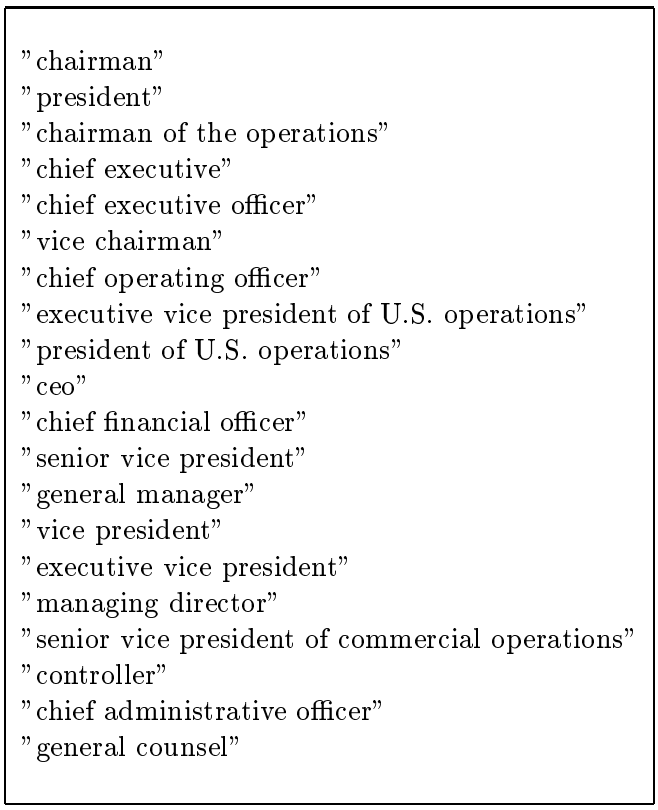


Figure 4.3: Some position names as extracted by the FASTUS IE system.

> "executive vice president of AT&T in charge of a newly formed seven-member Global Operations Team"

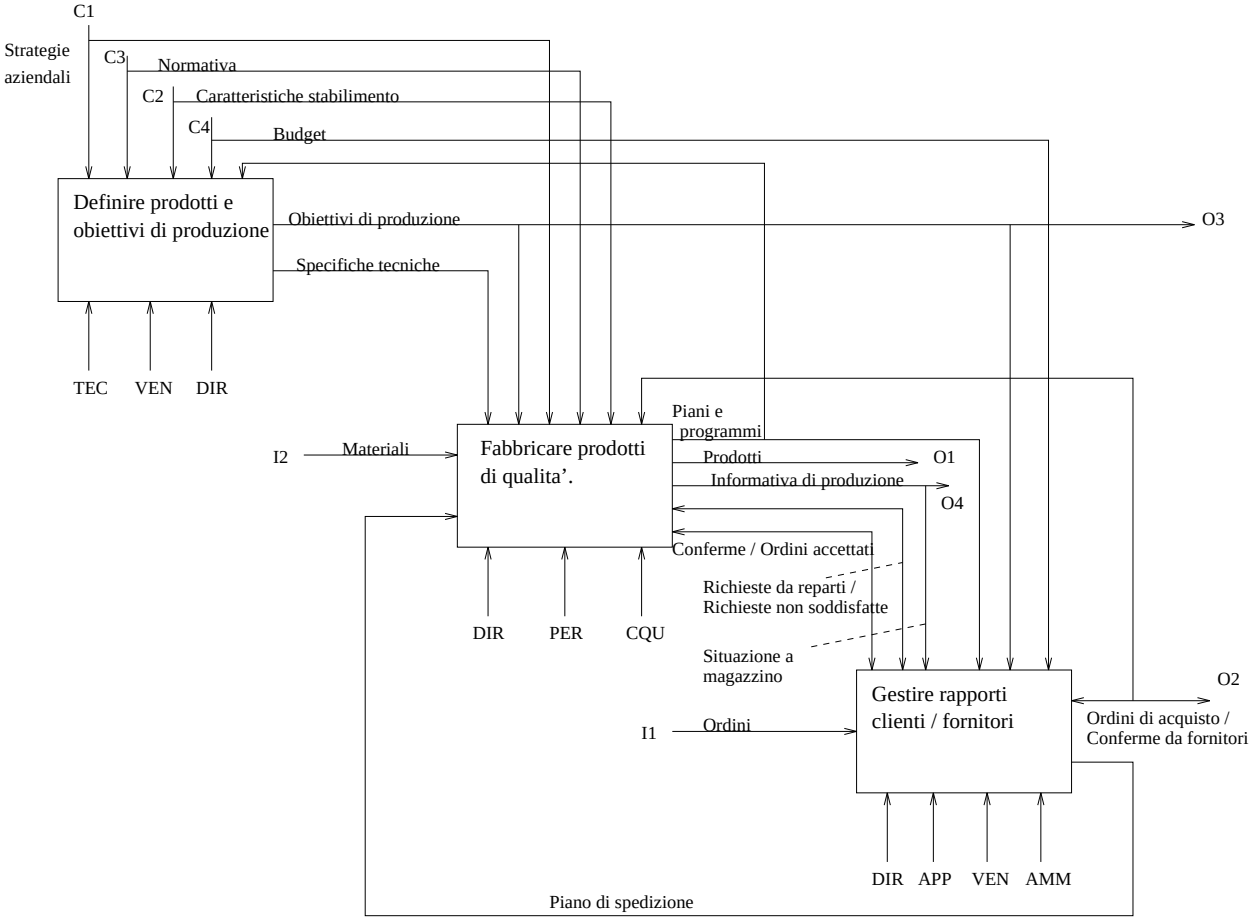
the IE system will probably extract:

> "executive vice president of AT&T"

As the text generator will eventually be coupled to an actual IE system, and not to a human annotator, it seems unreasonable to declare the task infeasible and give up just because a sufficient coverage would be too costly to achieve. The issue of linguistic coverage in textual string parsing and its relations with design decisions will be examined again later in this chapter, when the proposed architecture will be presented in detail.

This section discusses properties of MUC templates as input to a text generation system, and previous considerations were based on the assumption that summaries were to be generated in the same language as the source text. If cross-linguistic generation is pursued, textual string analysis is a fundamental step in their translation. This will be discussed in Section 5.5.

The problem of textual strings in the input of a Natural Language Generation system is not new. SADT models used as input to the SAX generator [Cancedda *et al.*, 1997] display much the same property. In that case, too, the input has a formalized component — the rules according to which interaction of processes and streams of data is represented in diagrams — and an informal component based on textual strings: the labels given by analysts to processes and streams (an example SADT diagram is depicted in figure 4.4). As will be described in detail later, the approach developed for the generation of SADT model descriptions at the sentence level was modified to suit the similar situation encountered in expressing MUC templates.



GESTIRE LA PRODUZIONE DI BENI INDUSTRIALI

Figure 4.4: A SADT diagram.

Traditional text planning systems based either on explicit planning [Hovy, 1993] or on communicative schemata instantiation [McKeown, 1985] are essentially top-down: there is a knowledge base of potentially relevant information, and propositions are extracted from it to satisfy some communicative goals. In the case of generation from MUC templates, however, the communicative goal boils-down to "say everything in the knowledge base". As observed in [Marcu, 1997b], when there is a goal of this kind top-down approaches cannot ensure that all the content of the knowledge base will be expressed. An alternative, then, consists in adopting a bottom-up approach: content is clustered into messages for which a linguistic realization is known to exist. Optionally, messages are manipulated and aggregated in order to identify units more suitable to lead to concise and natural sentences. Messages can then be organized according to rhetorical principles in order to obtain a coherent and cohesive plan, and be sent to the content realizer. With respect to the traditional NLG architecture, then, this solution inverts the order in which sentence aggregation and discourse planning are performed. [Marcu, 1997b] presents an algorithm for building a rhetorical tree for a text in a bottom-up fashion, in a way that ensures completeness over the initial content specification. Once the tree is built, discourse markers known to be correlated to relations by means of an empirical study may be inserted to achieve coherence.

This method, though very appealing in some cases, assumes that one starts with a set of *semantic units*, corresponding to messages, and the knowledge of all the semantic relations holding between pairs of such units. Marcu indicates MUC templates as a potential source of input rich enough to characterize rhetorical relations between semantic units. This should be done by inspecting the nature of the paths connecting two objects (that is, based on the semantics of slots). Indeed, in some cases a MUC scenario template could provide slots for causal and temporal relations between scenario template objects, and such slots would provide a suitable clue to establish rhetorical relations between units. In general, however, if messages are organized around scenario template objects, the only relation that can be safely established is one of conjunction, and it trivially holds between every possible pair of sentences. The management succession domain is an example of such a situation: no slot identifies a succession to be a consequence of another succession, nor there is one to state that a succession took place before or after another. This general lack of causal and temporal relations is also an argument for adopting a bottom-up approach: as semantic knowledge cannot be relied upon in structuring the text, it is even more important that sentence aggregation is performed at its best. If aggregation is to be carried out on a fully specified text plan, however, it will be either significantly inhibited, if only adjacent messages are aggregated, or complicated by the need of manipulating the text plan as a whole in a sound way. A bottom-up approach, instead, allows sentence aggregation to be performed thoroughly and builds the text plan only once aggregation has been completed. These observations are confirmed by the fact that cases in which the communicative goal was "say everything in the knowledge base", and a top-down approach to text planning was successful [Cancedda *et al.*, 1997, Not and Pianta, 1995], were characterized by richer input representations in terms of temporal and causal relations.

The communicative schemas needed for a top-down approach, moreover, are complex domain-dependent resources that should be rewritten from scratch every time the generator is brought to a new application domain. As we will see, design choices have to mediate in the trade-off between coherence on one side and ease of portability to new domains on the other.

4.3.1 A logical perspective

The previous part of this section examined input formalisms adopted in text generation and discussed properties of MUC templates from a linguistic point of view. A different perspective on MUC templates, however, will turn out useful. Consider the fragment of template in Figure 4.5. The template notation may easily be translated into a conjunction of literals:

- object types and slot names are mapped onto predicate names;
- individual objects are mapped onto constants;
- string and atomic slot fillers are mapped onto constants.

The template fragment in Figure 4.5 can thus be expressed as the conjunction of the following positive literals:

$template(TEMPLATE-1)$
$doc\_nr(TEMPLATE-1, "930106012300)$
$content(TEMPLATE-1, SUCCESSION\_EVENT-1)$
$succession\_event(SUCCESSION\_EVENT-1)$
$succession\_org(SUCCESSION\_EVENT-1, ORGANIZATION-2)$
$post(SUCCESSION\_EVENT-1, "president")$
$in\_and\_out(SUCCESSION\_EVENT-1, IN\_AND\_OUT-1)$
$in\_and\_out(SUCCESSION\_EVENT-1, IN\_AND\_OUT-2)$
$vacancy\_reason(SUCCESSION\_EVENT-1, REASSIGNMENT)$
$in\_and\_out(IN\_AND\_OUT-1)$
$io\_person(IN\_AND\_OUT-1, PERSON-2)$
$new\_status(IN\_AND\_OUT-1, OUT)$
$on\_the\_job(IN\_AND\_OUT-1, UNCLEAR)$
$in\_and\_out(IN\_AND\_OUT-2)$
$io\_person(IN\_AND\_OUT-2, PERSON-3)$
$new\_status(IN\_AND\_OUT-2, IN)$
$on\_the\_job(IN\_AND\_OUT-2, UNCLEAR)$
$other\_org(IN\_AND\_OUT-2, ORGANIZATION-2)$
$rel\_other\_org(IN\_AND\_OUT-2, SAME\_ORG)$
$organization(ORGANIZATION-2)$
$org\_name(ORGANIZATION-2, "Fox\ Broadcasting\ Co:")$
$org\_descriptor(ORGANIZATION-2, "the\ Fox\ network")$
$org\_type(ORGANIZATION-2, COMPANY)$
$org\_locale(ORGANIZATION-2, Hollywood)$
$org\_country(ORGANIZATION-2, United\_States)$
$person(PERSON-2)$
$per\_name(PERSON-2, "Jamie\ Kellner")$
$per\_alias(PERSON-2, "Kellner")$
$per\_title(PERSON-2, "Mr:")$
$person(PERSON-3)$
$per\_name(PERSON-3, "Lucille\ S\ Salhany")$
$per\_alias(PERSON-3, "Salhany")$
$per\_title(PERSON-3, "Ms:")$

Not any literal constructed using appropriate predicate names and constants from the template can actually occur in such a representation: the set of legal conjunctions is constrained by the BNF description of the template. The description itself, indeed, can be translated into a set of axioms:

<TEMPLATE-1> :=
  DOC_NR: "9301060123"
  CONTENT: <SUCCESSION_EVENT-1>

<SUCCESSION_EVENT-1> :=
  SUCCESSION_ORG: <ORGANIZATION-2>
  POST: "president"
  IN_AND_OUT: <IN_AND_OUT-1>
  <IN_AND_OUT-2>
  VACANCY_REASON: REASSIGNMENT

<IN_AND_OUT-1> :=
  IO_PERSON: <PERSON-2>
  NEW_STATUS: OUT
  ON_THE_JOB: UNCLEAR

<IN_AND_OUT-2> :=
  IO_PERSON: <PERSON-3>
  NEW_STATUS: IN
  ON_THE_JOB: UNCLEAR
  OTHER_ORG: <ORGANIZATION-2>
  REL_OTHER_ORG: SAME_ORG

<ORGANIZATION-2> :=
  ORG_NAME: "Fox Broadcasting Co."
  ORG_DESCRIPTOR: "the Fox network"
  ORG_TYPE: COMPANY
  ORG_LOCALE: Hollywood
  ORG_COUNTRY: United States

<PERSON-2> :=
  PER_NAME: "Jamie Kellner"
  PER_ALIAS: "Kellner"
  PER_TITLE: "Mr."

<PERSON-3> :=
  PER_NAME: "Lucille S. Salhany"
  PER_ALIAS: "Salhany"
  PER_TITLE: "Ms."

Figure 4.5: A fragment of a MUC template.

$\forall t, s(content(t, s) \supset template(t) \wedge succession\_event(s))$
$\forall s(succession\_event(s) \supset \exists o(succession\_org(s, o)))$
$\forall s, o(succession\_org(s, o) \supset succession\_event(s) \wedge organization(o))$
$\forall s, o1, o2(succession\_org(s, o1) \wedge succession\_org(s, o2) \supset o1 = o2)$
$\forall s, i(in\_and\_out(s, i) \supset succession\_event(s) \wedge in\_and\_out(i))$
$\forall s(succession\_event(s) \supset \exists i(in\_and\_out(s, i)))$
...

This different way of looking at templates will be useful when generation steps will be presented.

4.4 Conclusions

In designing an architecture for text generation —or for any relatively complex task— it is important that desired features are clearly formulated in advance. This section was indeed dedicated principally to illustrate what requirements were defined for this project. After a concise introduction to some terms and concepts from Natural Language Generation (NLG), seven desiderata were identified, some of them common to other generation tasks and some specific of generation from MUC templates. As it often happens when several goals are stated, these desiderata can conflict with each other, so that design turns into a search for the most satisfactory compromise. The way this compromise was pursued is the object of the next chapter.

A discussion of the specific characteristics of MUC templates as semantic formalism, as compared to input formalisms adopted by other generation systems, completed the chapter. MUC templates were considered both from a linguistic and from a logical perspective. A feature that sets MUC templates apart from most other input formalisms was pointed out: the presence of textual strings as slot fillers. MUC templates were eventually shown to be amenable to be interpreted as sets of positive literals on a signature obtained from template BNF definitions. Template definitions themselves can be seen as collections of constraints that must be satisfied by every legal template.

Chapter 5

The Architecture

The present chapter illustrates the main contribution of this thesis: an original architecture for generating text expressing the content of MUC templates. The desiderata stated in the previous chapter are met by means of a decomposition in processing tasks and linguistic resources that takes into account both the special nature of the input formalism and the availability of resources developed for Information Extraction purposes. The overall philosophy behind all practical decision is presented in the first part of the chapter, devoted to design principles. This discussion will be followed by a brief introduction to FASTUS architecture, necessary to understand the reason of some design choices. Section 5.3 presents the architecture by describing in detail:

- The decomposition of the system into components.
- A description of the knowledge sources accessed by each component.
- A description of the information exchanged among components.
- The thread of control among components.

Subsequent sections illustrate two extensions to the basic architecture, the first concerning additional processing required to ensure that the output will always be grammatical, and the second conceived to allow texts to be generated in a language different from that of the source texts. The description of the basic architecture and of its two extensions will be made more concrete by exemplifying from an implementation, the GeM prototype system.

5.1 Design principles

In designing a system for text generation from MUC templates, a first fundamental decision concerns the degree of decoupling one wants to achieve with respect to any specific IE system.

A first possibility consists in aiming at a complete independence from any IE system: MUC templates, after all, are the same for all MUC participants, and this independence would bring a higher generality to the architecture. The choice of a complete decoupling, however, would have a significant drawback: resource duplication. However good the designer is in keeping necessary linguistic resources modular and manageable in size, any nontrivial text generation system will require resources that, if not extensive, at least have to be developed by highly qualified individuals. If a significant fraction of these resources is domain-dependent, the price of flexibility with respect to IE systems is a much reduced portability to new applications, that is one of our desiderata. Work presented in [McKeown and Radev, 1995, Radev, 1997], goes in the direction of a complete decoupling. The architecture proposed therein is instantiated in the terrorist attack domain, and

avoiding development of costly domain-dependent resources is not set forth as an objective. Resource development is however partially relieved by adopting, for content realization purpose, the SURGE grammar, a broad-coverage systemic grammar of English implemented as a functional unification grammar in the FUF language.

An alternative to complete decoupling is tight coupling to a specific IE system. Despite the fact that IE and generation are very different tasks, some of the subtasks involved are quite similar. The most obvious example is probably parsing of textual strings occurring as slot fillers. Remember from Section 4.3 that there are cases in which parsing of such strings is unavoidable. Now, if the IE system already performs some kind of linguistic analysis, at least of source text segments containing information relevant to the template, it is likely that the result of such an analysis would be valuable to generation as well. Parsing results, and other partial results potentially useful to generation, however, are maintained internally by the IE system: the latter should be modified to spill them off in a suitable format. This is not always feasible, above all if the IE system was designed without this need in mind.

A step in the direction of facilitating the reuse of processors, resources and intermediate data is represented by some emerging proposals of architecture standardization. The idea is to establish rules on how modules interface with each other in a generic text processing system. A module compliant to these rules can then be plugged seamlessly into a system about which nothing was known to the module designers. Probably the most interesting proposal for such a generic architecture is the TIPSTER architecture[1] set forth by DARPA. GATE, designed and implemented at the University of Sheffield[2] [Gaizauskas *et al.*,], is an extended implementation of the TIPSTER architecture. Architectures of this kind, of course, are a compromise between the opposite necessities of posing strict constraints on interfaces to achieve modularity on one side and ensuring a sufficient communication bandwidth and not forcing an unacceptable overhead onto modules on the other. Moreover, as for every standard, to be interesting they should be reasonably stable and possibly widely accepted. Being implemented and freely available, GATE seems in a good position in this sense.

If an available IE system is not compliant with any generic architecture standard, and is anyway infeasible from a system engineering standpoint to have it share its internal results, there is a third alternative between no coupling and tight coupling: knowledge sources may be reused. Excluding those entirely based on statistical models, IE systems will need linguistic resources to operate. Such resources can have been written by hand, have been extracted from a corpus using machine learning algorithms, or result from a combination of the two. In any case, they will store a good deal of the knowledge on how relevant information, that is to be expressed again in textual form, is usually conveyed in text written by humans. This knowledge is likely to be useful, possibly in a different form, to the text generation system as well. A text generation system can thus be designed having in mind the particular set of linguistic resources developed for a specific IE system. This means:

- taking into account what resources have to be developed anyway when porting the IE system to a new application;
- taking into account how knowledge is divided into different repositories;
- assessing how declarative each resource is: a procedural resource probably requires more effort to reuse than a declarative resource;
- eventually, designing the architecture so as to minimize domain-specific resources that have to be designed expressly for the text generation system. This can be achieved by establishing a systematic correspondence between IE system resources and generation resources.

[1] See TIPSTER Home Page: http://www.tipster.org

[2] http://www.dcs.shef.ac.uk/research/groups/nlp/gate.

This alternative leads to a text generation system that is independent from any IE system from the point of view of operations: once it has been realized and ported to a specific application, it can be used in conjunction with any IE system that produces appropriate MUC templates. On the other hand, development time is reduced if compared to those of a generation system conceived independently from any existing IE system, as resources from the IE system that was chosen as a reference point are available. This is an important advantage, because development time is being addressed as a central issue in IE.

The proposed architecture was designed according to the third of the three alternatives. In terms of design decisions, this means that a specific IE system was chosen as a reference point, its knowledge sources were examined and their composition was taken into account in conceiving text generation resources.

Last editions of MUC conferences indicated that IE architectures based on cascades of finite-state transducers, or however on consecutive steps of shallow parsing, are particularly successful on the scenario template task [DARPA, 1995]. The author was kindly granted access to one such system, FASTUS, developed at the Artificial Intelligence Center of SRI International [Appelt *et al.*, 1995, Hobbs *et al.*, 1997]. As a brief description of its architecture will highlight, FASTUS was designed with a special attention for separating domain-dependent and domain-independent resources. Resources are all written by hand, but are organized in a way that allows for relatively short development time. The validity of the FASTUS approach is confirmed by the fact that other IE systems have been developed after it that imitate its internal organization [Grishman, 1995]. These features make FASTUS a suitable candidate to be taken as a reference point for a text generation system. Section 5.2 presents a brief introduction to FASTUS.

FASTUS is organized as a cascade of finite-state transducers (FST). Each FST is responsible for recognizing patterns more complex and abstract than those recognized by the FST that precedes it. The first FST takes raw text[3] as input, and the last generates an internal representation from which the template to be output can be constructed.

Now, FSTs are inherently bidirectional: they simply state a relation between two strings of characters in an alphabet (see [Roche and Schabes, 1997] for an introduction to the theory of Finite-State Transducers especially geared towards natural language processing). In principle, one could think of simply "reversing" transducers, inverting the roles of input and output. Although appealing, this approach is not practical for several reasons. First of all, though organized as an FST cascade at a macro level, FASTUS is not a cascade of FSTs in a strictly mathematical sense. During FST traversal, a feature-based data structure is manipulated in a way resembling the registers of an ATN [Woods, 1970]. Assignment actions performed on data structures coupled to each FST are procedural in nature, and are not well suited for inversion.

Were FASTUS really a cascade of FSTs in a strict mathematical sense, a second problem would make reversal impracticable: the loss of *sequentiality*. An FST is sequential if in any given state there is at most one outgoing edge with any given input label. As we will see in greater detail speaking about FASTUS architecture, FSTs are compiled from a set of patterns written by a system developer. Now, if these patterns define an unambiguous transduction, that is if they map any input string into at most a single output string, then it can be shown that a sequential device, called a *bimachine* can be built that is equivalent to it, in the sense that it emits the same output for every input string. A bimachine can in turn be transformed into the composition of two sequential FSTs whose output must be reversed after traversing each of them. A good deal of FASTUS efficiency derives from a compilation that leads to sequential FSTs, but the fact that these FSTs are sequential when used to go from raw text to more and more structured information does not say anything about their determinism when used in the other direction. Indeed, it turns

[3]Formatted according to a special SGML DTD as required to all MUC systems.

out that FSTs compiled for FASTUS are not sequential when used in the other direction. This non-sequentiality cannot be removed in any way because the inverse underlying transduction is not functional: it maps a same template to more than one string. This should not be a surprise, as it is obvious that there are many ways to say the same thing.

A third problem in reversing FASTUS FSTs relies in the fact that they were expressly designed with extraction in mind. This means, above all, that they overgenerate from a linguistic point of view. The most obvious example is that grammatical number and gender agreement is not enforced: this might not be a problem if analysis is the only concern, because other contextual information can ensure that, for instance, the main verb of a sentence will be correctly related to its subject, but it is a problem for generation because grammatical number must be known in order to produce a correct sentence.

Directly reversing FASTUS FSTs is thus an unpractical way of doing generation. The idea of reversing FASTUS computation, however, if infeasible at the "microscopic" level of FSTs, can still be a practical way if done at the "macroscopic" level of processing phases. In other words, if reversing any individual phase is unpractical, one can at least keep the same decomposition into phases and reverse the order phases are applied. If the same decomposition is kept, then it is more likely that it will be possible to reuse resources developed for FASTUS. On the other side, however, this choice turns into constraints on the order in which decisions concerning the produced text will be taken, and on the intermediate representations that will be manipulated throughout.

The following section provides an overview of FASTUS architecture.

5.2 FASTUS architecture

FASTUS [Hobbs *et al.*, 1997], performs five levels of processing:

1. Complex-word recognition, including multi-words and proper names;

2. Basic-phrase recognition, by means of which sentences are segmented into noun groups, verb groups and particles;

3. Complex-phrase analysis: Complex noun groups and complex verb groups are identified;

4. Domain-event analysis: The sequence of phrases produced at the previous level is scanned for patterns corresponding to events of interest to the application. When such patterns are found, an appropriate data structure is instantiated to encode information about entities and events;

5. Merging of structures: structures produced by the previous step are compared to one another, and structures found to be related to the same event are merged.

The first level of processing recognizes composite expressions such as "set up", "trading house" and proper names such as "Bridgestone Sports Taiwan Co.". Once recognized at this stage, these expressions will be treated as individual lexical tokens in the following processing. To perform this phase, FASTUS relies on a phrasal lexicon and a database of proper names.

The basic phrase recognition level is based on the observation that, though finite-state models are notoriously inadequate for full language parsing, they are well suited for identifying simple syntactic groups such as noun groups and verb groups. At the same time, also prepositions, conjunctions, relative pronouns and other words that will play some role in the following steps are recognized. An example of the way a sentence is segmented is the following:

[Bridgestone Sports Co.]$_{CompanyName}$ [said]$_{Verb\ Group}$ [Friday]$_{Noun\ Group}$ [it]$_{Noun\ Group}$ [had set up]$_{Verb\ Group}$ [a joint venture]$_{Noun\ Group}$ [in]$_{Preposition}$ [Taiwan]$_{Location}$ [with]$_{Preposition}$ [a local concern]$_{Noun\ Group}$ [and]$_{Conjunction}$ [a Japanese trading house]$_{Noun\ Group}$ [to produce]$_{Verb\ Group}$ [golf clubs]$_{Noun\ Group}$ [to be shipped]$_{Verb\ Group}$ [to]$_{Preposition}$ [Japan]$_{Location}$

The Finite-State transducer responsible for this phase is compiled from a set of patterns written by the system developer. These patterns rely on syntactic knowledge only, and are highly domain-independent.

The following phase, complex phrase recognition, detects those complex noun and verb phrases that can reliably be recognized on the basis of domain-independent syntactic information only. These include, for instance, the attachment of appositives, as in:

"The joint venture, Bridgestone Sports taiwan Co.,"

and noun group conjunctions, as in:

"a local concern and a Japanese trading house."

As basic and complex phrases are recognized, data structures for entities and events of interest are constructed.

The domain event recognition phase is the first phase in which domain-dependent knowledge comes into play. This knowledge is in the form of patterns designed to detect relations among domain entities that are of interest to the information extraction task. Patterns get compiled into a finite-state machine. Transitions in the machine are driven by pairs composed of the head word in a complex phrase and the phrase type. An example of one such pattern is:

<Company/ies: NounGroup> <set-up: VerbGroup> <Joint-Venture: NounGroup> <with: Preposition> <Company/ies: NounGroup>.

The instantiation of this pattern could lead to a structure like the following:

Relationship:	TIE-UP
Entities:	"Bridgestone Sports Co."
	"a local concern"
	"a Japanese trading house"
Joint Venture Company:	-
Activity:	-
Amount:	-

The last level of processing is not based on finite state machines. Structure merging is necessary to recognize when different instantiated structures actually refer to the same event or entity and should thus be conflated. At this stage the previous structure would be recognized as co-referring with:

Relationship:	TIE-UP
Entities:	-
Joint venture Company:	"Bridgestone Sports Taiwan Co."
Activity:	-
Amount:	NT$20000000

to yield:

Relationship:	TIE-UP
Entities:	"Bridgestone Sports Co."
	"a local concern"
	"a Japanese trading house"
Joint Venture Company:	"Bridgestone Sports Taiwan Co."
Activity:	-
Amount:	NT$20000000

Starting from the data structures thus obtained, a post-processing module outputs a file in the format required for MUC evaluations.

An interesting feature of the fourth phase, domain event recognition, is the way patterns are specified. Apart from being a simplification interesting in itself, it is relevant with regard to how resources will be organized in the text generator. Many subject-verb-object patterns are systematically related to each other. So, for instance, the sentence:

"General Motors manufactures cars"

holds the same semantic content as:

"Cars are manufactured by General Motors"
"...General Motors, which manufactures cars..."
"...cars, which are manufactured by General Motors..."
"...cars manufactured by General Motors..."
"General Motors is a car manufacturer"

and others. There is no reason the developer should specify all syntactic variations independently. A special purpose language, FASTSPEC, was designed to specify in a declarative way how to generate a set of patterns for the desired syntactic variations from that for the simple active form. This pattern generation is performed at compile time, so that efficiency is not affected.

Several of the resources developed for FASTUS are reused in the generation process:

1. The lexicon is imported almost as is, but had to be enriched with values for grammatical features such as agreement features.

2. FASTUS grammar for simple noun and verb groups was taken as a starting point to write the part of the generation grammar responsible for the same ranks.

3. FASTSPEC domain-independent description of syntactic variations was used as a starting point for developing the top part of the grammar, that is, the part concerned with whole sentence structure.

4. Domain-dependent patterns in the simple active form are used as an intermediate sentence representation between the purely semantic level represented by the template and a syntactically fully specified description of the sentence. The need for such an intermediate representation is advocated by many authors. The level of abstraction and dependence on linguistic knowledge of this intermediate representation will be discussed in more detail later.

5. Another knowledge source that does not come from FASTUS, but can be considered as given and was reused, is the BNF description of the template that is the target of the extraction task. This BNF description can easily be translated into the description of the database or the knowledge base that will contain the template in a format suitable for answering queries.

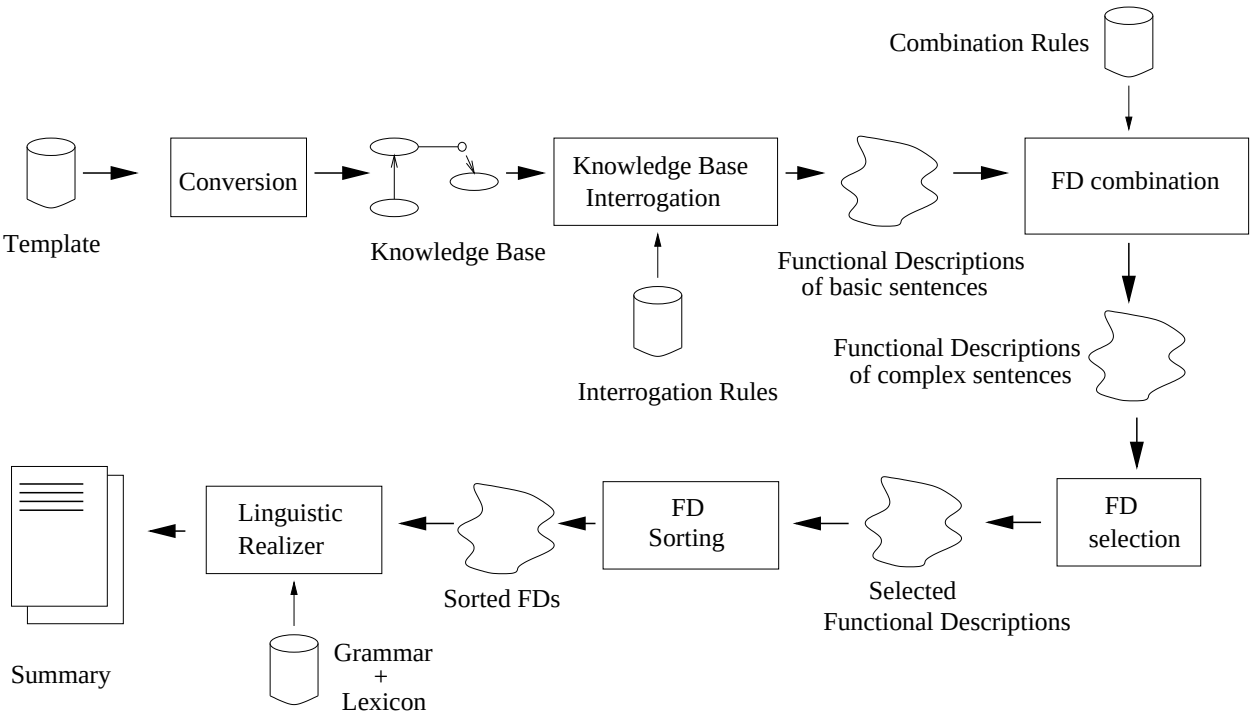


Figure 5.1: The architecture for generating texts from MUC templates. Boxes represent processing modules, cylinders stand for accessed resources, and irregular shapes represent intermediate products of the processing.

In the following sections the proposed architecture will be presented. At first I will introduce a basic architecture for monolingual text generation from MUC templates. This architecture has been fully implemented in Common Lisp and has been tested on a test set of 100 messages from the official MUC-6 evaluation. After the basic architecture will have been described, two extensions will be discussed. The first extension concerns parsing of the textual strings in the templates for inflection purposes. All the necessary components (the parser, the grammar for position names and the rules for changing morphological properties) have been implemented. The integration of these components into the basic architecture, however, was not part of the thesis project, so that they did not undergo a quantitative evaluation. The second extension, at last, concerns generation in a language different from that of the source text. As for the first extension, all modules were implemented, even if only with limited coverage, but they were not integrated and tested extensively.

5.3 Basic architecture

The basic architecture depicted in Figure 5.1 generates a short sequence of sentences from a MUC template.

The template to be expressed is initially contained in a file in the format required for MUC evaluation. A first step consists in converting it into a format suitable for populating a database or a knowledge base. The implemented version can use either LOOM [MacGregor and Bates, 1987] or CLASSIC [Resnick *et al.*, 1993], two knowledge base management systems based on description logics. This initial conversion is performed by a simple deterministic LALR transducer implemented with *bison* and *flex*, two public domain tools. The converted file is then loaded into a knowledge base, whose schema has been previously defined to reflect the structure of the domain as given by the template definition documents. It turns out that the description of the knowledge base schema can

be easily constructed by manipulating the BNF description of the template for a given scenario, at least if tools such as LOOM and CLASSIC are used. The next stage, indicated as *Knowledge Base interrogation*, consists in querying the knowledge base and building Functional Descriptions[4] (FDs hereafter) of *basic sentences*. These FDs are then combined into FDs of more complex sentences by iteratively applying a set of *combination rules*. Both the FDs for basic sentences and those for *complex sentences* emerging from the application of combination rules are redundant with respect to the knowledge base. This means that for each event in the knowledge base more than one FD is in general generated. It is thus necessary to select a subset of the FDs covering the knowledge base. Once this subset has been selected (in a *FD selection* phase), its elements can be sorted and sent to the *linguistic realizer*, that transforms FDs into actual sentences.

Each of these steps will now be described in more detail.

5.3.1 Preliminary conversion

In order to be accessed by subsequent processing modules, the template must be converted in a form suitable for being interrogated. As we will see, these queries are such that any relational database management system would suffice. For ease of integration, we experimented with two knowledge base management systems, CLASSIC and LOOM, that offer a broader range of computational services than conventional databases. Though both based on description logics, CLASSIC and LOOM follow two different philosophies. LOOM offers a very comprehensive set of primitives for concept definitions and knowledge base interrogation. This is in general achieved at the cost of a high computational load. CLASSIC, on the other side, offers only those services that can be implemented keeping computational effort reasonably low. Both systems provide services adequate to the text generation purpose, so CLASSIC was eventually preferred, even if it required queries to be stated somewhat more verbosely.

The conversion into a knowledge base description file is performed by means of a deterministic transducer implemented using two public domain tools, *flex* and *bison*. The time required for this conversion is negligible if compared to the other steps involved.

5.3.2 Knowledge Base interrogation

In the Knowledge Base Interrogation phase the knowledge base is consulted to figure out what "can be said" about it. This is done by means of *interrogation rules* similar to PC pairs described in [Jacobs, 1991]. The left-hand side of each interrogation rule is a query on the KB, whereas the right-hand side is a FD schema. If the query succeeds, the FD schema is instantiated with values coming from the answer and a FD for an actual basic sentence is produced.

Consider for instance the following query[5]:

```
(retrieve (?io ??subj ??obj ??pobj1)
  (:for-some (?se ?p ?o)
             (:and (succession_event ?se)
                   (post ?se ??pobj1)
                   (in_and_out ?se ?io)
                   (new_status ?io in)
                   (io_person ?io ?p)
                   (per_name ?p ??obj)
```

[4] *Functional Description* is a term imported from Functional Unification Grammars [Kay, 1979]. An FD is a possibly partial description of the linguistic properties of a sentence, encoded as a feature structure.

[5] LOOM queries are somewhat more readable than the corresponding CLASSIC queries, so we present LOOM queries for the sake of clarity.

(succession_org ?se ?o)
(org_name ?o ??subj))))

This query searches the knowledge base for a SUCCESSION_EVENT object (first conjunct in the :and clause) such that the NEW_STATUS of the corresponding IN_AND_OUT object has IN as its value (second and third conjunct), meaning that the information is about a person entering a certain position, and not leaving it. For such a SUCCESSION_EVENT it returns the name of the organization involved (contained in the ??subj variable), the name of the person (??obj) and the position (??pobj1). The IN_AND_OUT event checked by the query is also retrieved (?io) as it will be used later, in the cover selection stage.

If the query succeeds then the following FD schema is instantiated:

$$\begin{bmatrix} & \begin{bmatrix} ??subj & v(??subj) \\ ??head & appoint-word \\ ??obj & v(??obj) \\ ??prep1 & as \\ ??pobj1 & v(??pobj1) \end{bmatrix} \\ args & \end{bmatrix} \tag{5.1}$$

where $v(??x)$ stands for the value associated with the variable ??x by the query. This FD corresponds to a sentence having a main verb suitable for expressing an appointment, such as "to appoint" or "to name", and arguments provided by the variables instantiated by the query (e.g.: "$v(??subj)$ named $v(??obj)$ as $v(??pobj1)$").

At this point of processing, then, a class of verbs characterized by a specific semantic and subcategorization frame is selected, but the superficial syntactic structure of the sentence is still undetermined. A set of syntactic variants in which the FD can be realized is also stored.

A significant amount of time may be necessary to develop interrogation rules. In the design of this library, however, the availability of resources developed for FASTUS can save a good deal of effort. In the implementation of the architecture, interrogation rules are designed starting from FASTUS domain-dependent patterns. Such patterns identify a class of verbs, and map events participants and roles onto verb deep arguments. FDs for basic sentences can thus be written "isomorphic" to these patterns: in principle they could be obtained by means of an appropriate compiler. The FD for appointments above, for example, was taken from the FASTUS domain-dependent pattern in Fig.5.2. These patterns have the form:

Instantiate
<list-of-variants>
Rules By
<constraints-on-variables>

The list of variants expresses all different superficial forms the pattern can occur in. Constraints on variables are imposed in the second part of the rule. In the example, the $??subj$ variable is constrained to being bound to something on which either the predicate *company*? or the predicate *Company−Agent*? is true, variable $??head$ can be bound only to an *appoint−word* and so on. Note that constraints on variables that, in generation, are instantiated with material taken from the knowledge base can safely be ignored when writing the FD. Constraints on tokens that will come from the lexicon, such as the verb, are kept and will be used for lexical selection.

Even if syntactic variants do not appear within the FD schemas taken from FASTUS patterns, they are stored in an appropriate data structure for further use. The corresponding query that makes for the left-hand side of the interrogation rule, however, cannot be written automatically not even in principle. The problem, in this case, is that FASTUS internal semantic representation does

```
Instantiate
        ActiveBase, ActiveBaseDitrans, ActiveGerund,
        ActiveInfinitive, ActiveRelativeSubj, ActiveRelativeObj,
        PassiveBase, PassiveBaseDitrans, PassiveGerund,
        PassiveInfinitive, PassiveRelativeSubj
    Rules By
    ??subj = company?=T | Company-Agent?=T &&
    ??head = Appoint-word &&
    ??obj = Person?=T &&
    ??prep1 = "as" | "to" | "to be" &&
    ??pobj1 = Position?=T &&
    ??semantics = [...] ;;
```

Figure 5.2: A simplified FASTUS domain-dependent pattern.

not match exactly the MUC template: it is a structure better suited for merging information on the same event distributed across the source text. In FASTUS, this internal semantic representation is in turn translated into the actual MUC template. None of:

1. the mapping from sentence patterns onto the internal semantic representation;
2. the merging of information on the same event, and
3. the final mapping onto the MUC template

are performed, in FASTUS, relying on fully declarative resources. Instead of trying to reverse the behaviour of a procedural component, handwriting the relatively few queries sounded a more sensible choice. Furthermore, once the FD structure is known, as it is because it is the same as that of the corresponding FASTUS domain-dependent pattern, writing the query is straighforward.

From an abstract logical point of view, interrogation rules are implications having as antecedent a formula on literals from the template and as consequent a formula stating linguistic properties of a sentence. As an example, the previous query rule could be translated as:

$$\forall ?se, ?p, ?o, ??pobj1, ?io, ??obj, ??subj \quad \left(\begin{array}{l} succession\_event(?se) \wedge \\ post(?se, ??pobj1) \wedge \\ in\_and\_out(?se, ?io) \wedge \\ new\_status(?io, in) \wedge \\ io\_person(?io, ?p) \wedge \\ per\_name(?p, ??obj) \wedge \\ succession\_org(?se, ?o) \wedge \\ org\_name(?o, ??subj) \end{array} \right) \supset \quad \exists f1, f2 \left(\begin{array}{l} fd(f1) \wedge \\ fs(f1) \wedge \\ args(f1, f2) \wedge \\ fs(f2) \wedge \\ ??subj(f2, v(??subj)) \wedge \\ ??head(f2, appoint{-}word) \wedge \\ ??obj(f2, v(??obj)) \wedge \\ ??prep1(f2, as) \wedge \\ ??pobj1(f2, v(??pobj1)) \end{array} \right)$$

where fd is a unary predicate verified by sentence functional descriptions and fs is a unary predicate verified by any feature structure. Basic sentence construction corresponds then to computing the closure of the set of literals from the template under such implications.

A functional description f *covers*, or *expresses*, a content $S(f)$, where $S(f)$ is a conjunction of literals, if f was obtained by applying an interrogation rule whose left-hand side matched $S(f)$ in the template representation.

5.3.3 Sentence combination

Sentences produced up to this point are very simple: namely they are made of a single main clause with its arguments and possibly some adjuncts.[6] A summary composed uniquely of sentences of this kind would be of poor quality, since the entities involved would be repeated many times in an unnatural way. For instance, one could generate something like:

- Barry Diller was appointed chief executive officer of QVC Network Inc.
- Barry Diller was appointed chairman of QVC Network Inc.
- Joseph M. Segel resigned as chairman of QVC Network Inc.
- Joseph M. Segel resigned as chief executive officer of QVC Network Inc.
- ...

A much more readable summary would contain instead a single sentence such as:

- Barry Diller replaced Joseph M.Segel as chairman and chief executive officer of QVC Network Inc.
- ...

More complex and concise sentences of this kind could in principle be generated using the same interrogation rule approach adopted for basic sentences. The problem is that the number of different queries and FD schemas that one should explicitly take into account would grow enormously. A much more sensible alternative consists then in generating only the basic sentences in the knowledge base interrogation stage, and then combining their FDs by means of appropriate rules.

The use of combination rules for generation purposes is not new. [Robin, 1994] presents a study of the rules used in the domain of newspaper articles on basketball games and on stock market. As Robin's primary interest is on integrating contextual information, he focuses on *content adding* rules, that is, on rules conceived for enriching a summary draft, made from what he calls *fixed concepts*, with background knowledge. Nevertheless he considers also a class of *side aggregation rules* aiming at increasing summary readability and coherence. Such side aggregation rules, however, are different from what is needed in our context in that they are explicitly conceived to correct problems created by the application of content adding rules.

The experimental system YH performs sentence combination to improve fluency. As it is presented in [Gabriel, 1991], YH takes an abstract description of a Lisp program for the Dutch National Flag game and generates a natural language description. As the input is always the same, YH is intended to show how different output may be obtained by modifying parameters. Interesting transformations are applied on a draft to improve its quality, including ellipsis and passivization. The overall generation process, however, relies on deep semantic knowledge that can reasonably be gathered only for toy domains such as the game used for experiments.

If the task is that of generating text from MUC templates then combination rules will tend to be *content-preserving* rather than *content-adding*: there is no clear distinction between an obligatory

[6] The distinction between arguments and adjuncts for a verb is roughly that arguments are compulsory and in a fixed configuration, while adjuncts are optional and can come in any order.

content specification and a supplementary content specification as in Robin's STREAK, because in this case all of the template must be expressed.

Some of the functions combination rules can perform in our setting are:

1. Factorization of constituents through conjunction;
2. Embedding of sentences as subordinate clauses;
3. Insertion of cue phrases;
4. Pronominalization and use of referential expressions different from the proper name of an entity.

We will consider these functions in turn.

Factorization

A factorization can be performed whenever there are two sentences sharing some constituents. In such a circumstance, a new sentence can be generated by conjoining the different constituents. For instance, from the (FDs for the) sentences:

- Barry Diller was appointed chief executive officer of QVC Network Inc.
- Barry Diller was appointed chairman of QVC Network Inc.

the (FD for the) sentence:

- Barry Diller was appointed chairman and chief executive officer of QVC Network Inc.

can be produced. The example above refers to the insertion of a conjunction in the argument for the company position, but of course the same idea can be applied to different grammatical roles.

Subordinate clauses

Sentences may be embedded as relative clauses into other sentences to improve readability and conciseness. This can be done when at least one constituent is shared between the two original sentences. For instance, from:

- Barry Diller was appointed chairman and chief executive officer of QVC Network Inc.;
- Barry Diller resigned as chief executive of Twentieth Century Fox Film Corp.

it is possible to form:

- Barry Diller, who was chief executive of Twentieth Century Fox Film Corp., was appointed chairman and chief executive officer of QVC Network Inc.

Barry Diller is an entity mentioned in both sentences and can thus be factored. Relative clauses have an immediate effect in avoiding the repetition of the same entity, but other subordinate clauses achieve the same result. For instance, one could embed a sentence referring to somebody entering a position as a purpose clause in a sentence reporting of the same person leaving another position, as in:

> Barry Diller resigned as chief executive of Twentieth Century Fox Film Corp. *to become* chairman and chief executive officer of QVC Network Inc.

Subordinate clause embedding must be performed with some care, however: the choice of a particular type of clause conveys information about a relation between the event reported in the main clause and that reported in the subordinate clause. If the semantic representation does not contain enough clues to infer such a relation, then the embedding should not take place to prevent from generating a misleading text (thus decreasing fidelity).

Cue phrases

In some situations adding a redundant word, or phrase, can be of help in making the text sound more natural. Consider for instance the two sentences:

- Barry Diller, who was chief executive of Twentieth Century Fox Film Corp., was appointed chairman and chief executive officer of QVC Network Inc.;
- Barry Diller resigned as Coca Cola Co. chairman and chief executive.

The addition of the adverb "also" in the appropriate position in the second sentence improves the overall coherence of the text:

- Barry Diller, who was chief executive of Twentieth Century Fox Film Corp., was appointed chairman and chief executive officer of QVC Network Inc.;
- Barry Diller *also* resigned as Coca Cola Co. chairman and chief executive.

Referring expressions

In all the previous examples entities were referred to explicitly: by means of a proper name in the case of companies and persons, and by the position name in the case of the position involved. In real texts, however, after an entity has been introduced, it tends to be referred to also using different kinds of referring expressions. One case is that of pronouns. Instead of:

- Barry Diller, who was chief executive of Twentieth Century Fox Film Corp., was appointed chairman and chief executive officer of QVC Network Inc.;
- Barry Diller also resigned as Coca Cola Co. chairman and chief executive.

one could better generate:

- Barry Diller, who was chief executive of Twentieth Century Fox Film Corp., was appointed chairman and chief executive officer of QVC Network Inc.;
- *He* also resigned as Coca Cola Co. chairman and chief executive.

Another alternative consists in using a definite reference, so that the second sentence could be:

- He also resigned *from the same positions* at Coca Cola Co.

Factorization, subordinate clause introduction, cue phrase and variation in referring expressions are manipulations that can greatly improve effectiveness and readability.

An important issue before presenting how such rules have been or could be implemented concerns the characteristics of the representation combination rules act on. The next section is devoted to this issue.

Intermediate sentence representation

Text generation can be seen as a mapping from an initial semantic specification to a well formed text in a natural language. This mapping can be direct, in the sense that it can not involve any intermediate passage: this is the case of "canned text" systems that, when determinate situations occur, produce a message from a set of messages previously stored in a repository. Actually, most of the cases in which natural language is used to communicate with the user of a software package fall into this category. This, however, is a somehow degenerate case from the point of view of natural language generation. NLG, indeed, can pragmatically be seen as the study of the methods that can be deployed in cases where the canned text approach is infeasible because the number of messages that should be stored in advance grows combinatorially with some input parameter.

In cases when true text generation is required, then, the mapping from the initial content specification to the final text is the composition of two or more partial transformations, each going from or to some intermediate representation. Partial transformations are generally related to the decomposition of text generation into subtasks that was introduced in the Chapter 4, so intermediate representations can be more or less specified from the point of view of the linguistic information that must be present (and the decisions that must be taken) to completely determine the text that will be output. An important choice that must be done when implementing combination rules involves the nature of the representation combination rules will act on: which decision concerning the linguistic properties of the sentences should have been already taken when combination takes place?

The problem in this case is that different combination operations would be better performed on intermediate representations at different levels of abstraction. Factorization of constituents through conjunction, for example, can be performed on a fairly unspecified representation: it is perfectly sufficient to know a generic class of head verbs and how entities are mapped onto deep cases. Considering again the example in Section 5.3.3, there is no need to know whether "to name" or "to appoint" will be chosen to express the fact that a person has entered a position, nor that the person entering the position will be realized as the grammatical subject of the sentence because the active form was chosen. To decide for the pronominalization of a constituent, however, both the order in which sentences will appear (to find which of two mentions of an entity can be pronominalized and which cannot) and the exact grammatical role must be known (to choose the case of the pronoun).

If a very abstract representation is chosen, then manipulations requiring a more detailed knowledge of the sentence syntax will not be possible. If a representation very close to the final text is preferred, on the other side, more abstract transformations will be complicated as an unnecessary variety of cases will have to be accommodated for.

Robin's STREAK follows an interesting approach to this problem. All along the generation of the paragraph that forms the basketball match statistics summary, a three-layered representation is used. For the text, the following representations at different level of linguistic commitment are maintained:

1. a *Deep Semantic Specification (DSS)*, that is a flat semantic network encoding the initial content specification. It is a purely conceptual semantic representation, independent of any linguistic consideration;

2. a *Surface Semantic Specification (SSS)*, that is a semantic tree capturing content organization. It specifies how semantic elements are to be grouped together inside linguistic constituents and which element should head the group. It does not specify, however, syntactic categories for constituents nor specific lexical items. Whereas DSS is domain-specific and language independent, SSS is domain-independent and linguistically motivated;

3. a *Deep Grammatical Specification (DGS)*, that is a lexicalized syntactic tree in which each

node corresponds to a syntactic constituent of the sentence and specifies its category, lexical head and syntactic features.

When a combination rule is applied, information from all three levels can be taken into account. This representation, however, requires that the three levels are kept aligned as sentence combination goes on. The complexity of the representation, coupled with the fact that, though non-monotonic, combination rules are implemented using a typically monotonic computational device such as graph unification, is probably the cause of the low efficiency of STREAK, which, according to its designer [Robin, 1994], can take more than two hours on a Sparc 10 workstation to produce a summary.

The solution I adopted is a pragmatic compromise between flexibility and efficiency. Combination rules operate on a linguistically fairly abstract level: that of the FDs produced in the knowledge base interrogation stage. At this level, semantic objects are grouped in constituents, but the syntactic type of constituents is not specified yet. Furthermore, the syntactic variant of the sentence is still to be determined. At this level of abstraction, transformation may be specified in a reasonably simple way and, most important, can be applied efficiently. In order to perform also transformations involving a more detailed level of syntactic specificity, however, rules can pose constraints on syntax that must be satisfied by the following processing stages. As an example, consider the case of a combination rule for inserting the cue-word "also". Consider again the example in Section 5.3.3, that we repeat here for the reader's convenience:

Source 1: "Barry Diller, who was chief executive of Twentieth Century Fox Film Corp., was appointed chairman and chief executive officer of QVC Network Inc.;"
Source 2: "Barry Diller resigned as Coca Cola Co. chairman and chief executive"
Target 2: "Barry Diller *also* resigned as Coca Cola Co. chairman and chief executive"

In order to produce a fluent and non misleading text it is necessary to specify that the second sentence will have to be realized in the basic active form (as it is in the example) and not, say, with the company name topicalized, as in:

Target 2': "At Coca Cola Co., Barry Diller *also* resigned as chairman and chief executive."

Were this alternative syntactic variant selected, in fact, the reader would be misled to infer that there were other events involving Barry Diller at Coca Cola Co.

Constraints of this kind, involving a single sentence, can be easily introduced as appropriate feature values by the designer, without running into the risk of overconstraining a sentence to the point where it cannot be realized. Note however that it can be necessary to pose constraints involving more than one sentence at a time. In the previous case, for instance, the additional constraint that sentence 1 must be realized before sentence 2 needs be enforced. If several transformations implying inter-sentence constraints take place, the overall set of constraints could end up being unsatisfiable. As will become clearer after combination rule control will have been explained, however, in the current implementation unsatisfiable constraints are not a problem, because the implemented combination rules impose only constraints on single sentences. How combination rules imposing inter-sentential constraints can be dealt with will be discussed in the section on combination rules implementation.

A possible alternative to choosing a single level of abstraction for performing sentence combination consists in performing several steps of sentence combination. At each step, an appropriate subset of combination rules is called into play, namely those requiring the level of linguistic specification that has been attained at the current stage of processing. In this way, combination rules for factorization through conjunction, for example, would be applied on FDs as they come out after

knowledge base interrogation, whereas combination rules concerning cue phrase insertion would be applied only later, after sentence order has been determined. The distribution of combination steps must be studied carefully, but it could turn out not to be detrimental to efficiency, especially if the same machinery can be used in all cases and the overhead for applying rules in more than one occasion is low.

5.3.4 Combination rule implementation

Objects manipulated by combination rules are functional descriptions of sentences. FUF provides an appropriate formalism for manipulating such objects. Rules themselves may be conceived as feature structures with a number of *socket-features* in which FDs to be aggregated can be plugged. If appropriate constraints are satisfied and the plugging succeeds, then the FD specified as value of a special *result-feature* is fully instantiated and can be extracted for further use, otherwise unification fails and nothing happens. For those familiar with PROLOG, the result feature plays the role of the head in a PROLOG clause, whereas socket features form the body.[7]

An example of combination rule, namely one for introducing conjunctions in the subject position, is shown in Figure 5.3. The first feature-value pair serves only for rule identification. Feature "arg0" is what we called above the result-feature: the subgraph under it will be the result of the combination operation. Constraints are expressed in terms of coreference links to nodes under features "arg1" and "arg2": the socket-features. So, for instance, we can see that a condition for the combination to succeed is that none of the subjects of the original FDs is already a conjunction. If this and the other conditions are satisfied, then the resulting FD will be a conjunction, its grammatical number will be "plural", and the resulting subject will be the conjunction of the two original subjects. Other constraints require that all the other constituents in the original sentences be unifiable[8].

Combination rules have been implemented for some of the functions listed in the previous section. In particular, rules for factoring constituents through conjunction and for embedding relative clauses have been written and tested. A rule of the first kind was shown in figure 5.3. A simplified version of a rule for relative clause embedding is presented in Figure 5.4

Combination rules for cue phrases and referring expressions alternative to proper names have not been implemented, yet. Note that these rules are somewhat more problematic since they impose constraints on the relative positions of sentences: a pronoun can occur only if the sentence containing its intended referent precedes it, and the same is true for a sentence in which a reference is marked as a repetition by means of a cue phrase such as "also". There are several ways of dealing with this problem:

- the first consists in allowing FDs representing more than one sentence. In this way one can stick together sentences that must be in a given relative position. The problem in doing so is that FDs for sentence sequences are definitely more complicated to manipulate if further combination is to be performed. Note however that this can be avoided if combination rules are divided into clusters and sequence-forming rules are applied last. If multi-sentence FDs are allowed, the realization grammar must be extended to deal with sequences of sentences and not only with individual sentences;

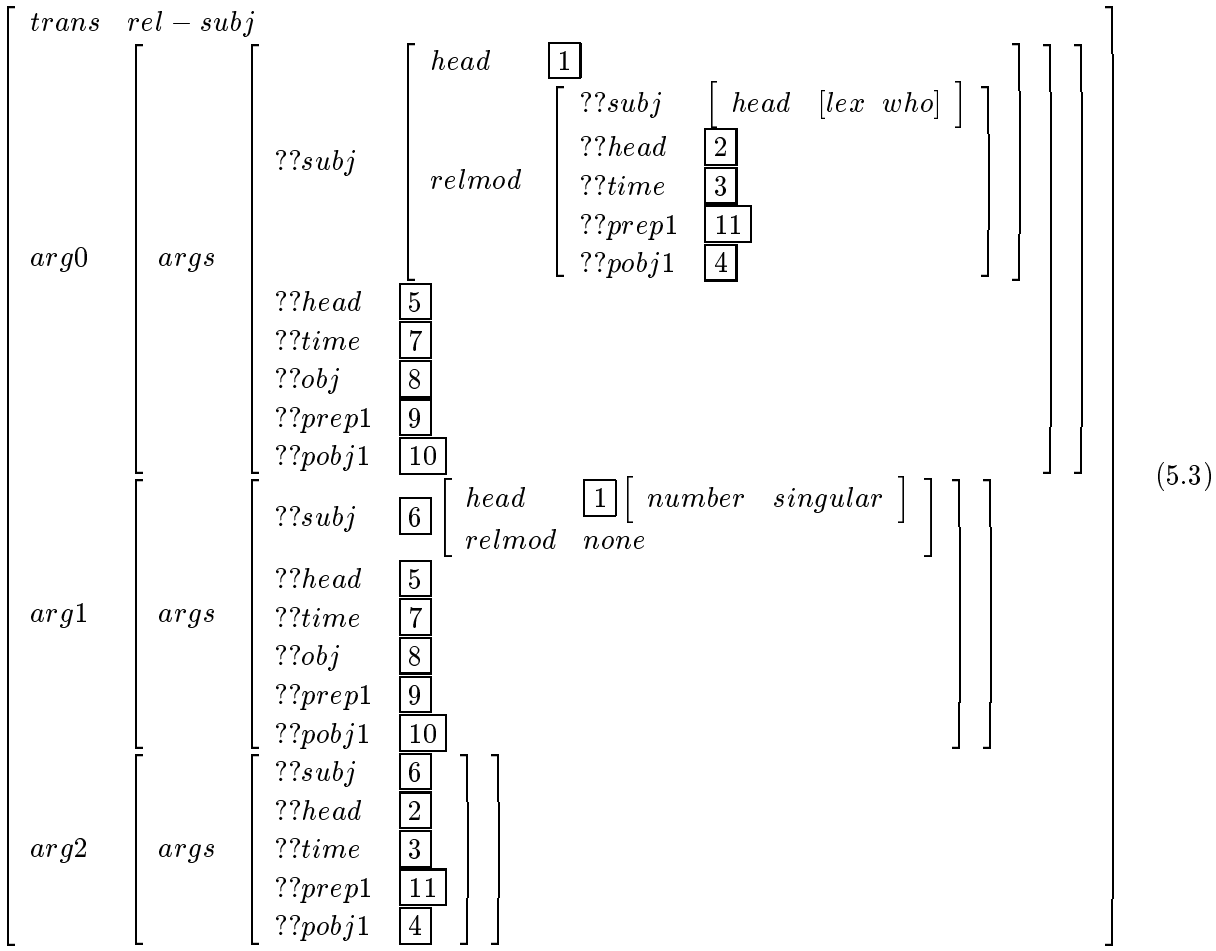
- a second way consists in attaching additional constraints to FDs formed using these combination rules. To an FD created by a combination rule for pronominalization, for example,

[7] The term *head feature* would have been better reminiscent of Horn clauses, but it was avoided because of ambiguity with features describing the head of a constituent.

[8] The combination rule in Figure 5.3 is simplified for the sake of clarity. In the actual rule additional constraints are posed to deal with the possible presence of conjunctions and modifiers in the constituents to be unified.

$$
\left[\begin{array}{lll}
transformation & conj-subj & \\
arg0 & args & \left[\begin{array}{ll}
??subj & \boxed{1}\left[\begin{array}{ll} head & \left[\begin{array}{ll} conjunction & + \\ number & plural \\ first-conj & \boxed{1} \\ second-conj & \boxed{2} \end{array}\right] \end{array}\right] \\
??head & \boxed{3} \\
??obj & \boxed{4} \\
??prep1 & \boxed{5} \\
??pobj1 & \boxed{6} \\
??prep2 & \boxed{7} \\
??pobj2 & \boxed{8} \\
??prep3 & \boxed{9} \\
??pobj3 & \boxed{10}
\end{array}\right] \\
arg1 & args & \left[\begin{array}{ll}
??subj & \boxed{1}\left[\begin{array}{ll} head & \left[\begin{array}{ll} conjunction & - \end{array}\right] \end{array}\right] \\
??head & \boxed{3} \\
??obj & \boxed{4} \\
??prep1 & \boxed{5} \\
??pobj1 & \boxed{6} \\
??prep2 & \boxed{7} \\
??pobj2 & \boxed{8} \\
??prep3 & \boxed{9} \\
??pobj3 & \boxed{10}
\end{array}\right] \\
arg2 & args & \left[\begin{array}{ll}
??subj & \boxed{2}\left[\begin{array}{ll} head & \left[\begin{array}{ll} conjunction & - \end{array}\right] \end{array}\right] \\
??head & \boxed{3} \\
??obj & \boxed{4} \\
??prep1 & \boxed{5} \\
??pobj1 & \boxed{6} \\
??prep2 & \boxed{7} \\
??pobj2 & \boxed{8} \\
??prep3 & \boxed{9} \\
??pobj3 & \boxed{10}
\end{array}\right]
\end{array}\right] \qquad (5.2)
$$

Figure 5.3: A combination rule for conjunction introduction in subject position.



(5.3)

Figure 5.4: A combination rule for relative clause embedding.

one could attach a constraint stating that it can be chosen only in conjunction with the FD containing the antecedent, and must be placed immediately after it. This would achieve the same effect of the approach above, but avoiding the necessity of dealing with FDs for sequences of sentences. This approach, however, has the drawback that all these constraints must be resolved when FDs are selected for generation: as only a subset of the FDs will be selected, it will be necessary to check that the chosen subset of FDs does not have, attached, an unsatisfiable set of constraints;

- a third alternative consists in applying order-dependent combination rules later, only after FDs have been selected and sentence order has been decided. This would be the simplest solution, but note that it could prevent some possibly effective combination from taking place.

It is of course possible to give a logical interpretation also to combination rules. As for interrogation rules, a combination rule corresponds to an implication. In the case of combination rules, however, both the antecedent and the consequent are formulas over literals stating linguistic properties of sentences. As an example, the combination rule for relative clause embedding can be expressed as:

$$\forall fd_1, fs_{11}, fs_{111}, fs_{1111}, fd_2, fs_{21}, co_2, co_3, co_4, co_5, co_7, co_8, co_9, co_{10}, co_{11}$$

$$\left(\begin{array}{l} fs(fd_1)\wedge \\ fs(fd_2)\wedge \\ fd(fd_1)\wedge \\ fd(fd_2)\wedge \\ fs(fd_1)\wedge \\ fs(fd_2)\wedge \\ fs(fs_{11})\wedge \\ fs(fs_{111})\wedge \\ fs(fs_{1111})\wedge \\ fs(fs_{21})\wedge \\ args(fd_1, fs_{11})\wedge \\ ??subj(fs_{11}, fs_{111})\wedge \\ head(fs_{111}, fs_{1111})\wedge \\ number(fs_{1111}, singular)\wedge \\ relmod(fs_{111}, none)\wedge \\ ??head(fs_{11}, co_5)\wedge \\ ??time(fs_{11}, co_7)\wedge \\ ??obj(fs_{11}, co_8)\wedge \\ ??prep1(fs_{11}, co_9)\wedge \\ ??pobj1(fs_{11}, co_{10})\wedge \\ args(fd_2, fs_{21})\wedge \\ ??subj(fs_{21}, fs_{111})\wedge \\ ??head(fs_{21}, co_2)\wedge \\ ??time(fs_{21}, co_3)\wedge \\ ??prep1(fs_{21}, co_{11})\wedge \\ ??pobj1(fs_{21}, co_4) \end{array}\right)$$

$$\supset$$

$$\exists fd_0, fs_{01}, fs_{011}, fs_{0112}, fs_{01121}, fs_{011211}$$

$$\left(\begin{array}{l} fs(fd_0)\wedge \\ fd(fd_0)\wedge \\ fs(fs_{01})\wedge \\ fs(fs_{011})\wedge \\ fs(fs_{0112})\wedge \\ fs(fs_{01121})\wedge \\ fs(fs_{011211})\wedge \\ args(fd_0, fs_{01})\wedge \\ ??subj(fs_{01}, fs_{011})\wedge \\ head(fs_{011}, fs_{1111})\wedge \\ relmod(fs_{011}, fs_{0112})\wedge \\ ??subj(fs_{0112}, fs_{01121})\wedge \\ head(fs_{01121}, fs_{011211})\wedge \\ lex(fs_{011211}, who)\wedge \\ ??head(fs_{0112}, co_2)\wedge \\ ??time(fs_{0112}, co_3)\wedge \\ ??prep1(fs_{0112}, co_{11})\wedge \\ ??pobj1(fs_{0112}, co_4)\wedge \\ ??head(fs_{01}, co_5)\wedge \\ ??time(fs_{01}, co_7)\wedge \\ ??obj(fs_{01}, co_8)\wedge \\ ??prep1(fs_{01}, co_9)\wedge \\ ??pobj1(fs_{01}, co_{10}) \end{array}\right)$$

The notion of content covered, or expressed, by an FD, can easily be extended from basic sentences to sentences obtained through sentence combination. The content covered by the aggregated

sentence is the union of the content of the sentences it comes from.

Combination rule control

Combination rules are applied exhaustively in the proposed architecture: new FDs are added until it is no longer possible to apply any rule, using neither the original basic FDs nor the FDs created by means of previous rule application. This policy is motivated by the following reason: the main idea behind sentence combination, at least in the case of factorization and clause embedding, is that a reduced number of relatively complex sentences is a better way of conveying content than a big number of elementary sentences. Paying the appropriate care in not exceeding a complexity threshold that would make sentences difficult to understand, it is thus important that combination rules are applied as much as possible. Note however that the same elementary sentence can in general be combined with several others depending on which combination rule is chosen. Moreover, this is true not only of elementary sentences, but also of complex sentences resulting themselves from the application of combination rules. Any strategy that didn't consider some possible combination of sentences would be incomplete, in the sense that it would not ensure that the most readable and concise sentences were generated.

Control strategies alternative to exhaustive application will be discussed in the following section, after the issue of cover selection has been presented.

5.3.5 FD selection

The set of FDs coming out from sentence combination is strongly redundant, in the sense that several FDs overlap as for the content they express. As in the final summary repetitions must be avoided, a subset of non-overlapping FDs covering as much as possible from the initial content specification must be selected. Not only the FDs must cover as much as possible of the template content, but they should also be the best from a linguistic point of view, in the sense that, once realized, they should lead to a fluent and concise summary.

To make this selection, a scoring schema for FDs has been conceived. The "desiderata" affected by the selection are completeness, conciseness and readability. Completeness is affected because the extent to which the template content is expressed depends of course on the subset of FDs that is chosen. Conciseness is an issue as well, because the same content may be expressed in a different amount of space by different FDs. If FDs with overlapping content are selected, finally, the text will contain repetitions detrimental to readability. Too long sentences, also, can be difficult to read and, if combination rules for cue phrases and for referring expressions other than proper names are considered, sentences created by such combination rules will (hopefully) be more readable.

To compare FDs in terms of their contribution to completeness, conciseness and readability, auxiliary information is attached to them. This information includes:

- for each FD f, the set $S(f)$ of literals from the initial content specification that are expressed by the sentence[9];
- for each FD f, the number $C(f)$ of constituents (simple noun groups and verb groups) composing the FD;
- a relation $P \subset F \times F$, where F is the set of all the FDs available after sentence combination, such that $P(f_1, f_2)$ is true if and only if f_2 was obtained by f_1 (and possibly some other FD) by means of an combination rule. We call P^* the transitive closure of P.

[9] Actually only literals from binary predicates, and only some of them, are considered for reasons that will be explained later.

To avoid generating too complex sentences a threshold on the number of constituents may be set as a system parameter: FDs with more constituents than the threshold can be simply ignored.

Given that completeness and conciseness are properties to pursue, we can define an optimal selection of FDs as follows:

> Let L be the set of literals in the conjunction representing the template content, and F the set of functional descriptions available after sentence combination. A *selection* is a subset $O \subseteq F$ such that $\forall f_1, f_2 \in O, S(f_1) \cap S(f_2) = \emptyset$. A selection O is *optimal* if there is no other selection O' such that:
>
> - $\bigcup_{f \in O'} S(f) \supset \bigcup_{f \in O} S(f)$, or,
> - $\bigcup_{f \in O'} S(f) = \bigcup_{f \in O} S(f)$ and $\sum_{f \in O} C(f) > \sum_{f \in O'} C(f)$, or,
> - $\bigcup_{f \in O'} S(f) = \bigcup_{f \in O} S(f)$ and $\sum_{f \in O} C(f) = \sum_{f \in O'} C(f)$ and $\exists f' \in O', f \in O$ such that $S(f) = S(f')$ and $P^*(f, f')$

In other terms, a selection is a set of FDs covering mutually disjoint sets of literals, whereas an optimal selection is a selection such that there is no other selection covering a superset of its literals, or covering the same set of literals but with fewer constituents, or anyway including an FD obtained from a corresponding FD in the chosen selection by means of sentence combination.

The problem of finding an optimal selection is intractable, as there is an easy reduction from another NP-hard problem, the *set packing* problem (see [Garey and Johnson, 1979] for a proof of the intractability of the set packing problem), to it. Let S be a finite set, let C be a collection of subsets of S, and let k be a positive integer $k \leq |C|$. The set packing problem consists in answering the question:

> Does C contain at least k mutually disjoint sets?

Of course this is equivalent to asking:

> What is the largest number of mutually disjoint sets in C?

because if we can answer the latter question we can surely answer the former simply comparing the maximum number with k, and if we can answer the former we can start with $k = |C|$ and then decrease k until we obtain a positive answer, so that we can find the maximum number k of mutually disjoint sets.

Now, given an instance of the set packing problem, it is easy to build an instance of the optimal selection problem such that providing an answer to the latter is equivalent to providing an answer to the former (Figure 5.5). Let $C = \{c_1, \ldots, c_n\}$, and let $m = \max_{1 \leq i \leq n} |c_i|$. We can add to each c_i new elements from outside S so that every c_i has exactly the same cardinality: $B = \{b_1, \ldots, b_n\}$, where $b_i = c_i \cup \{a_{i,1}, \ldots, a_{i,m-|c_i|}\}, i = 1, \ldots, n$. Now, if the newly added elements are all distinct, that is if $\forall i < j, 1 \leq i, j \leq n, \{a_{i,1}, \ldots, a_{i,m-|c_i|}\} \cap \{a_{j,1}, \ldots, a_{j,m-|c_j|}\} = \emptyset$, then two elements of B are disjoint if and only if the corresponding elements of C are disjoint. Let the weight of each b_i, corresponding to the number of constituents in an FD, be uniform, so that it does not interfere in the selection, and let $P^* = \emptyset$ for the same reason: the optimal selection problem boils down to finding the subcollection of mutually disjoint sets in B covering the maximum number of elements of $S \cup A$, where $A = \cup_{1 \leq i \leq n} \{a_{i,1}, \ldots a_{i,m-|c_i|}\}$. Once we now $B' \subseteq B$ is such a subcollection, we can compute $C' \subseteq C$ by projecting each $b'_i \in B'$ onto the corresponding $c'_i \in C$, that is, by determining $c'_i = b'_i \cap S$. $|C'|$, eventually, provides the answer to the set packing instance, because the number of elements of $S \cup A$ covered by B' is related to the cardinality of B' by means of the constant m.

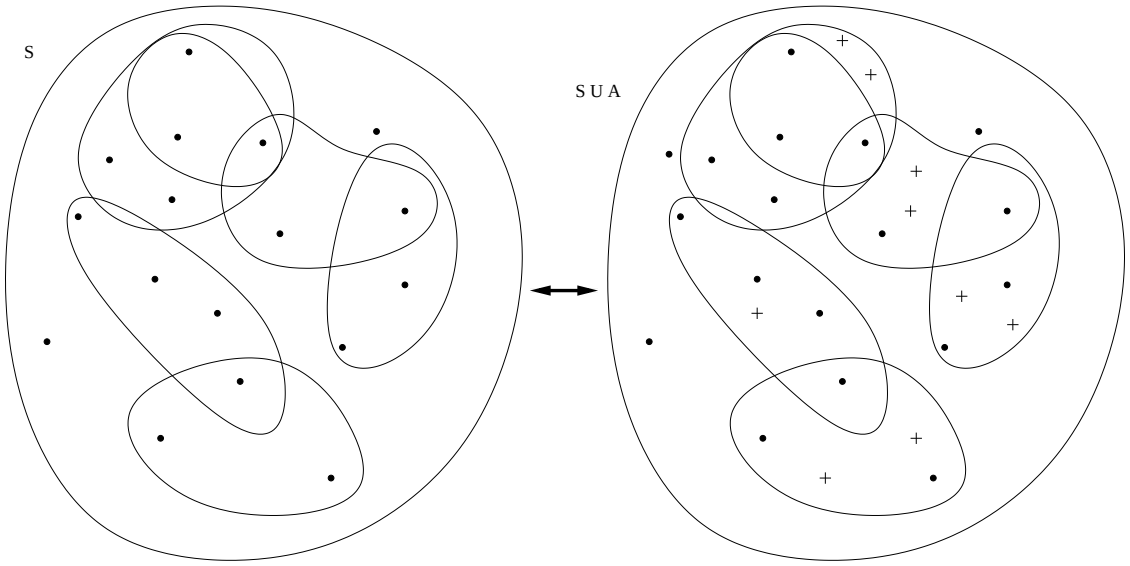


Figure 5.5: A schematic representation of the reduction from a *set packing* instance to an *optimal selection* instance. Crosses in the right part of the figure stand for elements of A added into subsets to equalize cardinality.

As determining an optimal selection is an intractable problem, a heuristic solution based on a greedy strategy has been adopted. First of all, FDs are sorted according to a partial order relation $\leq_R$. At this point, an FD f maximal according to $\leq_R$ is selected, the literals it covers are removed from the set of literals still to be covered, FDs with a non-null intersection with the selected FD are removed and the remaining FDs are sorted taking into account the reduced set of literals. The new best FD is chosen, and the process is iterated until it is no longer possible to select any FD, either because there is none left or because all of them would have a non-null intersection with one of the FDs already selected. A pseudocode definition of the algorithm is presented in Figure 5.6.

The logical representation of the template is such that literals formed on unary predicates designate the type of a given object, whereas literals formed on binary predicates are used to make assertions on objects, and correspond to template slots. Sentences will likely explicitely express such assertions, but will leave object types to be inferred by the reader on the basis of the grammatical function they appear in: this is the reason why unary predicates are not considered by the selection algorithm. Furthermore, binary predicates to be considered are only those forming literals that we do not want to repeat, and about which we want to be as exhaustive as possible. In the management succession domain, for instance, such predicates will include *io_person* and *succession_org*, because we don't want to repeat twice neither the person who was involved in a succession nor the organization where the succession took place. On the other hand, *person_name* will not be included, as the same person may be involved in more than one succession, and we want to be able to mention her more than once if necessary. The choice of which binary predicates to consider and which should be ignored can be made only on a semantic basis, and is domain-dependent.

The algorithm assumes that a partial order relation $\leq_R$ for comparing two FDs exists. A partial order well tuned on the definition of optimal selection is the following:

input: L {The set of literals on binary predicates in the template representation.},
F {The set of FDs after sentence combination.}
output: O {A selection.}

local variables: C {The set of literals not covered by any FD already in O.}
G {The set of FDs still suitable for selection.}
f {The FD currently set for being introduced in the selection.}

begin
  $O \leftarrow \emptyset$;
  $G \leftarrow F$;
  $C \leftarrow L$;
  while $C \neq \emptyset$ and $G \neq \emptyset$ do
  begin
    $f \leftarrow$ a maximal FD in G according to the partial order
      relation $\leq_R$;
    $O \leftarrow O \cup \{f\}$;
    $C \leftarrow C - S(f)$;
    $G \leftarrow G - \{g \in G | S(g) \cap S(f) \neq \emptyset\}$;
  end; (while)
  return O
end.

Figure 5.6: The greedy algorithm for FD selection.

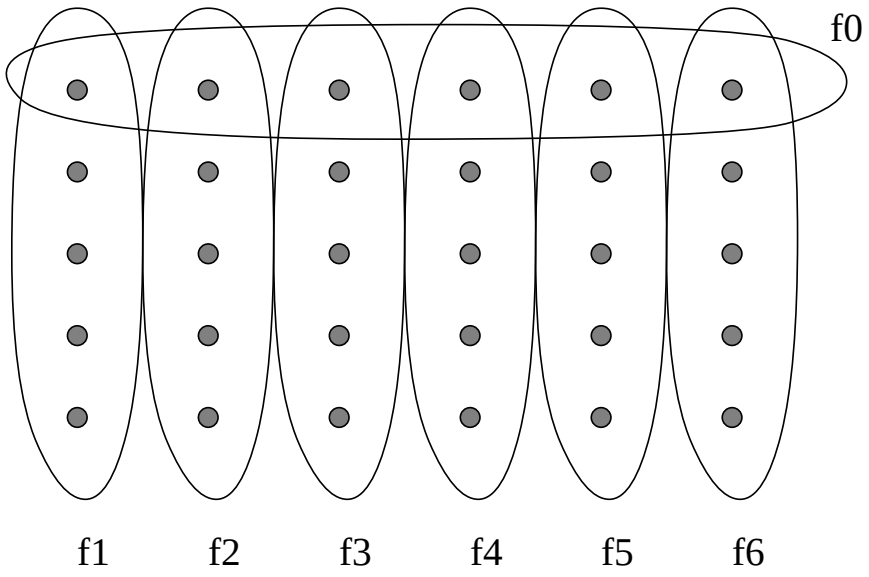


Figure 5.7: An unlucky configuration for the greedy selection algorithm: $f0$ is the FD selected first, because it covers more literals than any other, but its selection inhibits a better choice.

$$f_1 \leq_R f_2 \; if \; and \; only \; if \quad \begin{cases} |S(f_1)| < |S(f_2)|, \; or, \\ |S(f_1)| = |S(f_2)|, \; and \; C(f_2) < C(f_1), \; or, \\ |S(f_1)| = |S(f_2)|, \; C(f_1) = C(f_2), \; and \; not \; P^*(f_2, f_1). \end{cases}$$

In other words, f_2 is at least as good as f_1 if and only if:

- the number of literals covered by f_1 is lower than the number of literals covered by f_2, or;
- the two FDs cover the same number of literals but f_2 has fewer constituents than f_1, or;
- the two FDs cover the same number of literals, are made of the same number of constituents and it is not the case that f_1 was obtained by f_2 (and possibly some other sentence) by means of sentence combination.

Note that $S(f)$, $C(f)$ and P^* are correlated: sentences covering more literals will in general be longer. Combination rules, moreover, may combine sentences to form more sentences covering more literals.

The algorithm above is not particularly well suited to approximate the optimal solution of the abstract problem of selection. Situations such as the one in Figure 5.7 would lead to a choice far from optimal. In the picture, dots represent template literals, while each oval represents an FD. The first FD to be selected is f_0, as it covers the largest number of literals (six). Selecting f_0, however, prevents from selecting any other FD, so the selection composed of f_0 only would be returned. A much better selection, in this case, would of course be $O = \{f_1, f_2, f_3, f_4, f_5, f_6\}$, but it is not found. Note however that such situations are unlikely to occur in problem instances the algorithm is actually confronted with: FDs for basic sentences are either disjoint (because they tell about different events) or are contained in one another (because they tell about the same event, but one does it more exhaustively, or tells also about a different event). Sentence Combination, moreover, either leaves the set of literals unchanged or combines sets of literals. As a consequence, the selection returned by the greedy algorithm is usually either optimal or close to optimality.

The selection procedure described above assumes that all literals in the template are equally relevant. This assumption may not be valid: in the management succession domain, for instance,

expressing the name of the person entering a position is probably more important than expressing his title ("Ms.", "Mr." etc.). The procedure, however, can easily be generalized by giving each literal a weight depending on its predicate name.

A slightly simplified version of this sorting system is implemented and used in the prototype.

5.3.6 Alternative strategies for sentence combination and FD selection

As we said in the previous sections, both knowledge-base interrogation rules and combination rules are applied exhaustively: once a "fixpoint" where no more sentence combination can take place has been reached, a greedy algorithm is adopted for selecting a subset of FDs in the redundant set thus created.

Some simple profiling showed that, following this procedure, GeM spends more than 90% of its processing time in applying combination rules. A significant portion of this effort is eventually wasted, because it is devoted to producing FDs that will never be turned into sentences.

Having defined a way to score the quality of a given selection of FDs, the problem of finding an optimal selection can be seen as a problem of combinatorial optimization, to which well-known search algorithms can be applied. A possible procedure alternative to exhaustive sentence combination and subsequent selection consists in selecting a subset of FDs right after knowledge-base interrogation, before sentence combination takes place. This selection can be conveniently done using a greedy algorithm similar to that described above. Once the subset of basic FDs has been formed, combination rules can be used as operators to hill-climb the search space of all possible selections: once a rule has been applied to a set of FDs to form a new FD, the original subset is discarded and the new FD is retained. At each point in the search space, all combination rules are applied to all FDs in the current selection, and results are compared and sorted according to $\leq_R$: one among the best successor states is chosen, if it is better than the current state, and the search step is iterated. Hill-climbing stops, of course, when no combination rule can be applied, or when the quality of the selection cannot be further improved. In other terms, when a local maximum— hopefully also global— has been reached.

This search algorithm was implemented, and some experiments were carried out. Results, however, were discouraging. The problem with this simple hill-climbing approach is that the search space is not a connected graph: the choice of the initial selection greatly reduces the set of reachable states. This is easily understood by means of an example. Imagine that there are two succession events, the first one reporting about person P_1 replacing person P_2 in position S_1 within the company O, and a second one reporting on an analogous event, different only because position S_2 is concerned: this is actually a very frequent circumstance, because some positions— such as those of "president" and "chief executive officer"— are often held by the same person. We would like these two events to be combined and expressed by a single sentence in which position names appear conjoined. Now, if, for the two events, two basic FDs are selected having different classes for the head verb, as for instance the FDs corresponding to the two sentences below:

P_1 succeeded P_2 as O S_1;
O replaced P_2 with P_1 as S_2

then the desired combination would not take place, because head verbs would not match. This undesired effect is a consequence of the decision of performing a single step of sentence combination, operating on a representation with a given degree of linguistic commitment: the operation of factoring all constituents but the position could conveniently be done on a purely conceptual representation, before the knowledge-base is interrogated and basic FDs are formed. Spreading combination rules across different processing stages, however, would make the whole idea of hill-climbing the space of FD selections using combination rules as operators impossible.

A possible way to make the search space connected is by extending the set of operators. The problem in the example above, for instance, would be overcome if the substitution of a basic FD with another, with a different head verb but the same deep argument structure, were allowed. In this case, for example, the second FD could be replaced by that for the sentence:

P_1 succeeded P_2 as O S_2;

and the desired combination could actually take place. Note, however, that equivalence classes for basic FDs would still be an imperfect solution, because not only the search space is disconnected: it is also infested by local maxima. As it neither increases the number of literals covered by the selection nor decreases the number of constituents, a head-verb-replacing combination rule would be applied after all other rules were tried. If, in the previous example, a third FD could be combined with one of the two to embed a relative clause, this combination rule would be preferred. The presence of a relative clause, in turn, would inhibit the application of the combination rule for conjunction introduction, so that the best selection would not be reached. The same problems show up also if an hill-climbing search with two or three steps of look-ahead is adopted.

5.3.7 Sentence sorting and focus enforcement

Once a selection has been determined, the order in which FDs are sent to the linguistic realizer must be decided. In some cases temporal and causal relations among events are available, but this is not true in general. Usually, available information is not sufficient to structure the text according to rhetorical relations: FDs are simply sent as a sequence to the linguistic realizer.

This does not mean, however, that nothing can be done to make the text more readable. Remember that, at this stage, the syntactic variations according to which FDs will be realized are still underdetermined in most cases, FDs for which combination rules posed strict enough constraints being an exception. Sentence order and appropriate syntactic variations can thus be selected keeping into account phenomena related to *focus tracking* (see [Grosz and Sidner, 1986] for a general introduction to focus-related phenomena and [McKeown, 1985] for the specific uses of this phenomena in text generation).

At any given point in a discourse a single entity, or in some cases a set of entities, is said to be *the current focus*. This is the entity the writer is currently making assertions on, and corresponds generally to the grammatical subject of the sentence[10]. The focus for the next sentence cannot be chosen arbitrarily, if the discourse is to be cohesive. To explain how focus is constrained, two data structures need be introduced: the *potential focus list* and the *focus stack*. The potential focus list contains entities that were mentioned in the current sentence but with a grammatical function that did not put them in focus. The focus stack contains entities that were the focus of some previous sentence. There are four alternatives for what entity to consider as focus for the next sentence:

1. the focus can remain the same;
2. the focus can be an element of the potential focus list;
3. the focus can be an entity of the focus stack;
4. the focus can be an entity that is in none of the above structures.

For better cohesiveness [McKeown, 1985] establishes preference criteria that were found appropriate also for our texts in the management succession domain. These criteria state that:

[10] The grammatical role the focus is associated with depends on the structure of the sentence. In a sentence such as "This is the entity the writer is currently making assertions on", for instance, the focus is "the entity", and not "the writer", that would be the referent of the anaphoric "this" in subject position.

- The shift to an element in the potential focus list should be preferred over keeping the same focus. The motivation for this is that otherwise, when propositions dealing with such entity will have to be added, it will have to be reintroduced somehow later;

- Keeping the same focus is preferable to reverting to a previous focus, now in the focus stack. When understanding a text, in fact, the reader tends to assume that, when the focus is shifted to an entity that is neither a newly introduced entity nor the current focus, then all the writer wanted to tell about the old focus has been told. If the focus is put again on a "popped" entity the resulting text is unnatural;

- Popping a focus from the focus stack is preferable to focusing on an entity that was never mentioned before. The second situation, in fact, signals a fracture in the discourse structure that should be present only if there is a real shift in topic.

Sentence order and syntactic variants are chosen taking these considerations into account. The first FD is chosen as the one corresponding to the first event in the template, according to the assumption that the order of objects for events in the template corresponds to the order in which events were presented in the source text, and that the order of events in the source text should be kept, if there is no reason to do otherwise. The simple active syntactic variant is selected for the first sentence. The subject of the first FD is thus chosen as the current focus, and all other mentioned entities are inserted in the *potential focus list*. Remaining FDs are then searched for an FD mentioning the entity in the potential focus list. If such an FD is found, it is selected to be the next in the text, a syntactic variant is chosen that keeps that entity in the focus, the old focus is pushed onto the focus stack and the potential focus list is updated. If more than one such FD exists, then the FD reporting on an event occurring earlier in the template (thus, in the source text) is preferred. If no FD mentions entities in the potential focus list, then a FD mentioning the current focus is sought, and the potential focus list is updated. Again, ties are broken using positional precedence in the template. If neither the search in the potential focus list not the search of the current focus succeeds, then previous foci are popped from the stack until one is found for which an FD mentioning it is still to be produced. If the stack is empty, then the FD reporting on the earlier event in the text among those left is selected and the procedure is iterated.

As an example, consider the FDs in Figure 5.8, sorted according to the order in which reported events occur in the source text[11], and very simplified for the sake of clarity. Suppose also that only persons are considered as potential foci.
FD (1) is chosen as the first FD to be realized, as it reports about succession events earlier than all others in the source template. The syntactic form is set to be the simple active form. At this point, "Daniel Glass" is the focus, and "Charles Koppelman" is in the potential focus list. We indicate this as follows:

$$\begin{aligned} focus &= \{\text{“}Daniel\ Glass''\} \\ pfl &= \{\text{“}Charles\ Koppelman''\} \\ fs &= \emptyset \end{aligned}$$

Remaining FDs are searched for an FD mentioning "Charles Koppelman". FD (3) is such an FD, so it is selected. "Charles Koppelman" is chosen as the new focus and the FD (3) is put in passive form. At this point the focus state becomes:

[11]For FDs reporting about more than a single event, the earliest event is considered.

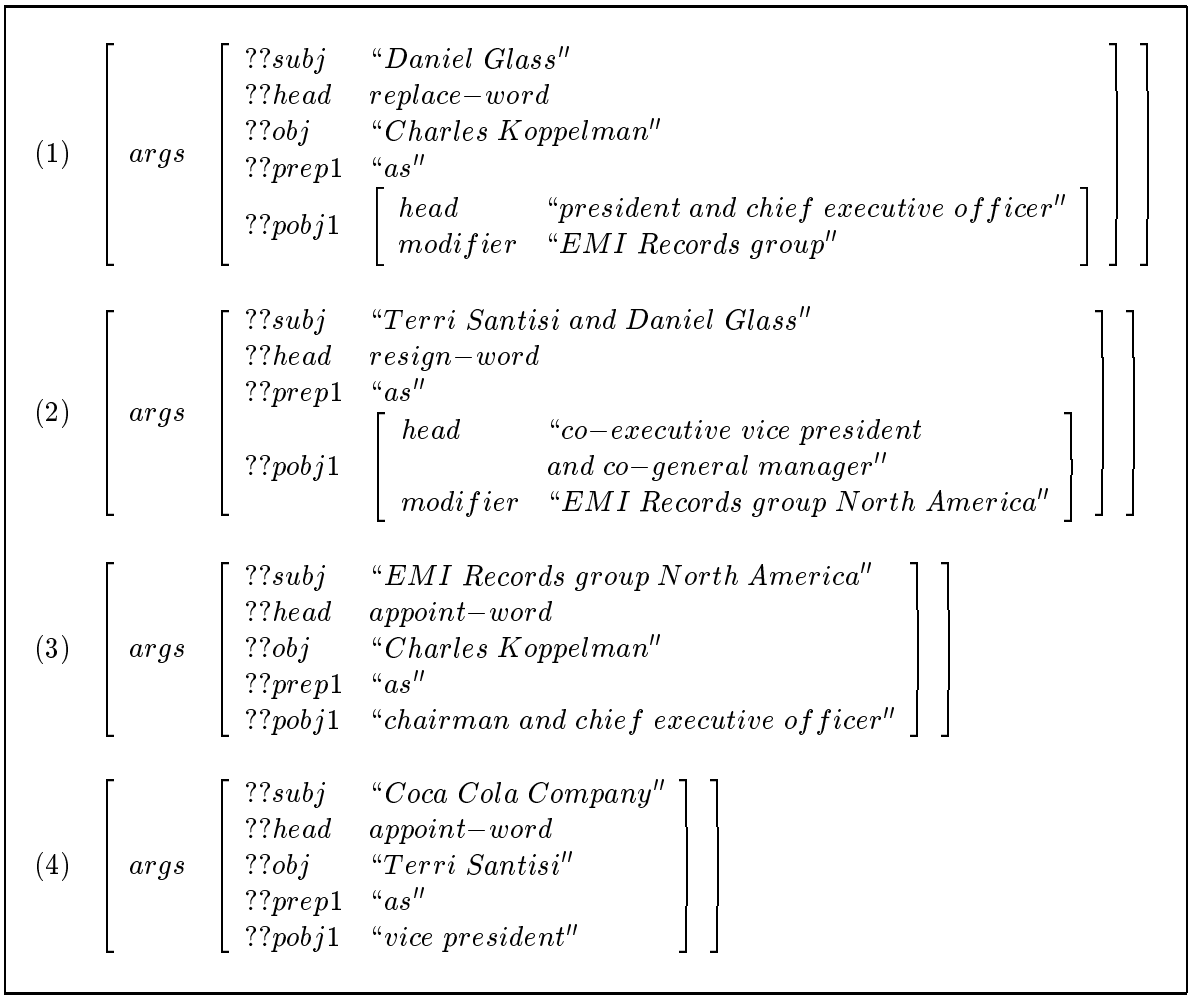


Figure 5.8: FDs sorted according to event order in the source text. Syntactic variants are still to be assigned.

$$\begin{aligned} focus &= \{\text{“}Charles\ Koppelman''\} \\ pfl &= \emptyset \\ fs &= \{\text{“}Daniel\ Glass''\} \end{aligned}$$

The potential focus list is empty, so an FD mentioning the current focus is sought. Apart from FDs (1) and (3), however, no other FD mentions “Charles Koppelman”, so “Daniel Glass” is popped from the focus stack. FD (2) has “Daniel Glass” as one of its logical subjects, so it is selected and put in active form. Now it is:

$$\begin{aligned} focus &= \{\text{“}Daniel\ Glass''\} \\ pfl &= \{\text{“}Terri\ Santisi''\} \\ fs &= \emptyset \end{aligned}$$

Eventually an FD in which “Terri Santisi” occurs is searched for, and it is found. The last FD is then number (4), and is put in passive form.

The FDs in the order in which they will be sent to the linguistic realizer and with the appropriate syntactic variants are shown in Figure 5.9. The output text will be:

> • Daniel Glass replaced Charles Koppelman as EMI Records group president and chief executive officer.
> • Charles Koppelman was appointed as chairman and chief executive officer by EMI Records group North America.
> • Daniel Glass and Terri Santisi resigned as co-executive vice presidents and co-general managers of EMI Records group North America.
> • Terri Santisi was named vice president of Coca Cola Company.

As we said in describing the FD ordering algorithm, the first sentence is selected as that reporting on the first event in the template. As weak as it is, this is a valid criterion since nothing is known about the reader's specific interest. A possible improvement, however, consists in having the reader specify her interests in terms of appropriate entities in the domain, such as persons and organizations in the management succession domain. Such a user profile could then be taken into account in selecting the first sentence by initializing the pfl variable to it.

5.3.8 Linguistic realization

Once FDs have been sorted and attributed a syntactic variant, they are sent to the linguistic realizer to be transformed into an actual sequence of sentences.

In the proposed architecture this is accomplished by means of a functional unification grammar (FUG) [Kay, 1979], a good deal of which can be obtained by transforming FASTUS' library of patterns.

A functional unification grammar is specified as a disjunction of feature structures, each possibly containing, in turn, other disjunctions (see [Carpenter, 1992] for a thorough introduction to feature structures). A very simple functional unification grammar is presented in Figure 5.10.

Each branch in the disjunction at the top level corresponds to a grammatical constituent category. For instance, in the example FUG, there is one branch each for sentences (S), noun phrases

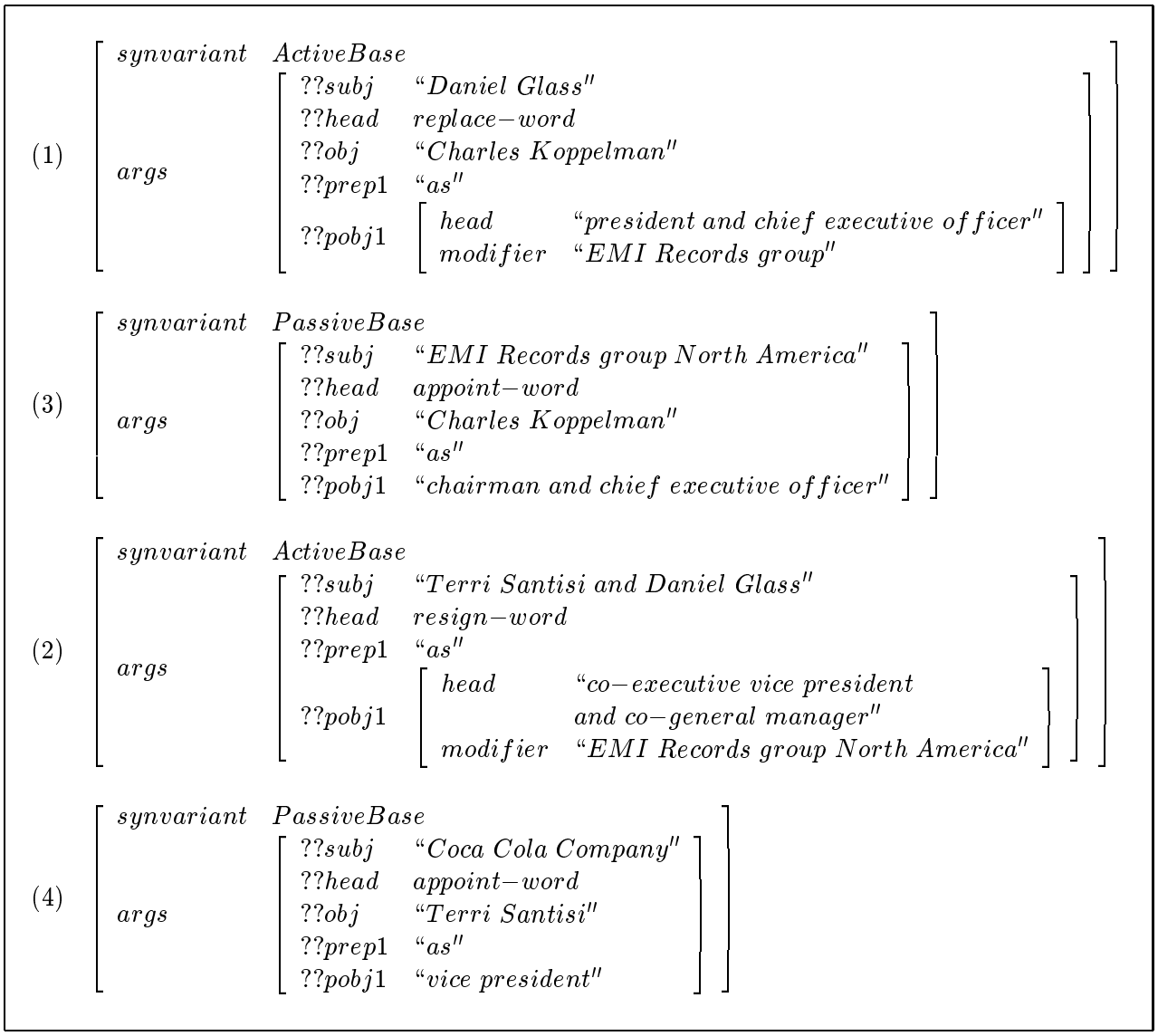


Figure 5.9: FDs in the order in which they will be sent to the linguistic realizer. Syntactic variants have been determined.

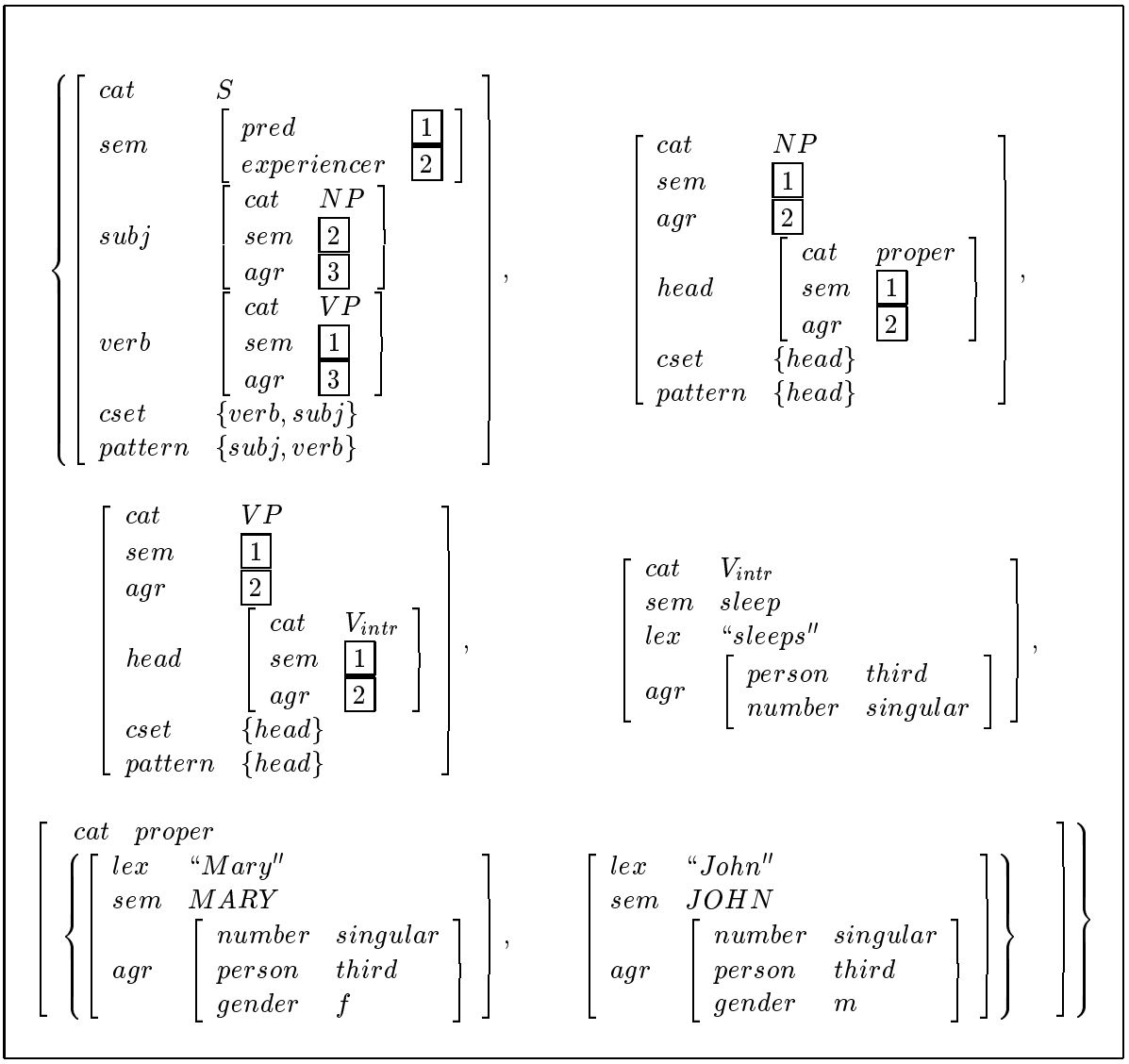


Figure 5.10: A very simple Functional Unification Grammar. Alternatives are enclosed in braces.

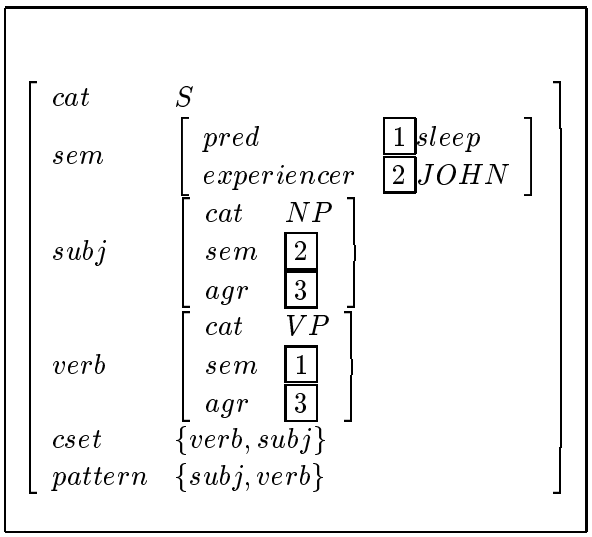


Figure 5.11: The input FD after unification at top level with the grammar in 5.10.

(NP), verb phrases (VP), intransitive verbs (V_{intr}), and proper names (*proper*) respectively. The initial sentence specification must be provided as a Functional Description (FD), that is, a feature structure without disjunctions, the category of which is specified by means of the distinguished feature *cat*. As an example, let's consider the following input FD:

$$\begin{bmatrix} cat & S \\ sem & \begin{bmatrix} pred & sleep \\ experiencer & JOHN \end{bmatrix} \end{bmatrix}$$

This input FD is initially unified with the grammar (namely, with the appropriate top level branch of the grammar). In our example, unification takes place with the first branch, for constituents of category S, and leads to the feature structure in Figure 5.11. If unification succeeds, then a subset of the features in the resulting feature structure is selected, and each of the corresponding values is unified with the grammar in turn. The subset of features to be considered for unification is specified as the value of another distinguished feature: *cset* (for *constituent-set*). In the example, the value of the *cset* feature is $\{verb, subj\}$, so the values of features *subj* and *verb* are unified with the grammar. This leads to the Feature Structure (FS hereafter) in Fig. 5.12. The process is iterated until either unification fails, and no output is produced, or all the *csets* still to be expanded are empty, and the complete feature structure can be *linearized*. Iterating unification in the example the feature structure in Fig.5.13 is eventually produced. Linearization is performed relying on two other distinguished features: *pattern* and *lex*. The *pattern* feature is meant to state linear precedence among constituents. In other terms, its value is an ordered list of feature names, and it plays the role of the Context Free backbone of other unification-based grammars such as Lexical Functional Grammars [Kaplan and Bresnan, 1982]. At each level of nesting within the unified feature structure, the *pattern feature* is searched for. If it is not present at the current level, then the linearization algorithm looks for the *lex* feature. This feature is constrained to always having a string as its value. The value of the *lex* feature is then returned as result of the linearization. Otherwise, if the *pattern* feature is present, the list of features is extracted. Each feature from the list is examined in turn, the linearization algorithm is applied to the feature structure appearing as its value, and the strings resulting from linearizing each of them is concatenated to form the overall result. In our example, there is a feature *pattern* at top level, whose value is $\{subj, verb\}$.

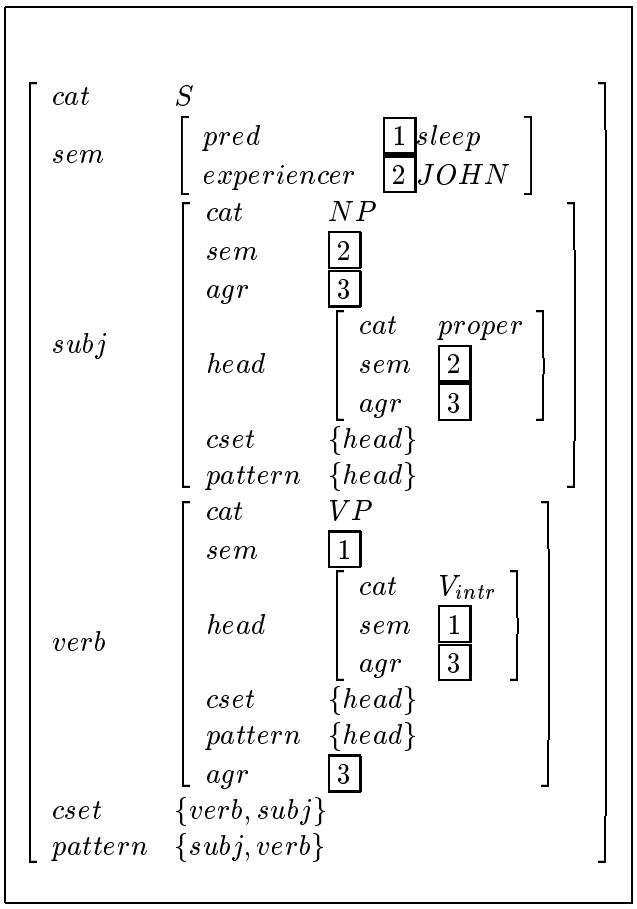


Figure 5.12: The input FD after unification of the *subj* and *verb* features.

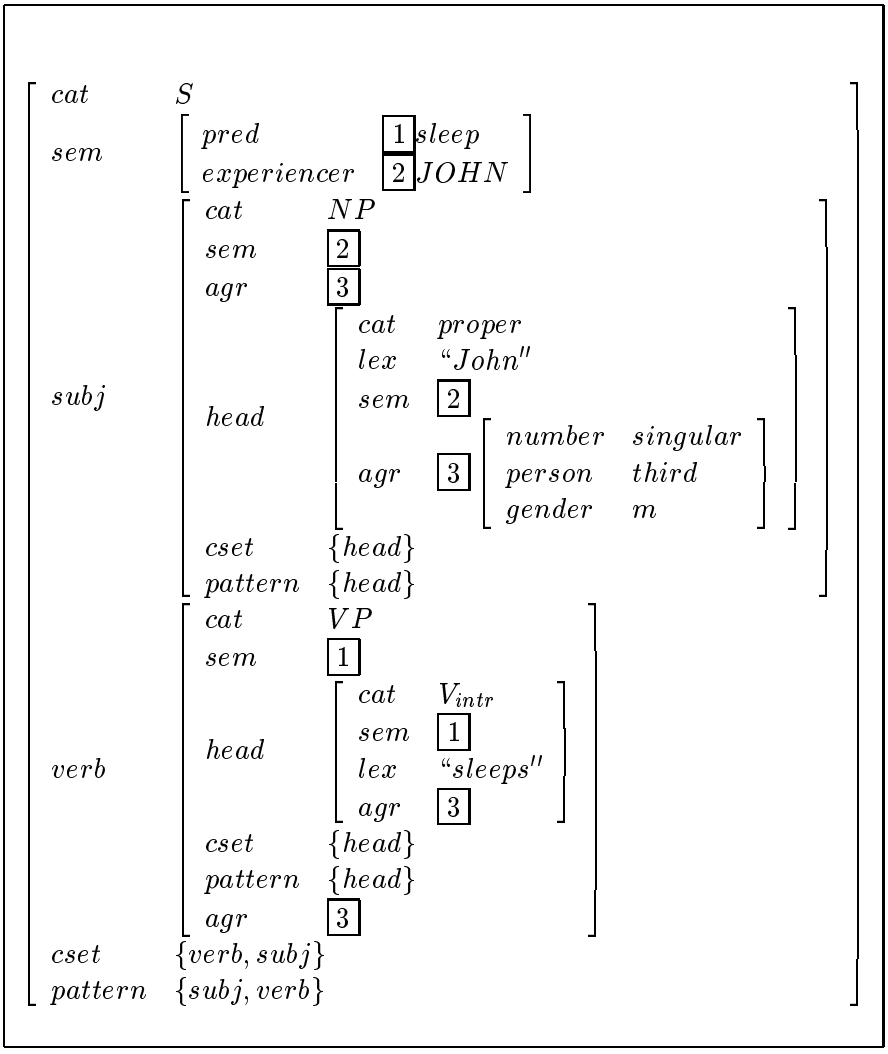


Figure 5.13: The input FD once recursive unification with the grammar is complete.

Both FSs appearing as values of *subj* and *verb* contain a feature *pattern* in turn, so the descent continues. Values of the paths $< subj, head >$ and $< verb, head >$, however, miss a *pattern* feature, so a *lex* feature is sought. Such a feature is found in both cases, and their respective values are concatenated leading to the sentence "John sleeps".

FUF [Elhadad, 1993][12] is a freely available unification engine specifically conceived for implementing Functional Unification Grammars and written in Common Lisp. Grammars and input FDs are written using an appropriate syntax and are loaded into the Lisp environment. FUF is accessed through an application program interface composed of a library of functions. Functions include different kinds of unification and other FD manipulation services. A linearization function that also takes care of morphological inflection of words for English is provided as well. When unifying an input FD and a grammar, FUF follows a default breadth-first strategy, going left-to-right with respect to features in *csets*, but special features are provided to guide the unification process, limiting backtracking and gaining efficiency.

A comprehensive grammar requires a significant effort to develop. Should a complete such grammar be rewritten from scratch every time the system is ported to a new domain, porting costs would be unacceptable. To avoid this problem, one could think of developing once and for all a very-broad-coverage grammar, that needs not be modified when going to a different domain or application. This is indeed the approach chosen in [McKeown and Radev, 1995] and in [Robin, 1994]. In both cases a comprehensive English grammar for generation is used (actually, the same one), and the claim is that the grammar is general enough that it would need only minimal adjustments to be ported to new applications. Such a large grammar, however, can be a very complex object to deal with, even if only minor changes are necessary. The effort required just to understand its structure, before thinking how to modify it, can be significant. Moreover, if a Functional Unification Grammar such as that used by the authors of the mentioned works is used, much care must be posed in setting features to carefully control unification, because otherwise the search space can grow to a point where efficiency is compromised: control features, however, introduce procedurality into a declarative resource, and should be used with caution.

For GeM a different solution was preferred: reuse of FASTUS resources. Remember that FASTUS grammar is stratified so that successive passes recognize increasingly structured objects.

Ideally, one would like to automatically compile FASTUS grammar into a grammar suitable for generating the same kind of texts FASTUS analyses. This is not possible, however, because being conceived for an Information Extraction task, FASTUS grammar is often allowed to overgenerate, when this turns into a significant simplification of its rules. Overgeneration means that the grammar will accept and parse also sentences that are not really grammatical for the natural language being parsed. As one can assume that actual input will be composed of grammatical, or nearly grammatical, texts, this is not a problem. Overgeneration, however, is not tolerable in generation, because it amounts to producing incorrect output. Moreover, the variety of constructs and phenomena that must be taken into account in an IE grammar is generally well beyond that required for a grammar for generating texts from the same templates. Even if overgeneration were not a problem, then, by compiling FASTUS grammar into a generation grammar one would end up with an unnecessarily huge and inefficient resource.

A more sensible way of making use of FASTUS resources consists in exploiting the separation between domain-dependent and domain-independent resources. As we said in Section 5.3.2, FDs for basic sentences are "isomorphic" to FASTUS domain-dependent patterns. This isomorphism is usefully carried on to the grammar, at least to its top level, where it deals with full-sentence structure. Allowed syntactic variants are given together with FASTUS domain-dependent patterns. For example, the instantiated FD schema (See Section 5.3.2):

[12] FUF 5.2 is available via anonymous ftp as ftp://ftp.cs.columbia.edu:/pub/fuf/fuf5.2.tar.Z.

```
ActiveBase:  EVENT-PHRASE →
             EVENT-ADJUNCT* (",") (NG[??subj]:1 (COMPL | COMPL1))
             VG[Active=T,Subcat=Basic,(??head)]:2
             (NG[??obj]:3)
             {P[??prep1] NG[??pobj1]:4 | EVENT-ADJUNCT}*;
             ??semantics;;
```

Figure 5.14: A simplified version of FASTUS domain-independent pattern for the simple active form. $COMPL$, $COMPL1$, and $EVENT-ADJUNT$ nonterminals eat up modifiers that have no effect on the event representation eventually produced.

$$\left[\begin{array}{ll} & \left[\begin{array}{ll} ??subj & "GeneralMotors" \\ ??head & appoint-word \\ ??obj & "RogerSmith" \\ ??prep1 & as \\ ??pobj1 & "president" \end{array}\right] \\ args & \end{array}\right] \tag{5.4}$$

is declared as being allowed to appear in variants such as the simple active form:

"General Motors appointed Roger Smith as president"

the simple passive form:

"Roger Smith was appointed as president by General Motors"

the active gerundive form:

"... General Motors appointing Roger Smith as president..."

and several others. In FASTUS, domain-independent FASTSPEC patterns for syntactic variants map deep arguments onto surface grammatical arguments. The portion of the functional unification grammar for sentence-level constituents that specifies the corresponding mapping can be derived from such domain-independent patterns. So, for instance, from the domain-independent FASTUS pattern for the simple active form (Fig.5.14), the corresponding branch for *cat S* in the grammar used for generation can be easily inferred (Fig.5.15).[13]

The grammar branch for full sentences can thus be inferred from FASTUS resources. Branches for lower level ranks, such as noun phrases and verb phrases, however, are conveniently written independently of FASTUS. First of all, the variety of structures sufficient to produce an acceptably fluent text is much lower than the variety that must be dealt with in analysis. Secondly, grammar rules at these ranks must deal with the structures built by combination rules. Sentence Combination does not have an explicit parallel within FASTUS, so there is no resource that can be reused in this case. Note however that most combination rules are quite portable across domains, and so is the part of the Functional Unification Grammar that must be written to accommodate their results.

[13] Agreement constraints, such as that represented by the coreference link in the figure between paths $<$ *subj head number* $>$ and $<$ *head number* $>$, are not present in FASTUS pattern and must be encoded directly.

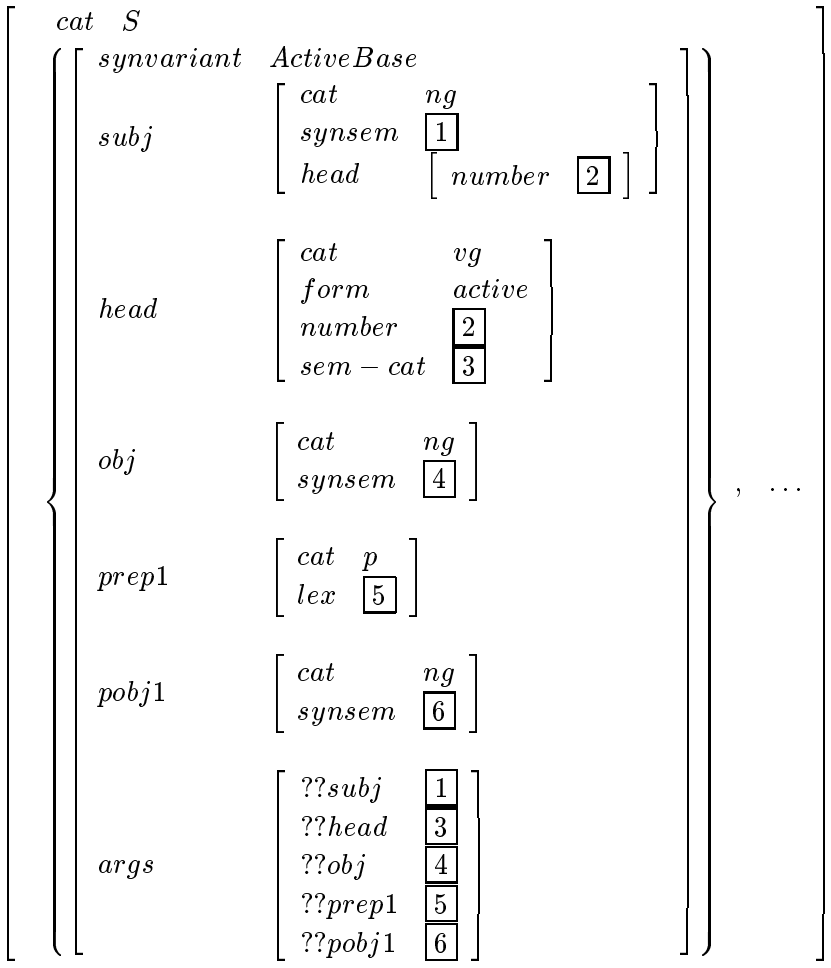


Figure 5.15: A fragment of the top level of the Functional Unification Grammar used for generation. The mapping from the deep cases under the *args* feature to the surface grammatical roles is indicated by coreference links.

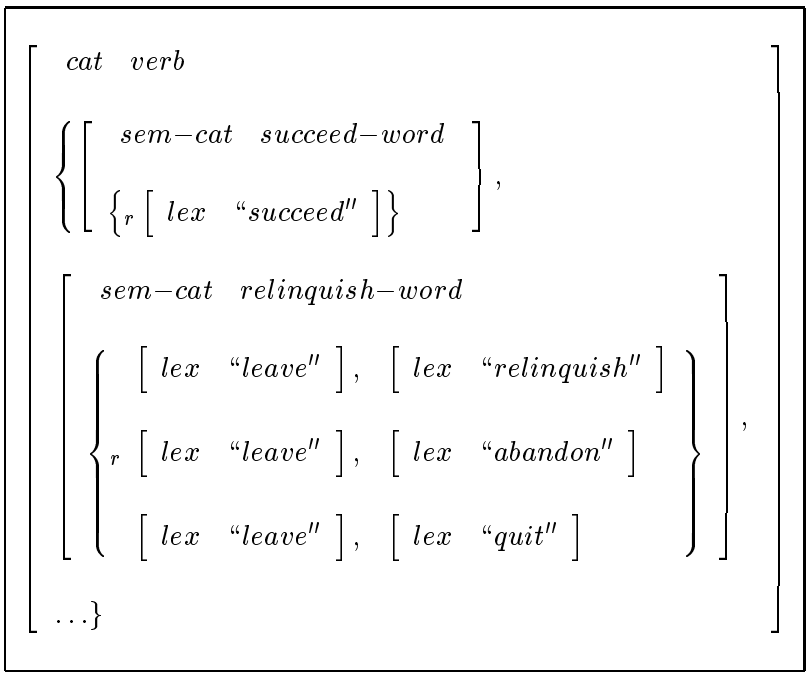


Figure 5.16: A fragment of the lexical portion of the grammar.

At the lowest level, however, there is another FASTUS resource suitable for reuse: the lexicon. FASTUS lexicon relates words with their syntactic and semantic properties. These properties are then checked across all levels of analysis. As for generation this process is reversed, lexical properties can be used to specify constraints on words at all levels in the grammar. The lexicon is then inverted to relate properties to all words possessing them. FASTUS lexicon, however, needs be integrated with information on agreement features, such as grammatical number, person and gender for nouns and adjectives, mood and tense for verbs and so on, that are missing. In a Functional Unification Grammar the lexicon is not a separate entity, as we saw in the simple example where there were grammar branches for individual words such as "sleeps". This is a useful feature in generation, because it leads to organizing the lexicon according to semantic features, factoring redundancies. In the simple lexicon implemented for experiments, for example, verbs are grouped according to the features attributed to them in FASTUS lexical entries. A fragment is represented in Figure 5.16. The small r appearing sometimes near braces is not standard for FUGs, but marks a special control feature provided by FUF: a *randomized disjunction*. When encountering a normal disjunction, FUF tries first unification with the first disjunct. If unification succeeds, then the result is returned, otherwise the second alternative is tried, and so on. In a randomized disjunction, instead, an alternative is selected randomly and unification is attempted. If it fails, a second alternative, among those still to be tried, is chosen at random, and so on. Randomized disjunctions are useful for introducing lexical variety in texts without the need of explicitly managing an *ad hoc* data structure. Note moreover, that, by repeating branches an uneven number of times, the generator may be induced to repeat a word more often than others. An example is the branch for *relinquish−words* in Figure 5.16, in which the verb "to leave" is repeated three times in order to be chosen as often as all the other alternatives considered as a whole.

Linguistic realization is the last processing phase in the basic architecture.

A first example summary produced by GeM is shown in Figure 5.17 together with the source text it comes from. This example shows how significant the reduction in length can be: only relevant

Source Text:

Insilco Corp., Midland, Texas, said it is emerging from Chapter 11 bankruptcy proceedings.

The holding company, which has interests in the automotive, electronics, defense and consumer products industries, also said Joel L. Reed resigned as president and chief executive officer but will continue as chief operating officer until a new chief executive is named. He then plans to leave the company, an Insilco spokesman said.

Under the reorganization plan, the principal holder of the company's subordinated debt, Water Street Corporate Recovery Fund I L.P., an investment partnership of which Goldman, Sachs & Co. is the general partner, will own about 51% of the company's newly issued common stock.

Summary:

- Joel L. Reed resigned as Insilco Corp. president and chief executive officer.
- Joel L. Reed will leave as Insilco Corp. chief operating officer.

Figure 5.17: A first text on management successions and the corresponding summary generated by GeM.

Source Text:

Paul G. Allen, the billionaire co-founder of Microsoft Corp., has started a company and named longtime friend Vern Raburn its president and chief executive officer.

The company, to be called Paul Allen Group, will be based in Bellevue, Wash., and will coordinate the overall strategy for the group of high-tech companies that Mr. Allen owns or holds a significant stake in. The multimedia and communications companies include Asymetrix Corp. and Starwave Corp., both of Bellevue, Wash., and Interval Research Corp. in Palo Alto, Calif.

In 1979, Mr. Raburn joined Redmond, Wash.-based Microsoft, where he was president of the consumer-products division. Before Paul Allen Group, he worked at Lotus Development Corp., Cambridge, Mass., as executive vice president, and at Cupertino, Calif.-based Symantec Corp. as chairman. He founded Slate Corp., a Scottsdale, Ariz., software company that makes pen-based applications.

Summary:

• Vern Raburn, who was Lotus Development Corp. executive vice president and Symantec Corp. chairman, will be Paul Allen Group president and chief executive officer.

Figure 5.18: A second example of a summary generated by GeM.

information concerning succession events is presented. The power of combination rules, however, emerges more clearly from the second example, in Figure 5.18. In this second case, information on position changes that was spread across the source text is condensed into a single, concise sentence. This sentence results from four IN_AND_OUT objects in the template: two of them are conjoined in the main clause, and two more form the embedded relative clause. An example of a longer summary actually generated by GeM — four sentences — was already proposed when the sorting schema based on focus tracking was illustrated (see Section 5.3.7).

Summaries in the management succession domain range from one to eight or nine sentences, each expressing from one to even six or eight individual descriptions of people entering or leaving positions (IN_AND_OUT objects).

5.4 The string inflection extension

In Section 4.3 we stressed that MUC templates contain strings of free text that must be inserted in the final summary. In the basic architecture, however, textual strings coming from slot values are considered as canned text: they are simply inserted as such. Unfortunately this very simple

approach can lead to ungrammatical text. If the semantics of the slot is not sufficient even to guess the grammatical category of the top constituent in the string, then the problem is obvious: noun phrases and adjectival phrases, for instance, must be immersed in different contexts for the whole sentence to be grammatical. This situation, however, is unlikely to occur, because usually the grammatical category of fillers of a same slot is uniform throughout different objects and templates. Even if we know the category, though, the risk of ungrammaticality is still present, because errors can be done in enforcing agreement. In the management succession domain, for example, proper names are always singular in number, and cause no problem. Position names, instead, can appear in singular form in the original template but require insertion in plural form as a consequence of applying combination rules.

In the basic architecture, textual strings are treated as individual lexical items: the FUF linearizer determines an appropriate ending according to number and person features as for any other word. So, for instance, if "vice president" is to be turned into plural form, the appropriate *number* feature is set to *plural* and FUF linearizer appends a final "s", for a resulting "vice presidents". In this special case everything works fine, because the syntactic head of the noun phrase in the string is the last word, so that adding an "s" to the whole string corresponds to adding an "s" to its head, that is the correct way of forming a plural. This works most of the times, but what happens if the syntactic head is not the last word in the string? In this case the plural is formed incorrectly. If the position is "vice president for U.S. operations", the FUF linearizer will produce "vice president for U.S. operationses", that is ungrammatical. It is thus clear that in order to form correctly "vice presidents for U.S. operations" there is no alternative to finding the syntactic head ("president" in the example), making the plural of it and leaving all the rest unmodified. To find the syntactic head, in turn, it is necessary to parse the position name string with an appropriate grammar. Position names are specific to the management succession domain, but analogous situations can emerge in any domain, whenever a string can be an arbitrary NP headed by a common noun, so the addition of a parser to the basic architecture goes surely in the direction of a greater flexibility.

A parsing step can be integrated into the basic architecture in at least two different ways. A first alternative consists in calling the parser right after the initial conversion from the MUC template file into the knowledge base. Some roles in the description of the knowledge base structure may be marked as "to be parsed", and a global parsing of strings occurring as fillers of such roles may be carried out before knowledge base interrogation begins. This approach has the disadvantage that all marked strings are parsed, even if they will not be included in the final text, or if they will but they will not need a change in any grammatical feature. To avoid useless parsing, then, one could parse only those strings that actually need have some features modified. As feature alteration is a consequence of sentence combination, a good position for a parsing step could be after sentence selection: in this way the number of calls to the parser is minimized. Note however that the availability of parsed textual strings could turn out useful for sentence combination purposes. To implement pronominalization rules, for instance, a shallow parsing of proper names, to figure out from the first name whether a feminine or a masculine pronoun should be used, would be necessary. This last consideration, but above all the fact that a parsing of all marked strings, independently of feature manipulation, will be necessary when cross-linguistic generation will be considered, lead to the choice of performing parsing right after knowledge base loading.

The way the parsing step has been integrated into the basic architecture is depicted in Figure 5.19.

To parse textual strings a right-to-left bottom-up chart parser has been implemented, together with a lexicon and a grammar for position names. The parser begins by initializing the chart with lexical edges. All chart vertices are then considered in turn, starting from the one before the last and proceeding backwards. At each vertex, all edges starting from it are considered, and for each of them the grammar is checked for rules whose right-hand side begins with an edge of the appropriate

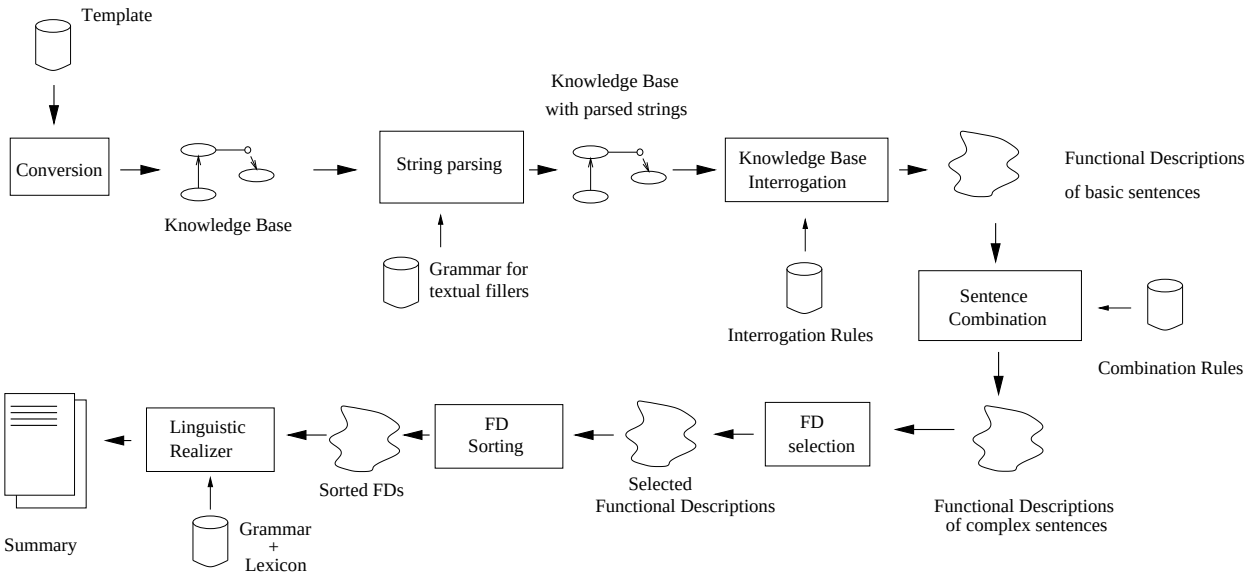


Figure 5.19: The generator architecture modified to parse textual strings.

category. Rules are indexed according to the grammatical category of the first element in their right-hand side, so that retrieval of relevant grammar rules is very efficient. As soon as it is not possible to add any new edge to the set of edges leaving the current vertex, the vertex immediately to the left is considered. Parsing ends when the leftmost vertex has been expanded, and it is not possible to add any other edge to the chart.

Note that by proceeding bottom-up and considering vertices in a direction (right-to-left) opposite to that followed when checking the elements in the right-hand side of rules (left-to-right), the chart remains much less crowded than in usual chart parsing. This is due to the fact that it is no longer necessary to explicitly store active edges. When a rule is considered for expansion, in fact, all edges that can potentially help in matching the right-hand side are already in place. Given a rule r and an edge e starting from a vertex v_i and ending in vertex v_j, then:

1. a *tree of edges* T is built. In this tree, each node corresponds to an edge. The root corresponds to e, and all its children are edges starting from v_j with a grammatical category suitable for matching the second constituent in the right-hand side of r. The construction is repeated recursively, and it ends when the last element of the right-hand side of r has been matched. T, eventually, has a depth equal to the number of elements in the right-hand side of r decreased by 1;

2. a *list of tasks* is formed, each task corresponding to a complete path in T, from the root to a leaf;

3. for each task in the list, unification is attempted and, if it succeeds, the resulting inactive edge is added to the chart.[14]

In other words, an agenda is recreated every time a triple composed of a vertex, an edge and a rule is considered. All the tasks in it are completed, and the agenda is emptied, before the next

[14] Were the grammar used a simple Context Free Grammar, this last step would not be needed: all "tasks" would correspond to new edges to be inserted in the chart. As will be explained later, however, the grammar used is a unification grammar, so that checking grammatical categories of edges is not sufficient: if the unification of the corresponding feature structures fails, then no new edge must be inserted in the chart.

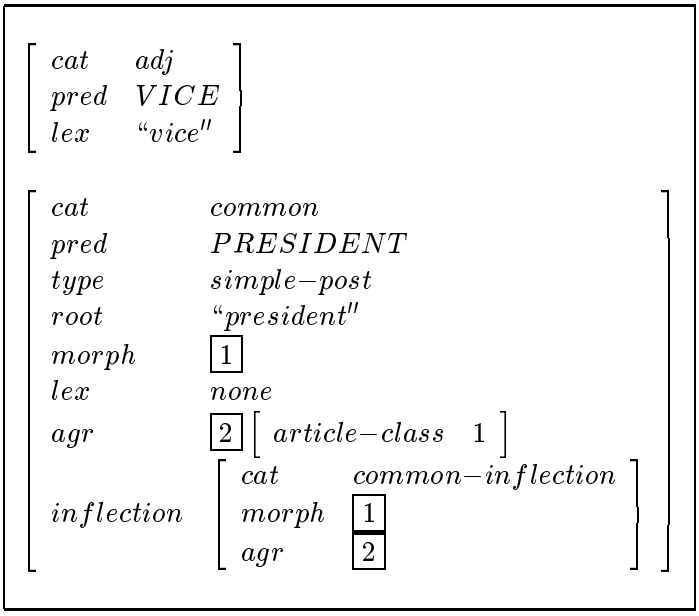


Figure 5.20: Lexical entries for words "vice" and "president".

triple is examined. Note that the way the tree of edges for a triple is transformed into a list of tasks amounts to introducing a form of local backtracking that increases the worst-case complexity of parsing: the check that an edge early in the right-hand side of the rule is unifiable is repeated for all the edges on its right. This could be avoided by performing rule unification incrementally, one edge at a time, and storing partial results — actually active edges — together with nodes in the tree of edges. This would require however a number of additional copies of possibly complex features structures to be carried out, thus possibly eroding the gain in execution time. This trade-off can be made object of a further analysis; so far, the overhead introduced by repeated unification appears negligible.

Once the leftmost vertex has been expanded, the chart is examined. If there is a single complete parse, i.e. an edge starting from the first vertex and ending in the last, then it is selected as result. If there is more than one such parse, then the whole set of complete parses is returned. If there is no complete parse, then the chart is searched for edges of the initial grammatical category spanning the largest number of vertices, and the set of these longest-spanning edges is returned as result.

The parser is designed to operate with a unification grammar, and relies on FUF for unifying feature structures related to edges. Lexical items are feature structures, and are retrieved at chart initialization time. As an example, feature structure for lexical items "vice" and "president" are shown in Figure 5.20. The lexical entry for "vice" is very simple: it just states that it is an adjective, its semantics is the atom VICE, and it is spelled out as "vice". The lexical entry for "president", instead, is a bit more complicated. Apart from being declared as a common noun and being related to the predicate PRESIDENT, it is given a type *simple−post*. Though based on generic linguistic categories, the grammar is specific to position names, and the presence of a semantic classification for lexical items helps reducing the generation of unnecessary edges. Features *morph* and *root* are motivated by the following considerations. Parse trees obtained from parsing are such that the corresponding strings can be produced by simply feeding them to a linearizer. As the idea is to have a single linearization engine also for other languages, however, that provided together with the FUF package is not sufficient, being very specific to the English language. A new linearizer, then, had to be implemented, allowing for more complex inflection systems typical of many languages.

$$\begin{bmatrix} cat & common\text{-}inflection \\ morph & \boxed{1} \\ agr & \left\{ \begin{bmatrix} number & sing \\ morph & \boxed{1}\text{“}\prime\prime \end{bmatrix}, \begin{bmatrix} number & plur \\ morph & \boxed{1}\text{“}s\prime\prime \end{bmatrix} \right\} \end{bmatrix}$$

Figure 5.21: Lexical entry for allowed inflections of common nouns.

$$\begin{bmatrix} cat & common \\ pred & PRESIDENT \\ type & simple\text{-}post \\ root & \text{“}president\prime\prime \\ morph & \boxed{1} \\ lex & none \\ agr & \boxed{2}\begin{bmatrix} article\text{-}class & 1 \end{bmatrix} \\ inflection & \begin{bmatrix} cat & common\text{-}inflection \\ morph & \boxed{1} \\ agr & \boxed{2}\begin{bmatrix} number & plur \\ morph & \boxed{1}\text{“}s\prime\prime \end{bmatrix} \end{bmatrix} \end{bmatrix}$$

Figure 5.22: The feature structure built by the lexicon look-up for the word “presidents”.

Each lexical item is then divided into a *root* and a *morph*. The idea is that all forms of a single lexical item share the *root*, while the *morph* depends on agreement features, as it is shown by the coreference link $\boxed{1}$ in the lexical entry. Note that agreement features are underspecified in the lexical entry for the root “president”: they just state that it falls into the *article−class* 1, that of words requiring “a”, and not “an” as indefinite article. Grammatical number, the only other agreement feature for nouns in English, is specified in the lexical entry for the suffix. The allowed class of suffixes is indicated as value of the $< inflection\ cat >$ path, *common−inflection* in the example. The lexical entry for suffixes of regular common nouns like “president” is shown in Figure 5.21. The feature structure for the word “presidents”, as returned from lexical look-up, is then the one in Figure 5.22. The mechanism described so far, based on the simple concatenation of root and postfix, works fine for words with regular morphology. The treatment of words with irregular morphology relies on two mechanisms:

1. if it is possible to find a small number of different roots for the word, then such roots can be all declared in the lexicon, and can be indexed according to the features that influence the selection among them;

2. otherwise, inflected forms are directly introduced in the lexicon: it is sufficient to omit the *root* and *morph* feature, and to put the inflected word as the value of the *lex* feature. In this case, the linearizer will not perform any concatenation, and will simply produce the value of the *lex* feature.

Even in its simplicity, this concatenation-based mechanism can greatly reduce the size of the lexicon, especially for languages with complex inflection.

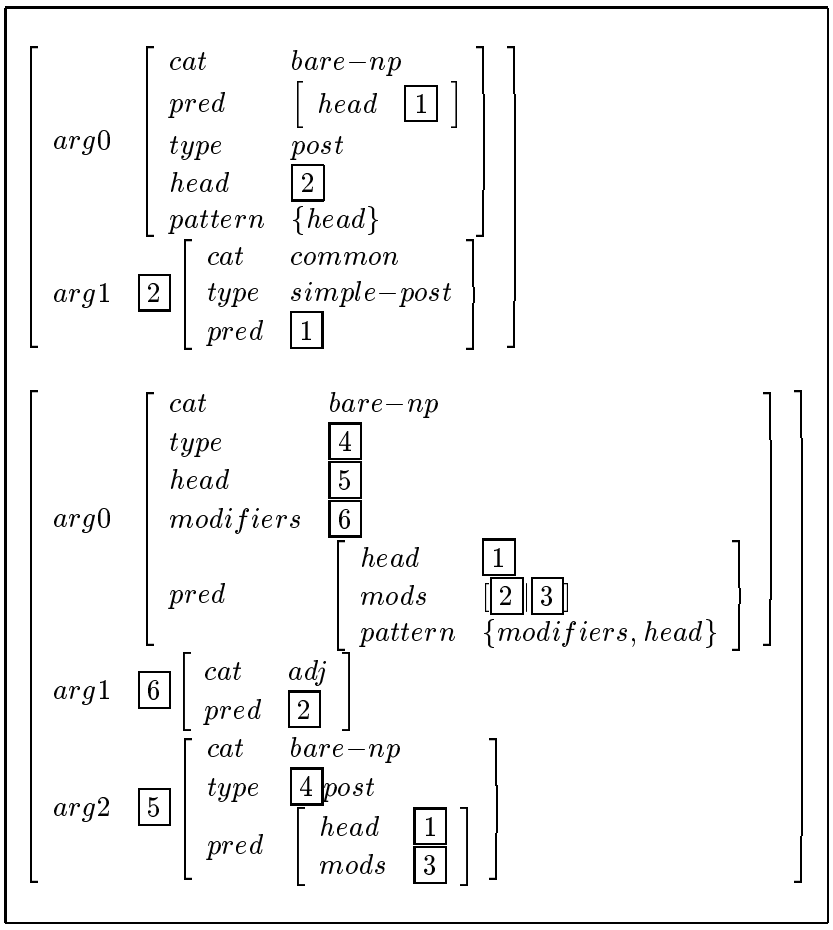


Figure 5.23: Two rules from the grammar used to parse position names.

Grammar rules are represented as feature structures too, and are very similar, in the way they work, to combination rules: feature structures belonging to the edges to be combined are plugged under appropriate *socket-features*, and if unification succeeds the result is available as the feature structure under a special *result-feature*. Two grammar rules are shown in Figure 5.23.

The first rule in the figure is meant to build an edge of category *bare-np* of semantic type *post* from an edge of category *common* and semantic type *simple-post*, such as the one built for the word "presidents". The feature structure under the *arg0* feature is the left-hand side of the rule, while that under *arg1* is its right-hand side.[15] When the rule is applied, the feature structure from the edge already in the chart is unified under *arg1*. If unification succeeds, then coreference links are resolved and a new edge is created with the feature structure under *arg0* associated with it. Suppose for instance that the chart contained an edge for the lexical entry "presidents" with the feature structure in Fig.5.22. The first grammar rule in Figure 5.23 could then be applied, and unification would yield the feature structure in Fig.5.24. The feature structure associated with the newly created edge would be that under *arg0* (Figure 5.25). This new edge, together with the lexical edge for the adjective "vice", could be combined by the second rule in Figure 5.23, and an edge covering the string "vice presidents" would be created with an appropriate feature structure. The grammar for position names is structured in such a way that the feature structure resulting from parsing a string, when linearized, returns the original string. This requires that *pattern* features,

[15] Were the right-hand side composed of more than one element, there would be features *arg1*, *arg2*, and so on.

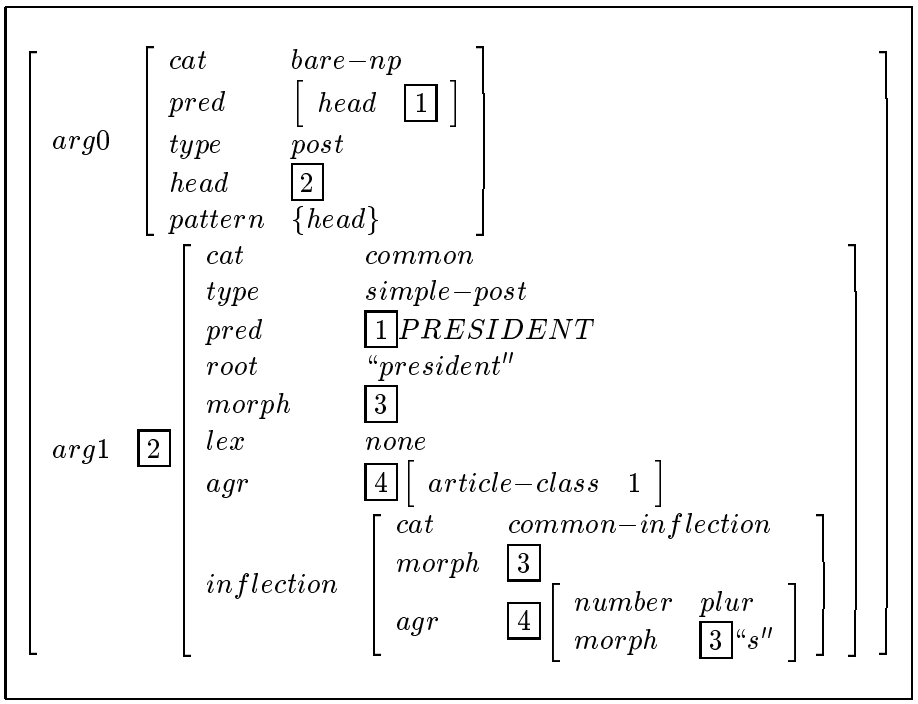


Figure 5.24: The first grammar rule in 5.23 after unification with the edge for "presidents".

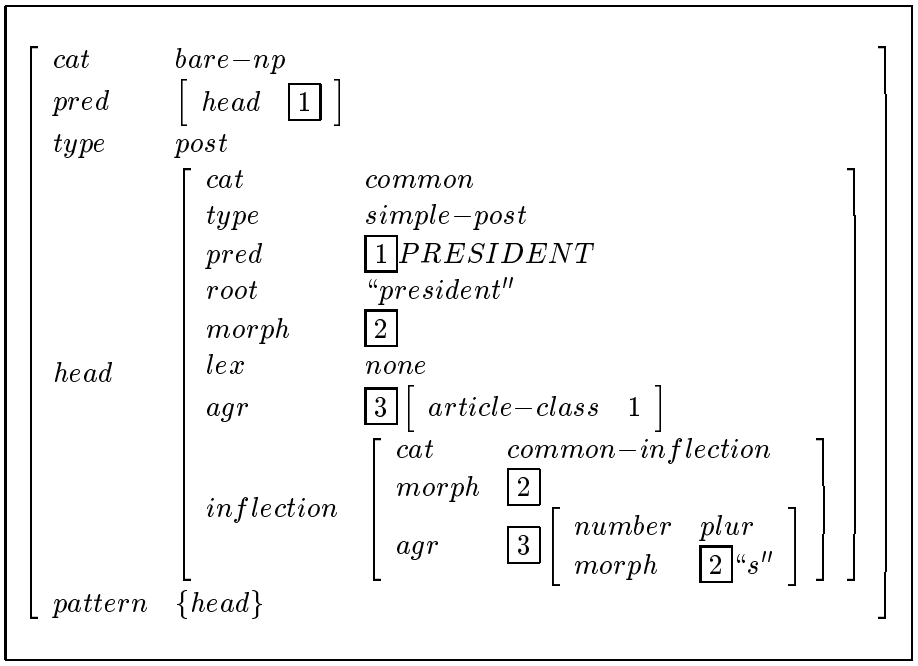


Figure 5.25: The feature structure associated to the newly created edge.

though not strictly necessary to parsing, are inserted in grammar rules with appropriate values.

Though very specialized, a lexicon covering all words potentially occurring in a position description is unlikely to be written. To be convinced of this, note that some positions include a description of a company business area ("Vice president for X"), and that there is no reasonable limit to the words that can occur in the name of a business area. This is hardly a problem, however, if we consider that, at least for the moment, parsing is required only to figure out the string's syntactic head and manipulate its agreement features, because the set of words that can appear as syntactic head of position names is, instead, quite small. Unknown words potentially occurring in position names are mostly common nouns and adjectives. The grammar has thus be adjusted to hypothesize both a noun and an adjective every time an unknown word is met. In this way ambiguity increases, and irrelevant edges are introduced in the chart. Note however that position names are never very long, and that as only inactive edges are inserted in the chart, data structure size is always manageable. The number of edges in the chart for parsing by far the most complex position name encountered,

> "executive vice president of AT&T in charge of a newly formed seven-member Global Operations Team"

is 143. Note moreover that this position was found in a key answer, written by a human operator: no position name actually extracted by FASTUS is nearly as complex. As we have access to FASTUS grammar and lexicon, we can ensure that all position names recognized by FASTUS will be covered by our specialized grammar.

As we said earlier in this Section, a linearizer more general than that offered with FUF was developed. The new linearizer behaves much like that of FUF, with the significant difference that, when a *pattern* feature is not found, *root* and *morph* are checked before *lex* and, if both are present, are concatenated and returned. *Root* and *morph* features have also an impact on the way lexical lookup is carried out. Lexical entries are indexed by means of *trees of letters*. Each arc in such a tree is labeled with a letter. When a word must be looked up in the lexicon (see Fig.5.26), the tree of letters for roots is considered first. The word is scanned left-to-right, and the corresponding path is followed in the tree. Entries for roots contained in the lexicon are linked to tree nodes at the end of the corresponding paths. When an entry is found, the feature *root* is assigned the corresponding string, and what remains is assigned to *morph*. The value of the feature $inflection$ (see Figure 5.20), with its feature *morph* now instantiated, is then unified with a lexicon of suffixes organized as a Functional Unification Grammar itself. If unification succeeds, then the resulting feature structure is returned (see the example above, for the word "presidents"). The result of the overall look-up is thus the set of all feature structures for all roots along the path in the tree of letters successfully unified with the grammar for suffixes.

Remember that string parsing is required because it can be necessary to alter agreement features to insert the string in the final text. To this purpose, a special destructive function was implemented. Such function is based on the assumption, which must be enforced by the grammar writer, that the syntactic head of the string may be reached, in the parse feature structure, by descending along *head* features. When there is no *head* feature, a $inflection$ feature is sought and the value of its *agr* feature is changed. If there is no $inflection$ feature, because the head is an invariant word, then the *lex* feature is sought, the value of its *agr* feature is altered, and the lexicon is looked up again for the new form. After this alteration has been carried out, the modified feature structure can be fed to the linearizer to yield a string that can be inserted into the final text with no risk of ungrammaticality.

So far we did not take into account a second slot that is filled by a textual string: that for

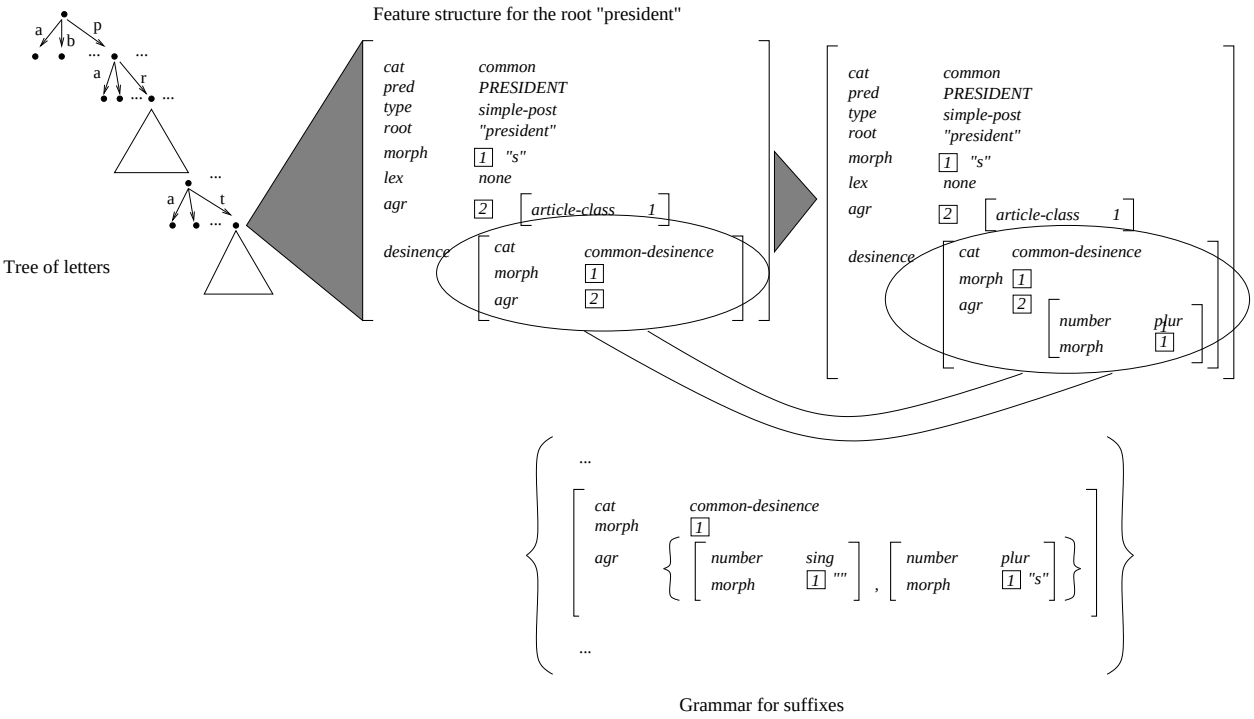


Figure 5.26: The word lookup process. Roots are searched by descending paths in a tree of letters. The value for the *inflection* feature is then unified with a grammar for suffixes to complete the feature structure of the inflected word. Unification with the grammar of suffixes is guided by the hypothesized value of the *morph* feature.

organization descriptions. Remind from Section 2.3 that the ORG_DESCRIPTOR slot of ORGANIZATION objects must be filled with:

> "a noun phrase describing or referring to the organization without naming it." [DARPA, 1995]

This slot is not queried by implemented interrogation rules, consequently its filler does not appear in generated summaries. Note however that inserting such a filler as an appositive to the first occurrence of the organization name it is related to would not pose any obstacle: it could be plugged in with no change. Multi-sentential combination rules, however, should be available to ensure that only the first occurrence is expanded with the description, and this is indeed the reason why organization descriptors are not currently considered by the implementation.

The fact that textual strings are parsed as part of the generation process could sound odd. If the IE system that produced the template performs some kind of linguistic analysis, then such strings are actually parsed twice: once by the IE system and once by the generator. Wouldn't it be better to store the results of the first parsing and reuse them? The answer to this question is to be found in design principles. As we said in section 5.1, reusing partial results of the Information Extraction process would amount to a tight coupling, at least if the whole system is not structured according to some standard architecture such as the TIPSTER architecture. Such tight coupling would make the generator work only in conjunction with a specific Information Extraction system. As a design choice, instead, we decided to allow only resource reuse, and not partial result reuse, so that the generation module can operate with any MUC system. Even if we decided for tight coupling, however, it is not obvious that partial results would be useful. In the case of FASTUS, for instance, the grammar for Information Extraction lacks a representation of agreement features

adequate for generation. Other systems could provide a better suitable representation, but the architecture would lose generality.

5.5 The cross-linguistic extension

So far, we assumed that our goal was a text in the same language as the source text. In Section 3.3, however, we mentioned cross-linguistic summarization as one of the advantages of IE+G over sentence extraction. Situations in which there is text potentially containing relevant information written in a language unknown to the user are not difficult to imagine, ranging from financial analysis to navigation on the World Wide Web. Professional translators are expensive, so an application that could extract information in a language unknown to the user and re-generate in a known language could be valuable, at least as a filter to decide what should be actually translated manually. The architecture proposed in this thesis is well suited for such a task. To the best of my knowledge, there is currently no other implemented prototype capable of approaching this task.

Ideally, one would like to have resources for generation as language-independent as possible, so that most of them could be used for generating in more than a language. Unfortunately, this requirement clashes against the decision of maintaining resource distribution close to that of FASTUS: language-specific properties appear as soon as FDs for basic sentences are generated, in the knowledge base interrogation phase. As a consequence, most resources have to be rewritten for any new language.

The initial representation, the MUC template, appears, at a first glance, strongly biased towards English. Object names, slot names, fillers: everything is in English. This first impression, however, is misleading, because in most cases English words are, at least from the point of view of our architecture, simply labels that could be words of any other language, or even just symbolic names. This is true for all words that do not directly appear in the final text as such: object names, slot names and fillers from predefined, limited sets. Only exceptions are textual slot fillers, because they must be introduced in the final text, possibly after some manipulation.

Note that not all textual fillers are actually a problem: person names, for instance, are not, because they don't need any translation. Translating some others, as person titles ("Ms.", "Mr.", "Dr.", etc.), is trivial. Organization names [Kameyama, 1997] and names of geographic areas are more delicate, because sometimes they can be left as such, but in general they require a translation. In both cases translation seems quite idiosyncratic, and it is not easy to imagine anything different from dealing with them as lexical items to be looked up in a dictionary. String fillers composed of arbitrary text, however, pose a serious problem of Machine Translation (MT). To focus attention, let's consider two such slots in the management succession domain: that for position names (slot POST of SUCCESSION_EVENT objects) and that for description of organizations (slot ORG_DESCRIPTOR of ORGANIZATION objects).

We already considered position names in the previous section, where the basic architecture has been extended to ensure agreement enforcement. Position names must necessarily be translated if the language of the final text and that of the source text are different. Two issues must be considered. The first is common to all MT applications: there may be more than one position name in the target language for a single position name in the source language. An obstacle, in this case, is that the string to be translated is deprived of all its context, so that disambiguation is virtually impossible. There is no easy solution to this problem, the most cautious approach consisting in selecting, in such cases, a target position name as generic as possible. The second issue is that the grammar for position names cannot be supposed to be complete. For string inflection purposes this was not a problem, because only the syntactic head had to be determined reliably, the rest of the name remaining untouched in the final text. If the string must be translated a complete parse is in principle necessary. To this purpose, the parser described in the previous

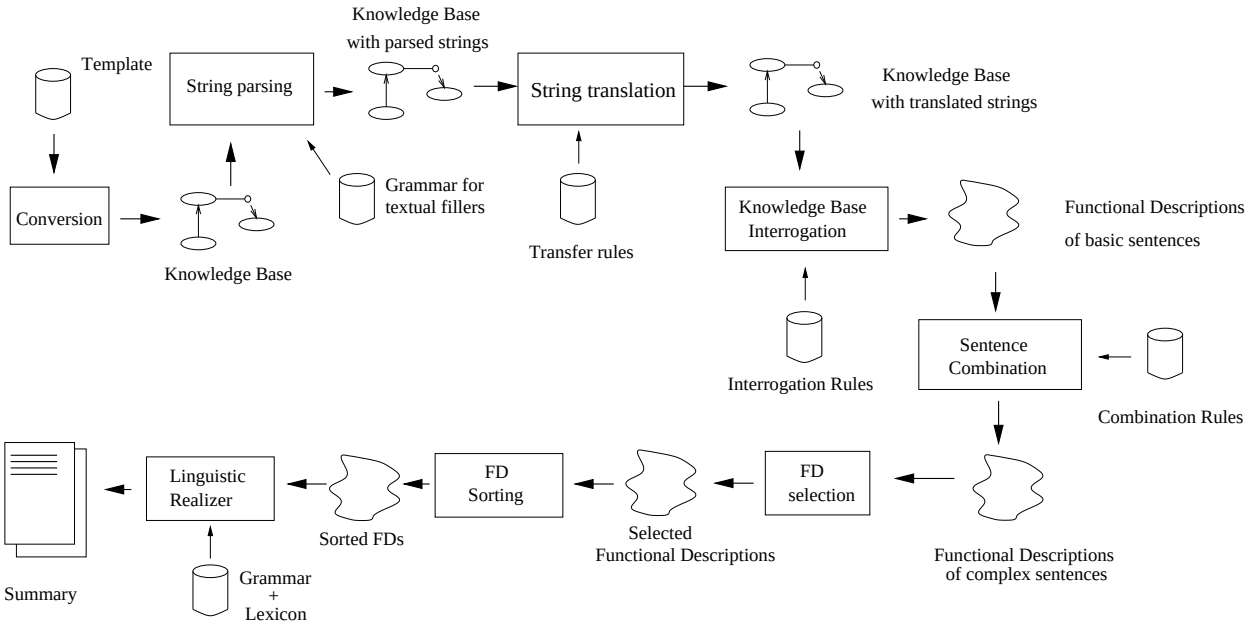


Figure 5.27: The architecture modified to accommodate the translation of textual fillers.

section can be employed. Whenever the whole string cannot be parsed, this parser is designed to return the longest edge of the top-level grammatical category in the chart. Usually this edge corresponds to the position name with some missing specifier: translating it instead of the whole string is generally an adequate solution.

Organization descriptors pose even harder obstacles to translation: apart from being constrained to noun phrases, they can contain virtually anything. Moreover, they differ from position names in that, though a grammar to parse at least the head noun is feasible, such a grammar would be of little use, because what is often really relevant in a descriptor is its specifiers. For instance, in descriptors such as:

> "a unit of London's Thorn EMI PLC"
> "a privately held cosmetics company"

the syntactic head ("a unit", "a company") hardly adds any information, and would sound very unnatural if inserted as an appositive phrase. As the idea of writing extensive linguistic resources that could provide a reasonable coverage clashes against the design principles underlying the architecture, and as organization descriptors, unlike position names, are not strictly necessary in a summary, they were left aside.

Concerning when to translate textual strings in the course of generation, considerations analogous to those made for string parsing hold. On one side, translating them just before linguistic realization could avoid some waste, on the other it would not make the parses available to sentence combination. As in the case of string parsing, then, and for the same reasons, string translation can conveniently be done right after parsing, before knowledge-base interrogation takes place. The modified architecture is schematically depicted in Figure 5.27.

A module for translating position names from English to Italian was implemented for this thesis.

Among all possible approaches to Machine Translation, a transfer grammar seemed preferable because of its simplicity. As the grammar for position names is relatively simple, the number of transfer rules is small. Moreover, if an approach based on an interlingua were preferred instead, a grammar for generating Italian position names should be written, whereas transfer rules can be

designed so that the transformed parse can be fed to our linearizer to yield the translated string. Transfer rules are applied to the feature structure resulting from string parsing, and produce a modified feature structure that can be directly linearized. Rules are organized in an ordered set. They are first checked, one after the other, on the top level feature structure. Each rule imposes conditions on the feature structure it is applied to: if this conditions are met, then the top level feature structure is altered, otherwise the next rule in the set is checked. If a rule is applied successfully, then the value of the feature *pattern* at the current level is extracted, and the transformation algorithm is applied recursively to all the corresponding values. If no *pattern* feature can be found, then the leaves of the parse tree have been reached and recursion ends. Also, if no rule matches the current feature structure, recursion ends, this time with a portion of the source string still not translated. The implemented algorithm offers then the choice whether to leave the feature structure as it is, so that part of the original string appears in the target string, or to delete the untranslated part of the feature structure, so that that part will be missing.

Transfer rules were implemented relying on the usual logic-programming-like mechanism of special feature structures with a *socket-feature* —where the source feature structure is plugged— and a *result-feature* —from where the transformed feature structure is extracted. As an example, consider the feature structure in Figure 5.28: it is obtained from the string "executive vice president" by applying the parsing procedure described in the previous section.

The position of "executive vice president" can be translated into the Italian "vice presidente esecutivo", where the adjective "vice" is still a pre-modifier, whereas "executive", that becomes "esecutivo", must be moved after the head. The first rule to apply, shown in Figure 5.29 carries out exactly this transformation. This rule matches a node of *cat bare-np* modified by an adjective whose semantics is either the atom SENIOR or the atom EXECUTIVE. The feature structure is left as is, but the relative order of the modifier and the head is set to be {*head,modifiers*}. Once this rule has been applied, the translation algorithm recurs on the values for the *head* feature and for the *modifiers* feature. The latter matches with the lexical transfer rule in Figure 5.30. This lexical rules maps the value *executive* for the feature *pred* into the root *"esecutiv"*, and the value *senior* into the root *"anzian"*. Moreover, the value of the *morph* feature is conditioned on agreement features. In our example, an appropriate feature structure to be linearized as "esecutivo" is prepared. The value of the head feature, instead, matches a translation rule for bare NPs pre-modified by adjectives that must stay on the left of the head noun also in the output sentence. This rule simply recognizes the adjective and allows for further recursion, on both the modifier[16] and on the head. Another simple recursion-enabling rule matches any bare NP whose head is of *cat common*. Eventually a lexical transfer rule for the word "president" is reached, and the feature structure for the Italian "presidente" is built. The final result is shown in Figure 5.31. Note that it can be directly fed to the linearizer, that produces the string "vice presidente esecutivo", as desired.

As it was mentioned at the beginning of this chapter, all other linguistic resources for Italian had to be rewritten. These resources include:

- interrogation rules, both queries and basic FD schemata;
- combination rules;
- generation grammar;
- lexicon.

For some of them, the corresponding resources for English were a useful starting point. Note that in this case there was no FASTUS resource to "copy" from, thus resource development was

[16]The word "vice" exists in Italian with exactly the same meaning as in English, though with a different pronunciation. It remains unmodified regardless of grammatical number and gender.

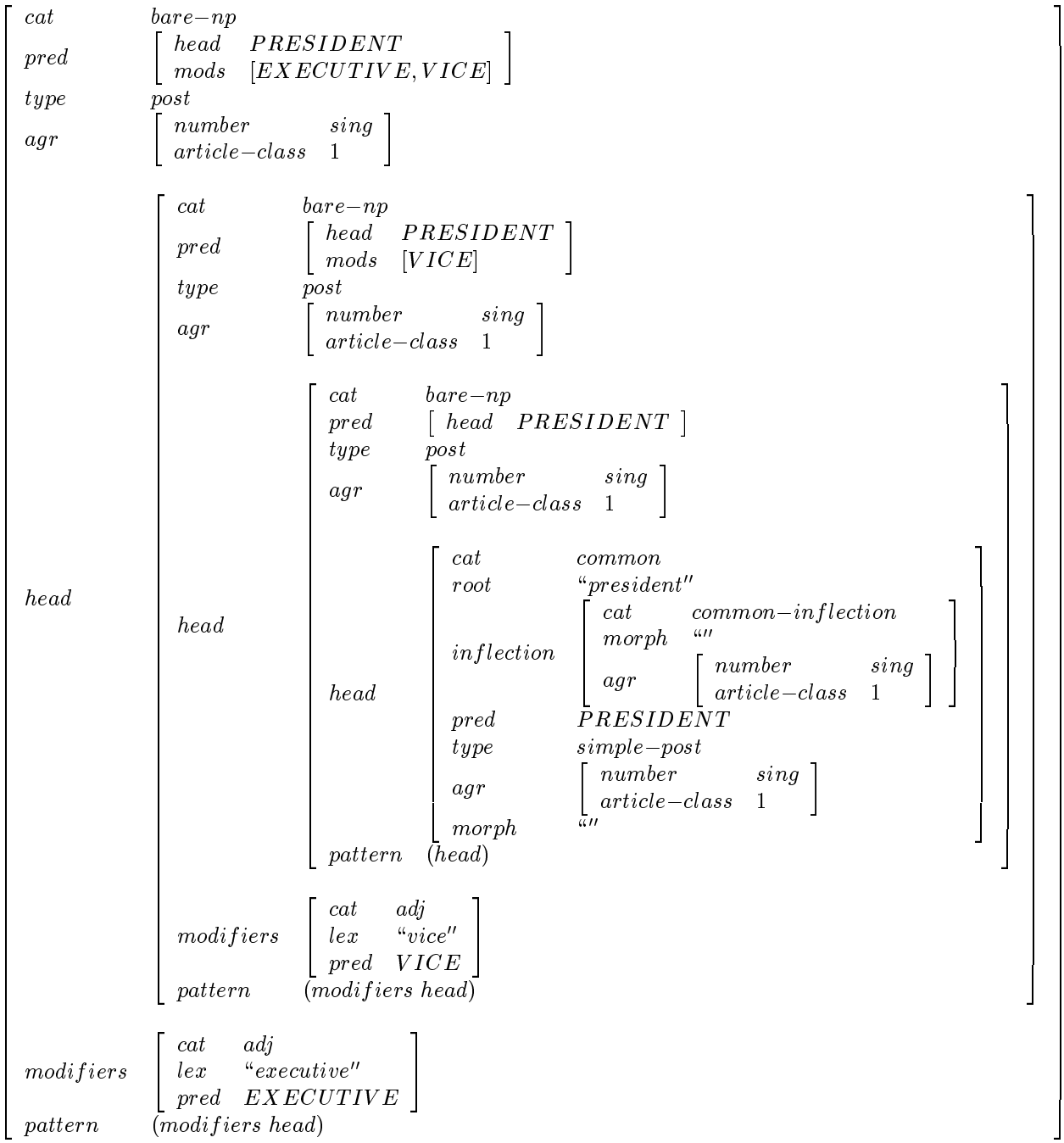


Figure 5.28: The parse tree for the string "executive vice president".

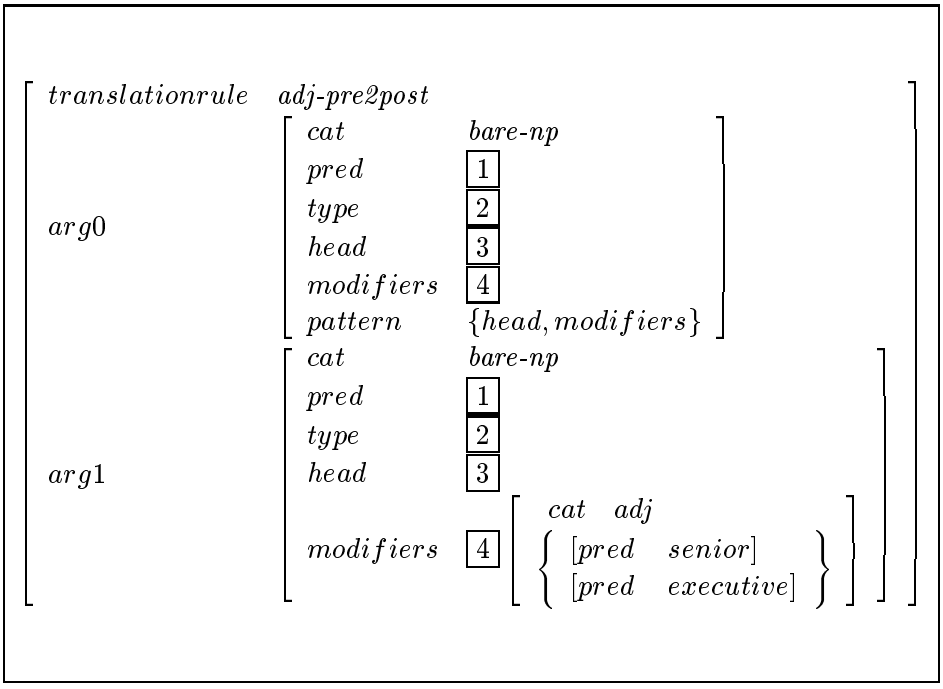


Figure 5.29: A transfer rule for translating adjectives "senior" and "executive" and moving them past the head noun.

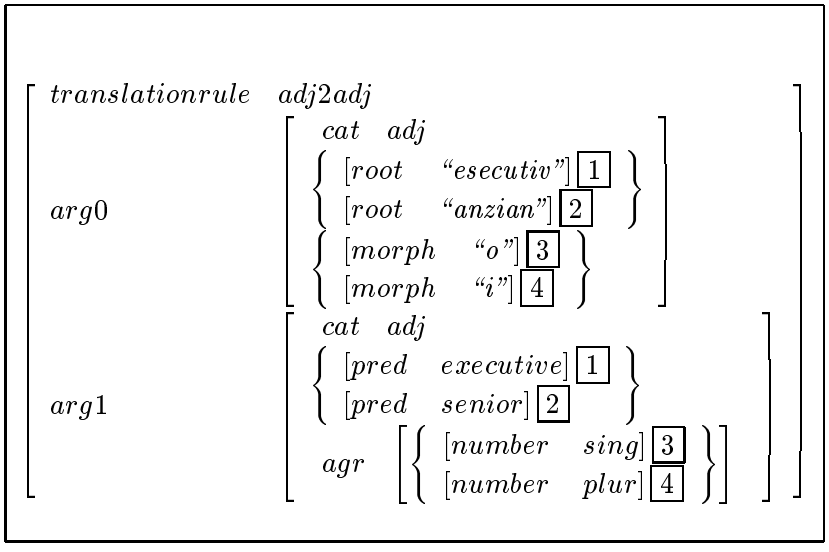


Figure 5.30: A lexical transfer rule. Note that FUF allows to constraint the selection in one alternative to that in another alternative in the same feature structure. Choice that must be selected together are marked by means of boxed number positioned *after* a feature structure, instead as before as usual for coreference links.

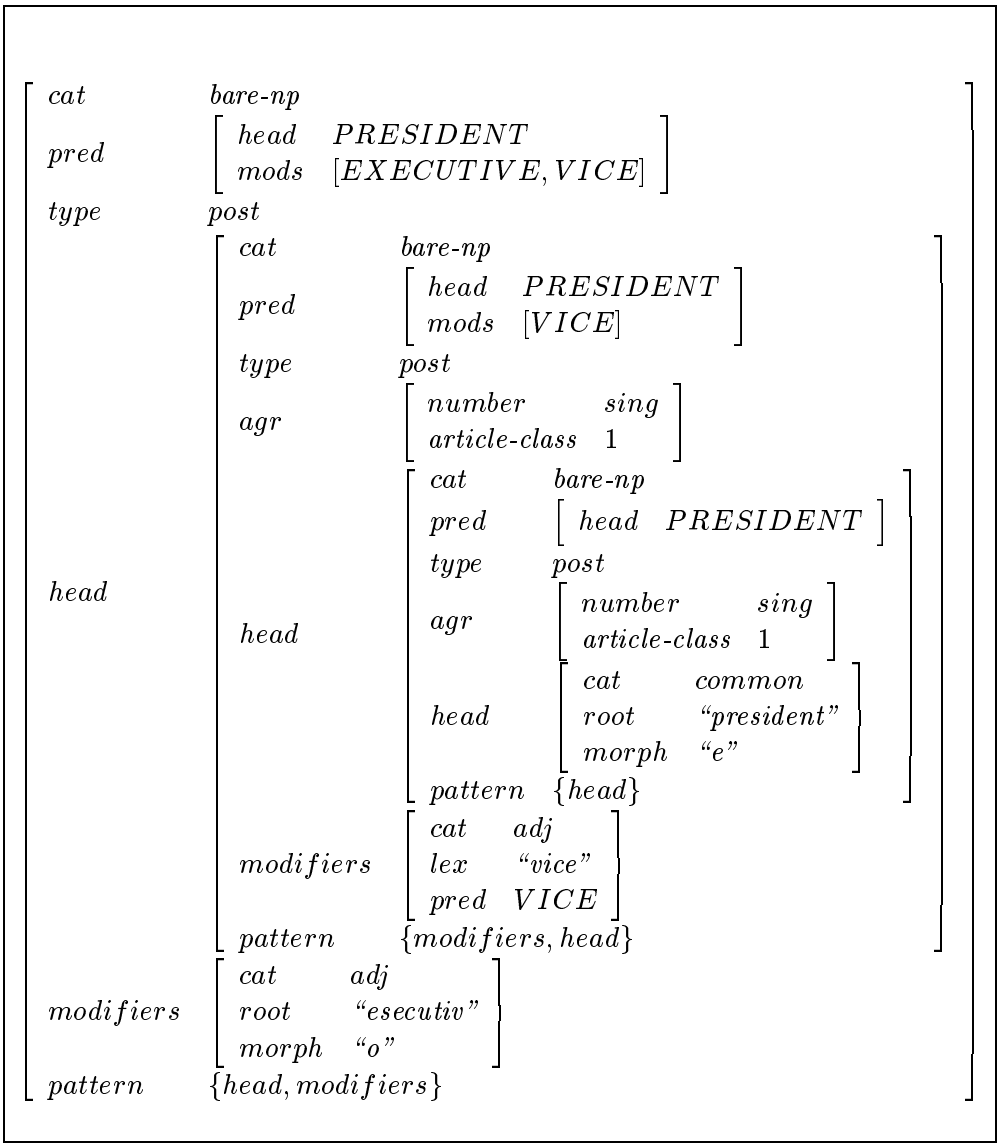


Figure 5.31: The final FD for the Italian position name "vice presidente esecutivo", as it is obtained following the recursive translation algorithm.

considerably slower. Even if developed resources were narrow in coverage, they were sufficient to be convinced of the feasibility of the approach. Broader resources were left for future development.

5.6 Conclusions

This chapter presented the principal contribution of this thesis: a knowledge-based architecture that defines in detail how to transform an instantiated MUC template into a concise and fluent sequence of sentences, and what is required to do it. This architecture differs from all known prototypes addressing the same task in that it is conceived from the beginning to be easily portable to new domains and applications. Such agility is a fundamental feature for a system that must operate in conjunction with an Information Extraction system, and it is accomplished on one hand by reusing resources developed for FASTUS, and on the other by limiting the necessity of sophisticated semantic models of the domain: a simple model derivable from the BNF description of the template itself is sufficient.

Even if the semantic representation is not particularly rich, however, its availability allows tasks to be undertaken that would have been impossible without it. The one among them that appeared most interesting, namely cross-linguistic generation, was experimented and with encouraging results.

The architecture was validated by means of the proof-of-concept system GeM, thoroughly described in this chapter as well, that produces texts in the MUC-6 management succession domain.

This success encourages to believe that a knowledge-based approach to text summarization can be convenient, and is possible also within applicative constraints that are realistic for most applicative scenarios for which Information Extraction was conceived.

Chapter 6

Evaluation

In introducing the proposed architecture for text generation from MUC templates we justified design decisions in terms of desiderata. As the architecture has been implemented, it is possible to verify which desiderata have been achieved by experimenting with it. The present chapter is devoted to discussing the extent to which GeM fulfills its requirements. Whenever reasonable, this discussion is based on quantitative experimental data.

6.1 The state of GeM implementation

Evaluating an implementation of an architecture is not exactly the same as evaluating the architecture itself: on one hand the implementation can show limitations that are due to the way the architecture was transposed, rather than to the architecture itself, while on the other it could hide deficiencies that would show up in a different context. Experimenting with an implementation, nonetheless, can cast some light on many design choices. With these considerations in mind, it is important to specify accurately what the state of the implementation being evaluated is.

Almost all aspects of the basic architecture are implemented in GeM. The main limitation, with respect to what was said in the previous chapter, consists in the reduced number of template slots actually taken into account. Of all object slots in the management succession domain, only some were actually considered in writing interrogation rules and, consequently, Functional Descriptions of basic sentences. Of the remaining slots, some were deliberately omitted for reasons that will be explained shortly, while others were simply temporarily left aside for future inclusion. Slots not included are:

- person titles;
- person aliases;
- organization aliases;
- organization descriptions;
- organization types;
- organization geographic locations;
- a classification of the reasons for positions being vacated;
- organizations, possibly different from those directly involved in the succession, the top manager comes from or is bound to;

- a classification of the relation holding between the organization affected by the succession event and the organization mentioned in the item immediately above.

Person titles and person aliases were left aside, as well as organization aliases, descriptions and geographic location, because they are somewhat related to *repeated* referring expressions. In naturally occurring texts, for instance, the first time a person is mentioned, it is usually by means of her full name, while subsequent references can be done by a combination of title and family name —the most frequent among aliases— or by means of other aliases. Similar considerations hold for organizations: the first time they tend to be referred to through their full name, possibly integrated with an appositive phrase corresponding to the ORG_DESCRIPTOR slot, whereas subsequent references are by means of aliases. To reproduce this behavior, that is, to distinguish between the first reference to an object and subsequent references, it must be possible either to constrain, at knowledge base interrogation time, the order in which sentences are realized, or to operate on the intermediate representation of sentences after their relative position have been set and before linguistic realization occurs. The issue of combination rules imposing inter-sentential constraints was discussed more in detail in Section 5.3.3. Such rules are the first task in the agenda for future work.

Allowed values for the ORGANIZATION_TYPE slot are: GOVERNMENT, COMPANY and OTHER. Such a coarse-grained classification makes sense if the intended goal consists in populating a database. After all, it would be perfectly reasonable for a user to constrain a query to one of the above categories. If a textual summary is the goal, however, the ORGANIZATION_TYPE slot is hardly of any help. Usually the fact that an organization is a private company or a government branch is immediately evident by its name, or by the name of a mentioned position within it: the reader can infer, from the fact that the post of "chief operating officer" is the object of the succession, that the succession itself takes place in a private company. Moreover, if this is not the case, then the Information Extraction system itself is likely to be puzzled, and the filled slot can be unreliable.

The VACANCY_REASON slot can have one of four values: DEPART_WORKFORCE, REASSIGNMENT, NEW_POST_CREATED and OTH_UNK. The first value is very rare, both in key answers and in real output. The choice between REASSIGNMENT and OTH_UNK, by far the two most frequent values, is presumably based on the presence, in the same article, of other succession events concerning another position being acquired by a person, and not on linguistic clues. In other terms, it is based on inferences the IE system draws from the whole set of succession events reported, and not on the structure of sentences or on specific lexical items. Such inferences are easily drawn by the reader of the final text as well, so it seemed unnecessary to explicitly take them into account.

Slots for other organizations top managers come from or are bound to seemed somewhat less central to the topic, and were left for future inclusion.

Apart for these missing slots, all the rest of the basic architecture was implemented and tested on the set used in formal MUC-6 evaluation, composed of 100 texts.

The rest of the chapter is divided in two main parts: the first explains what features are being evaluated and how, the second presents the actual results of the evaluation.

6.2 Evaluation methods

Text generation from MUC templates is a new task for which no evaluation method has been previously proposed. Evaluation methods proper of summarization systems, such as those presented in Section 3.1.5, are not valid candidates for several reasons. Intrinsic methods aiming at confronting subsets of sentences selected from the source text by an automatic program and by a human judge

are obviously inappropriate, being conceived and meaningful only for sentence extraction systems. Extrinsic methods, aiming at assessing the utility of summaries in completing other tasks, would tend to measure the performances of the IE system producing the MUC template more than those of the generation system built on top of it. Natural Language Generation systems, on the other side, are so different from each other in the task they are designed for, that no formal, quantitative measure has been successfully proposed.

To evaluate our architecture, then, new methods had to be devised. Remind from Section 4.2 that we defined seven desiderata for a text generation system to be coupled to an IE system. They are:

1. **Fidelity**: the accuracy in reporting information contained in the template;
2. **Completeness**: the extent to which information in the template is present in the generated summary;
3. **Correctness**: the grammaticality of the produced text;
4. **Conciseness**: the space necessary to convey in a text the information from the template;
5. **Readability**: text fluency and ease of comprehension on behalf of the reader;
6. **Portability to new applications**: the ease of porting the generation system to different domains, or to different tasks within the same domain;
7. **Efficiency**: computational resources —time and memory— required by the generation system.

The rest of this section is divided into two parts: the first part concerns five of these desiderata for which experiments were designed and a quantitative evaluation was carried out. The second part contains a qualitative discussion of Gem's behavior with respect to portability and readability.

6.2.1 Quantitative evaluation

In order to assess fidelity and completeness one could devise the following experimental protocol (See Figure 6.1): generated summaries are given to a human judge, not aware of the way summaries are generated, who manually fills templates based only on their content, that is, without accessing the original texts. By scoring her templates against the answer keys filled from the original texts it would then be possible to verify:

1. if information in the template is accurately presented. This could be induced by the *precision* score;
2. the extent to which information in the template is reported in the text, that would be directly measured by the *recall* score.

In other words, we would have a direct, numerical measure of *fidelity* and *completeness*. Unfortunately a human judge unaware of the way summaries were produced was not available.

As an alternative to a human judge, an Information Extraction system was available, and was used to fill templates from summaries: FASTUS.

The procedure followed to gather data for a quantitative evaluation was thus the following:

1. 100 summaries were generated from the answer keys of the official MUC-6 evaluation;

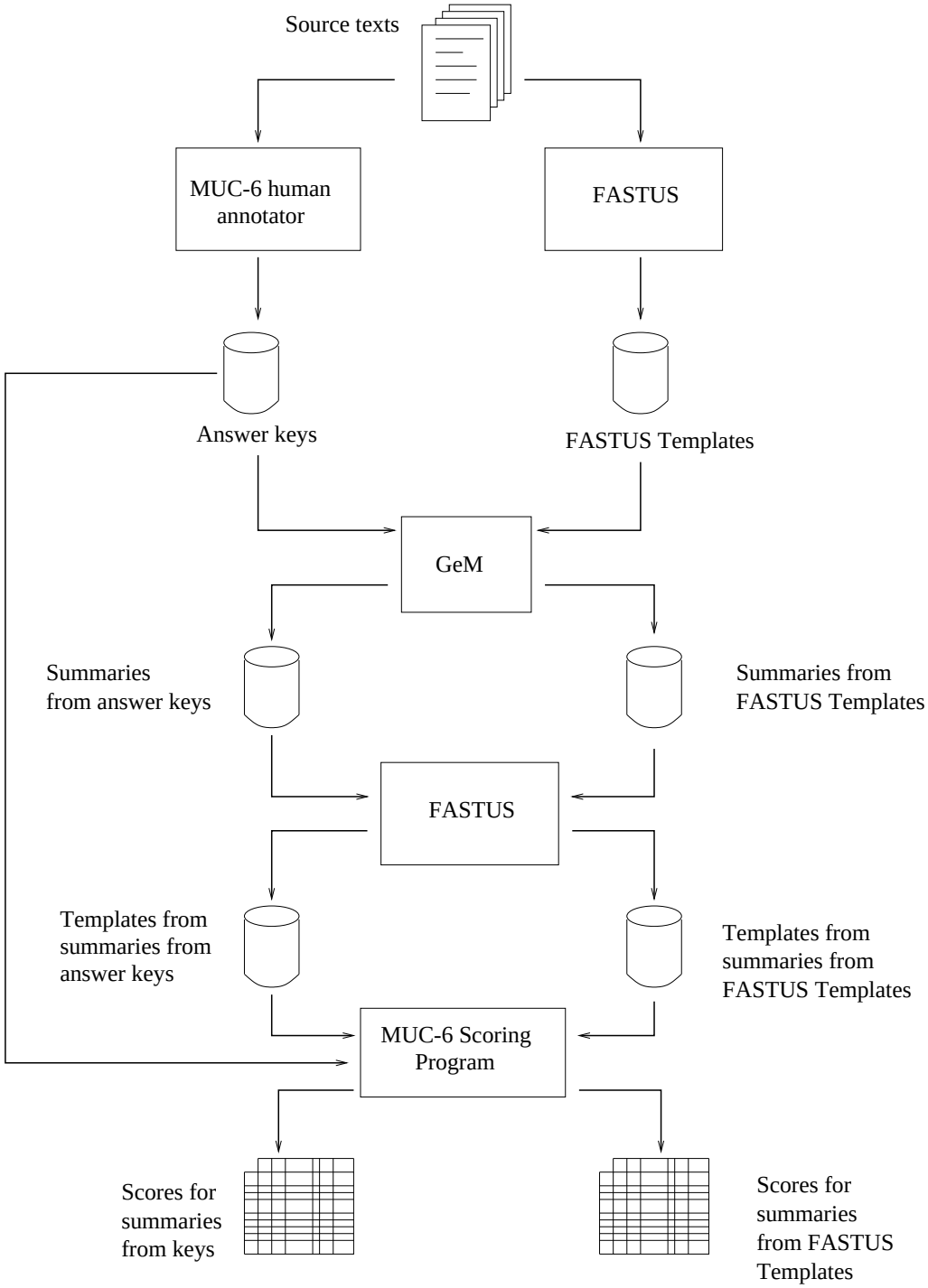


Figure 6.1: The recycling procedure for quantitative evaluation of *fidelity* and *completeness*.

2. another 100 summaries were generated from the templates produced by FASTUS from the same texts;

3. both batches were given as input to FASTUS, and two sets of filled templates were collected;

4. the two batches were scored against answer keys using the official scoring program.

Using an Information Extraction system instead of a human judge has the advantage that measures can be produced very quickly, and at virtually no cost, but have also a few disadvantages. The first disadvantage is that, because of its more limited inference capabilities, the IE system is likely to be much more sensitive to ungrammaticality than a human judge. As a consequence, a low correctness will affect *precision* and *recall* scores, that will cease to be pure measures of *fidelity* and *completeness* respectively. This means that, when considering precision and recall scores, it will be necessary to discriminate between errors caused by information missing or misleadingly conveyed by the text on one side, and errors caused by ungrammaticality on the other. A second disadvantage is that FASTUS has its weaknesses, too, and that such weaknesses will combine with those proper of GeM in determining the scores. Sources of errors will then have to be analyzed to figure out whether they depend on actual deficiencies in the generation process or they should be imputed to FASTUS.

Some quantitative evaluation of conciseness is possible, too. The degree of compression achieved by the summarization system made of an IE system and GeM can be measured in terms of the average ratio between the length of summaries and the length of the corresponding original text. Length has been measured in terms of number of words and number of sentences. The degree of compression was measured both for summaries generated from answer keys (in this case the summarization system is somewhat "semi-automatic") and for summaries generated from FASTUS output.

6.2.2 Qualitative evaluation

Experimental data collected with the procedures described above will be useful for assessing some of our desiderata, but others still fall out.

Readability will be discussed using the table of fluency defects proposed in [Kukich, 1991] and presented, slightly modified, in Figure 4.1, as a guideline. Portability to new domains and tasks will be discussed informally.

6.3 Results

In the previous Section our methods for evaluating GeM with respect to posted desiderata were introduced. The present Section is devoted to presenting, and discussing, the results of such evaluation. Results of quantitative evaluations will be presented first. Readability and portability are the object of Section 6.3.2.

6.3.1 Results of the quantitative evaluation

Batches of 100 summaries each generated by GeM were given as input to FASTUS, and the templates thus obtained were compared to those manually filled by official MUC-6 judges. Two such batches were produced: the first from MUC-6 answer keys, the second from actual FASTUS output on the same texts.

Summaries generated from answer keys are better suited to assess GeM's behavior in ideal condition, that is, if it could be coupled to a perfect (or nearly perfect) Information Extraction

	POS	ACT	COR	PAR	INC	MIS	SPU	NON	REC	PRE	UND	OVG	SUB	ERR
OBJ SCORES														
organization	104	56	54	0	1	49	1	16	52	96	47	2	2	49
person	126	102	101	0	1	24	0	11	80	99	19	0	1	20
in_and_out	256	185	178	0	1	77	6	20	70	96	30	3	1	32
succession_e	191	130	130	0	0	61	0	17	68	100	32	0	0	32
template	100	100	48	0	52	0	0	0	48	48	0	0	52	52
SLOT SCORES														
organization														
name	102	56	50	0	5	47	1	16	49	89	46	2	9	51
alias	60	2	1	0	0	59	1	21	2	50	98	50	0	98
descriptor	58	0	0	0	0	58	0	56	0	0	100	0	0	100
type	104	56	54	0	1	49	1	18	52	96	47	2	2	49
locale	40	0	0	0	0	40	0	9	0	0	100	0	0	100
country	40	0	0	0	0	40	0	6	0	0	100	0	0	100
comment	0	0	0	0	0	0	0	15	0	0	0	0	0	0
person														
name	126	102	101	0	1	24	0	11	80	99	19	0	1	20
alias	80	0	0	0	0	80	0	8	0	0	100	0	0	100
title	76	0	0	0	0	76	0	8	0	0	100	0	0	100
comment	0	0	0	0	0	0	0	1	0	0	0	0	0	0
in_and_out														
io_person	254	183	160	0	17	77	6	22	63	87	30	3	10	38
new_status	256	185	167	0	12	77	6	20	65	90	30	3	7	36
on_the_job	256	185	110	0	69	77	6	73	43	59	30	3	39	58
other_org	170	68	35	0	20	115	13	45	21	51	68	19	36	81
rel_oth_org	176	71	38	0	20	118	13	30	22	54	67	18	34	80
comment	0	0	0	0	0	0	0	399	0	0	0	0	0	0
succession_e														
success_org	185	103	86	0	17	82	0	23	46	83	44	0	17	54
post	191	130	110	0	20	61	0	39	58	85	32	0	15	42
in_and_out	252	183	153	0	11	88	19	25	61	84	35	10	7	44
vac_reason	191	130	65	0	65	61	0	40	34	50	32	0	50	66
comment	0	0	0	0	0	0	0	273	0	0	0	0	0	0
template														
doc-nr	0	0	0	0	0	0	0	100	0	0	0	0	0	0
content	191	130	130	0	0	61	0	55	68	100	32	0	0	32
comment	0	0	0	0	0	0	0	15	0	0	0	0	0	0
ALL SLOTS	2808	1593	1260	0	258	1290	75	1328	45	79	46	5	17	56
												P&R	2P&R	P&2R
F-MEASURES												57.26	68.63	49.12

Figure 6.2: The score summary table output for templates from summaries produced from answer keys.

system. Summaries generated from actual FASTUS output, however, give a better indication of what could be expected from such an architecture given the current state-of-the-art in IE. We will present results for the first group of summaries first, and results for summaries from FASTUS output right after.

Texts generated from answer keys

Results presented in this paragraph concern data collected according to the procedure on the left side in Figure 6.1. The score summary table output by the MUC-6 scoring program is presented in Figure 6.2.

The table is divided in three parts, from top to bottom. The first part, just below column headings, concerns scores on recovered template objects, while the middle part reports scores on individual object slots, grouped by the object type they belong to. The bottom part contains the totals of each column —or the average for percentages— and some synthetic measure (the F-MEASURES).

Columns should be interpreted as follows:

- **POS** contains the number of possible objects/fillers that should be retrieved by the IE system, i.e., the number of fills in the answer key plus any optional fills allowed by the key and generated by the system (equals **COR** + **INC** + **MIS**);
- **ACT** contains the number of objects/fillers that were actually generated —correctly or incorrectly— by the IE system (**COR** + **INC** + **SPU**);
- **COR** lists the number of correctly retrieved objects/fillers;

- **PAR** lists the number of partially correct objects/fillers among all those that were retrieved (no partial credit was given to these fillers in MUC-6);
- **INC** lists the number of incorrectly retrieved object/fillers;
- **MIS** is the number of missing objects/fillers, i.e., fillers in the key not generated by the IE system;
- **SPU** is the number of spurious objects/fillers, i.e., fillers generated by the IE system and absent from the key;
- **NON** is the number of non-committal (null fills generated by the system that were also null in the answer key);
- **REC** is the *recall* value on the specific object/slot (**COR**/**POS**);
- **PRE** is the *precision* value on the specific object/slot (**COR**/**ACT**);
- **UND** lists *undergeneration* (**MIS**/**POS**);
- **OVG** lists *overgeneration* (**SPU**/**ACT**);
- **SUB** lists the number of substitutions (**INC**/(**COR**+**INC**));
- **ERR** lists the *error rate* ((**INC**+**SPU**+**MIS**)/(**COR**+**INC**+**SPU**+**MIS**));

F-measures are a kind of weighted average of *precision* and *recall*. The parametric definition is as follows:

$$F(\beta) = \frac{(\beta^2 + 1)PR}{\beta^2 P + R} \tag{6.1}$$

where P and R stand for *precision* and *recall* respectively. For $\beta = 1$ (corresponding to the P&R score in the table) *precision* and *recall* have the same weight in the final score, whereas for $\beta^2 = 2$ *recall* has a weight double with respect to *precision* (P&2R) and for $\beta^2 = 0.5$ the situation is inverted (2P&R). The last line in the table displays a P&R F-measure of 57.26%. This score results from a 79% *precision* and a 45% *recall*.

Note that all synthetic measures (in the bottom portion of the table) refer to the complete set of slots, including those not considered by GeM interrogation rules, and whose content is hence not included in the final text. Most of these slots have 0% *recall* and *precision*, so overall scores are considerably affected. This is especially true of DESCRIPTOR, ALIAS, LOCALE and COUNTRY slots of ORGANIZATION objects, and of ALIAS and TITLE slots of PERSON objects, all of which have an assessed error rate of 100% (98% for ORGANIZATION ALIASes): when combination rules with inter-sentential constraints will be implemented, and it will hence be possible to insert this additional information without unnatural repetitions, part of this error will be absorbed. To have an idea of what results can be expected when all slots are considered, synthetic indices were computed not taking slots ignored by GeM into account. The bottom part of the table is shown in Figure 6.3. Both *precision* (+7%) and *recall* (+13%) results increase, consequently P&R F-measure grows from 57.26% to 69.27% (+12%). These results are definitely encouraging.

Interestingly FASTUS was able, in some cases, to infer values for OTHER_ORG and REL_OTH_ORG slots of IN_AND_OUT objects even if they were not considered by GeM. Slightly more than 20% of all fillers for these two slots were correctly recalled, and, of all guesses attempted by FASTUS, something more than one half was correct. Remaining ignored template slots were

	POS	ACT	COR	PAR	INC	MIS	SPU	NON	REC	PRE	UND	OVG	SUB	ERR
ALL SLOTS	1862	1265	1083	0	140	639	42	2308	58	86	34	3	11	43
												P&R	2P&R	P&2R
F-MEASURES												69.27	78.23	62.15

Figure 6.3: Result summary table for texts from answer keys, not considering slots ignored by GeM.

VACANCY_REASON, for SUCCESSION_EVENT objects, and TYPE, for ORGANIZATION objects. In the former case, FASTUS apparently assigned a REASSIGNMENT value every time the succession event pointed to two IN_AND_OUT objects, one for a person entering a position and another for a person leaving it. In all other cases, OTH_UNK was assigned as value. This heuristic accounted for the 34% *recall* and the 50% *precision* score for the corresponding slot. Organization TYPE was apparently always guessed to have COMPANY as value. Though crude, this decision seems quite justified, given that 99 out of 104 organization TYPE slots in the test corpus were assigned that value. This is reflected in the 96% *precision* in the table, whereas the *recall* value is roughly the same as for ORGANIZATION objects.

Precision scores for objects are extremely high: almost every time FASTUS guessed the existence of an object, that object was actually present. *Recall* scores on objects, instead, are somewhat lower than expected, ranging from 52% for ORGANIZATIONs to 80% for PERSONs.[1] From an inspection of results for individual texts it is possible to attribute a part of missing information to the fact that some generated sentences were difficult to deal with for FASTUS. From a sampling of texts it appears for instance that in most cases FASTUS misinterpreted the construction "X replaced Y as Z of W", and created for X an object of type ORGANIZATION. This is somewhat puzzling, as the misleading construction was used for generation precisely because the appropriate pattern appeared in FASTUS grammar.

Scores on slots covered by GeM's interrogation rules are easily explained given scores on objects. Names of organizations were correctly assigned in almost all and only the occasions in which the corresponding organization objects were created, and the same is true of person names as well. Scores for IO_PERSON and NEW_STATUS slots follow those for the IN_AND_OUT objects they belong to: if an IN_AND_OUT object is created, then in nine cases out of ten the person involved is recognized correctly, as well as whether he or she is entering or leaving a position. The ON_THE_JOB slot of IN_AND_OUT objects displays an undesired behavior: of all 185 times it was filled— the same number as the NEW_STATUS slot — it was filled correctly only 110 times, 59% of the cases. This lack of precision is due to the fact that the ON_THE_JOB slot is not expressed by means of ad hoc constituents, but through the choice of the verbal tense for the head verb of the sentence. When interrogation rules were designed, however, a distinction was not clearly marked in the lexical choice: that between cases in which the filler was UNCLEAR and cases in which the succession was known not to be effective at the time the article was written — that is, either NEW_STATUS was IN and ON_THE_JOB was NO, or NEW_STATUS was OUT and ON_THE_JOB was YES. The past tense was chosen for both cases, and this ambiguity is reflected in scores. A few mistakes, finally, were caused by the use of an infrequent regular form for the past of the verb "to quit"— "quitted"— instead of the usual irregular form "quit". This problem was immediately removed after it appeared in the results.

One of the greatest concern in designing combination rules was that, as they had no direct

[1] The line for objects of type TEMPLATE is irrelevant: MUC rules say that one such object must be created for each text, regardless of the fact that at least a succession event was actually found or not, and FASTUS behaves consequently. The scoring program, though, considers as an error when a TEMPLATE object is created for a text with no SUCCESSION_EVENT. This line in the table just tells that only 48 out of all 100 texts actually contained at least a succession event.

counterpart in FASTUS, they could produce sentences outside FASTUS grammar coverage. In general this concern proved excessive: in most cases, when none of the problems mentioned above occurred, sentences obtained by applying combination rules were parsed by FASTUS without any special trouble. There were some exceptions, however. The text below, for instance, turned out very difficult to FASTUS:

> Stefan Abrams was succeeded by William J. Newman as managing director and chief investment strategist of Kidder, Peabody & Co. Inc.
> William J. Newman, who was Bankers Trust Co. managing director and head of the active equity group, will be <no title> of Kidder, Peabody & Co. Inc.

Out of five succession events that were expected for this text, for a total of 7 IN_AND_OUT objects, only 1 SUCCESSION_EVENT object was instantiated, pointing to 2 IN_AND_OUT objects. None of the two organizations was extracted, and the position referred to was "chief". Note that MUC rules allow the POST slot to be filled by the string "no title" when the actual position name cannot be recovered.

An inspection of most frequent errors shows some limitations of the current implementation, but leaves us optimistic concerning the architecture. Limitations consists above all in the reduced number of slots that were considered, that impacted on *completeness*. An improved choice of lexical items and features can significantly reduce the error rate, especially the part coming from slots that are not expressed by means of whole constituents. Notice that better quantitative results in the experiments could have been achieved if no sentence combination at all were carried out: simpler sentences, though repetitive and verbose, would have been easier to FASTUS, and scores would have improved. Removing sentence combination, however, would have been detrimental for properties such as readability and conciseness.

Conciseness, one of our desiderata, can be measured through a strongly correlated index: compression ratio. The compression ratio is the ratio between the length of the summary and the length of the source text it is obtained from. For summaries automatically produced by GeM, a compression ratio was computed, both in terms of number of words and of sentences. The whole set of source texts was composed of 37,370 words, distributed in 1,589 sentences. Much of this text, however, was contained in texts for which the keys stated that no succession event were to be recognized. As the compression ratio is relevant only whenever some summary is actually produced, the subset of source texts corresponding to templates for which some summary was expected was measured: it amounted to 16,289 words distributed in 693 sentences, about 43% of the whole set. For each text in this subset GeM produced some summary, for total 1,935 words in 134 sentences.

Compression ratio for summaries generated from MUC-6 answer keys is thus 11.88% in terms of words, and 19.34% in terms of sentences. The difference between the two ratios depends on the different average length for sentences in the source texts and in the produced summaries: about 23 in the former case and 15 in the latter.

The last of our desiderata for which some quantitative data can be given is *efficiency*. Generating all 100 texts from keys required about ten minutes on a Sun UltraSPARC2 Workstation running Solaris and Harlequin Inc.'s Liquid Common Lisp. This would make for an average of six seconds per text, but as about half the templates did not contain anything to be generated, the average time for actually generating a text is somewhere between six and twelve seconds (template preliminary conversion and knowledge base interrogation, as well as the emission of an error message, are required even if the input template does not represent any succession event). This range of times is perfectly compatible with FASTUS processing time. This is of great importance, because any significantly worse result would have seriously compromised the practical utility of the generator.

	POS	ACT	COR	PAR	INC	MIS	SPU	NON	REC	PRE	UND	OVG	SUB	ERR
OBJ SCORES														
organization	106	47	41	0	0	65	6	14	39	87	61	13	0	63
person	124	71	49	0	8	67	14	13	40	69	54	20	14	64
in_and_out	248	106	87	0	0	161	19	28	35	82	65	18	0	67
succession_e	185	73	63	0	0	122	10	23	34	86	66	14	0	68
template	100	100	28	0	72	0	0	0	28	28	0	0	72	72
SLOT SCORES														
organization														
name	104	47	29	0	12	63	6	14	28	62	61	13	29	74
alias	60	0	0	0	0	60	0	21	0	0	100	0	0	100
descriptor	62	0	0	0	0	62	0	52	0	0	100	0	0	100
type	106	47	41	0	0	65	6	16	39	87	61	13	0	63
locale	40	0	0	0	0	40	0	9	0	0	100	0	0	100
country	40	0	0	0	0	40	0	6	0	0	100	0	0	100
comment	0	0	0	0	0	0	0	15	0	0	0	0	0	0
person														
name	124	71	49	0	8	67	14	13	40	69	54	20	14	64
alias	78	0	0	0	0	78	0	10	0	0	100	0	0	100
title	74	0	0	0	0	74	0	10	0	0	100	0	0	100
comment	0	0	0	0	0	0	0	1	0	0	0	0	0	0
in_and_out														
io_person	248	106	64	0	23	161	19	28	26	60	65	18	26	76
new_status	248	106	70	0	17	161	19	28	28	66	65	18	20	74
on_the_job	248	106	51	0	36	161	19	81	21	48	65	18	41	81
other_org	174	49	22	0	10	142	17	41	13	45	82	35	31	88
rel_oth_org	176	49	20	0	12	144	17	30	11	41	82	35	38	90
comment	0	0	0	0	0	0	0	399	0	0	0	0	0	0
succession_e														
success_org	185	65	44	0	13	128	8	23	24	68	69	12	23	77
post	185	73	42	0	21	122	10	45	23	58	66	14	33	78
in_and_out	246	104	63	0	10	173	31	31	26	61	70	30	14	77
vac_reason	185	73	34	0	29	122	10	46	18	47	66	14	46	83
comment	0	0	0	0	0	0	0	273	0	0	0	0	0	0
template														
doc-nr	0	0	0	0	0	0	0	100	0	0	0	0	0	0
content	185	73	63	0	0	122	10	61	34	86	66	14	0	68
comment	0	0	0	0	0	0	0	15	0	0	0	0	0	0
ALL SLOTS	2768	971	592	0	191	1985	188	1368	21	61	72	19	24	80
												P&R	2P&R	P&2R
F-MEASURES												31.67	44.50	24.58

Figure 6.4: The score summary table for templates from summaries produced from FASTUS output.

SLOT	POS	ACT	COR	PAR	INC	MIS	SPU	NON	REC	PRE	UND	OVG	ERR	SUB
ALL SLOTS	1836	753	468	0	147	1221	138	2352	25	62	67 1	8	24	76
												P&R	2P&R	P&2R
F-MEASURES												36.15	48.27	28.90

Figure 6.5: The score summary table for templates from summaries produced from FASTUS output, slots ignored by GeM are not taken into account.

Texts generated from actual FASTUS output

The scores on summaries produced from answer keys give an idea of GeM's behavior, mediated by FASTUS, in a somewhat artificial situation, in which input templates were produced by a human annotator. To gain a better insight on what the situation could be in a more realistic setting, summaries generated from FASTUS output were scored in the same way as those produced from answer keys (right side of Figure 6.1). The score summary table output by the MUC-6 scoring program is presented in Figure 6.4.

In this case, too, the effect of slots ignored by GeM can be filtered out. With scores computed without taking such slots into account, the bottom part of the table becomes that in Figure 6.5.

To correctly interpret these results, the corresponding table for FASTUS performances in the MUC-6 evaluation is reported in Figure 6.6 [DARPA, 1995].

Precision scores are systematically slightly higher when FASTUS is run on re-generated summaries than it is when it is run on source texts —with the exception of ON_THE_JOB slots, discussed above— whereas *recall* figures are consistently lower, the average dropping from 44% to 21%. The reason why *recall* decreases lies in how interrogation rules work. If a query on the knowledge base fails because even only a slot value is missing, then no FD at all is instantiated. Indeed, it is often the case for FASTUS to be able to extract information concerning mentioned entities, even

SLOT	POS	ACT	COR	PAR	INC	MIS	SPU	NON	REC	PRE	UND	OVG	ERR	SUB
template	52	52	45	0	0	7	7	41	87	87	13	13	24	0
content	182	136	115	0	1	66	20	0	63	85	36	15	43	1
succession_e	188	150	121	0	1	66	28	0	64	81	35	19	44	1
success_or	188	100	61	0	19	108	20	0	32	61	57	20	71	24
post	188	150	97	0	25	66	28	0	52	65	35	19	55	20
in_and_out	251	209	110	0	32	109	67	0	44	53	43	32	65	23
vac_reason	188	150	66	0	56	66	28	0	35	44	35	19	69	46
in_and_out	254	209	164	0	3	87	42	0	65	78	34	20	45	2
io_person	254	209	121	0	46	87	42	0	48	58	34	20	59	28
new_status	254	209	135	0	32	87	42	0	53	65	34	20	54	19
on_the_job	254	209	112	0	55	87	42	0	44	54	34	20	62	33
other_org	174	73	41	0	8	125	24	48	24	56	72	33	79	16
rel_oth_or	175	73	26	0	24	125	23	48	15	36	71	32	87	48
organization	112	74	61	0	0	51	13	0	54	82	46	18	51	0
name	109	69	38	0	19	52	12	0	35	55	48	17	69	33
alias	68	46	27	0	4	37	15	17	40	59	54	33	67	13
descriptor	65	35	10	0	14	41	11	16	15	29	63	31	87	58
type	112	74	60	0	1	51	13	0	54	81	46	18	52	2
locale	43	15	6	0	6	31	3	16	14	40	72	20	87	50
country	43	12	9	0	1	33	2	17	21	75	77	17	80	10
person	126	119	88	0	7	31	24	0	70	74	25	20	41	7
name	126	119	82	0	13	31	24	0	65	69	25	20	45	14
alias	79	77	60	0	4	15	13	20	76	78	19	17	35	6
title	75	74	64	0	0	11	10	22	85	86	15	14	25	0
ALL OBJECTS	2828	2039	1240	0	360	1228	439	204	44	61	43	22	62	23
												P&R	2P&R	P&2R
F-MEASURES												50.96	56.45	46.44

Figure 6.6: Score summary table for FASTUS at MUC-6.

if it fails in establishing all relations among entities. Consider for instance the template extracted by FASTUS from the text in Figure 6.7 reported in Figure 6.8. Apparently all the information needed to produce a sentence such as:

> "James L. McDonald will succeed Dominic A. Tarantino as chairman of Price Waterhouse"

is available. Such sentence would not be a perfect summary for the text—some succession event are not reported—but would be better than nothing. Despite that, no summary was generated from the template, because all interrogation rules required the SUCCESSION_ORG slot of the SUCCESSION_EVENT object to be filled. The scoring program assigned points to the template produced from the source text, even if the critical slot was missing. Those points, however, could not be assigned to GeM's summary. The problem of missing critical slots is the main cause of loss in *recall*. Three alternatives are open to approach this problem: the first consists in cautiously leaving things as they are. The second in adopting some form of default reasoning, and decide, for instance, that if only one organization was found — as in the example— and the SUCCESSION_ORG slot of the only SUCCESSION_EVENT is missing, than the organization can be assigned to the slot. This strategy would enhance *recall*, but would also lower *precision* because in some cases inferences of this kind could be wrong, so the trade-off must be carefully considered. The third, and probably best, solution, consists in adding sentences for fragmentary information, to use when sentences for complete events fail. This could easily be done in GeM, because FDs for fragmentary sentences would receive a lower score than FDs for sentences for complete events subsuming them when FDs are selected, so that they would appear in the summary only when appropriate. In the example above, for instance, the sentences:

> "James L. McDonald will succeed Dominic A. Tarantino as chairman of some organization."
> "Price Waterhouse was also mentioned in the article."

could be output.

Price Waterhouse said that James L. McDonald will
succeed Dominic A. Tarantino as co-chairman, effective July 1.

Mr. Tarantino, 60 years old, will become chairman on Oct. 1 of
the American Institute of Certified Public Accountants for a
one-year term. During that time he will continue as a consultant to
Price Waterhouse. It has not been determined whether Mr. Tarantino
will return to the firm full time.

Mr. McDonald, 50, is deputy vice chairman of tax for Price
Waterhouse, one of the Big Six U.S.accounting firms, and previously
was partner in charge of international tax services for the West
Coast region, a practice he established.

Figure 6.7: The text from which FASTUS produced the template in Fig.6.8

Missing critical slots, moreover, are also the main reason why scores for summaries from FASTUS output were so much lower than scores for summaries from answer keys: answer keys never miss such slots.

A last set of scores was computed comparing templates from texts generated from FASTUS output, not against answer keys but against FASTUS output itself (Figure 6.9). The scores collected in this case are in Figure 6.10. Synthetic indices when not considering ignored slots are in Figure 6.11.

This data are somewhat surprising. First of all, synthetic scores remain practically unmodified, whereas one would expect an increase as that obtained when ignored slots in answer keys were filtered out. To try to understand why, let's consider results more in detail. Results on whole object are substantially constant, as are scores on slots for ORGANIZATIONs and PERSONs. Concerning slots of IN_AND_OUT objects, however, there is some strange phenomena. Consider, for instance, the ON_THE_JOB slot. The correct number of fillers goes from 248 to 241, if FASTUS output is taken as the ground truth instead of the answer keys. Of the 106 times the slot was filled by FASTUS when running on the re-generated summaries, 51 are correct with respect to the answer keys, while only 31 are correct with respect to actual FASTUS output. In other words, when generating from FASTUS output GeM agrees more —on this specific slot— with the answer keys than with its own input. *Precision* goes from 48% to 29%, *recall* from 21% to 13%. An analogous decrease in *precision* can be observed also for OTHER_ORG and REL_OTHER_ORG slots, but in this case these slots are out of control, because they are not considered in generation. A drop in *precision* (-14%) can be observed also on SUCCESSION_ORG slots of SUCCESSION_EVENTs and, reduced (-5%), on IN_AND_OUT slots.

Compression ratio was computed for texts from FASTUS output as it was for texts from answer keys. The subset of source texts for which FASTUS produced templates from which some summary could be produced amounted to 15,893 words, in 671 sentences. The total number of sentences in the corresponding summaries was 74, containing 970 words. This makes for a 6.10% compression ratio in terms of words and a 11.03% compression ratio in terms of sentences. The reduced compression ratio with respect to summaries from answer keys is due to the reduced overall *recall*: less information

```
<TEMPLATE-9306240111-> :=
        DOC_NR:             "9306240111"
        CONTENT:            <SUCCESSION_EVENT-9306240111-1>

<SUCCESSION_EVENT-9306240111-> :=
        POST:               "chairman"
        IN_AND_OUT:         <IN_AND_OUT-9306240111-1>
                            <IN_AND_OUT-9306240111-2>
        VACANCY_REASON: OTH_UNK

<IN_AND_OUT-9306240111-1> :=
        IO_PERSON:          <PERSON-9306240111-11>
        NEW_STATUS:         OUT
        ON_THE_JOB:         NO

<IN_AND_OUT-9306240111-2> :=
        IO_PERSON:          <PERSON-9306240111-15>
        NEW_STATUS:         IN
        ON_THE_JOB:         NO

<ORGANIZATION-9306240111-8> :=
        ORG_NAME:           "Price Waterhouse"
        ORG_TYPE:           COMPANY

<PERSON-9306240111-11> :=
        PER_NAME:           "Dominic A. Tarantino"
        PER_ALIAS:          "Tarantino"
        PER_TITLE:          "Mr."

<PERSON-9306240111-15> :=
        PER_NAME:           "James L. McDonald"
        PER_ALIAS:          "McDonald"
        PER_TITLE:          "Mr."
```

Figure 6.8: A template extracted by FASTUS from which no summary could be generated.

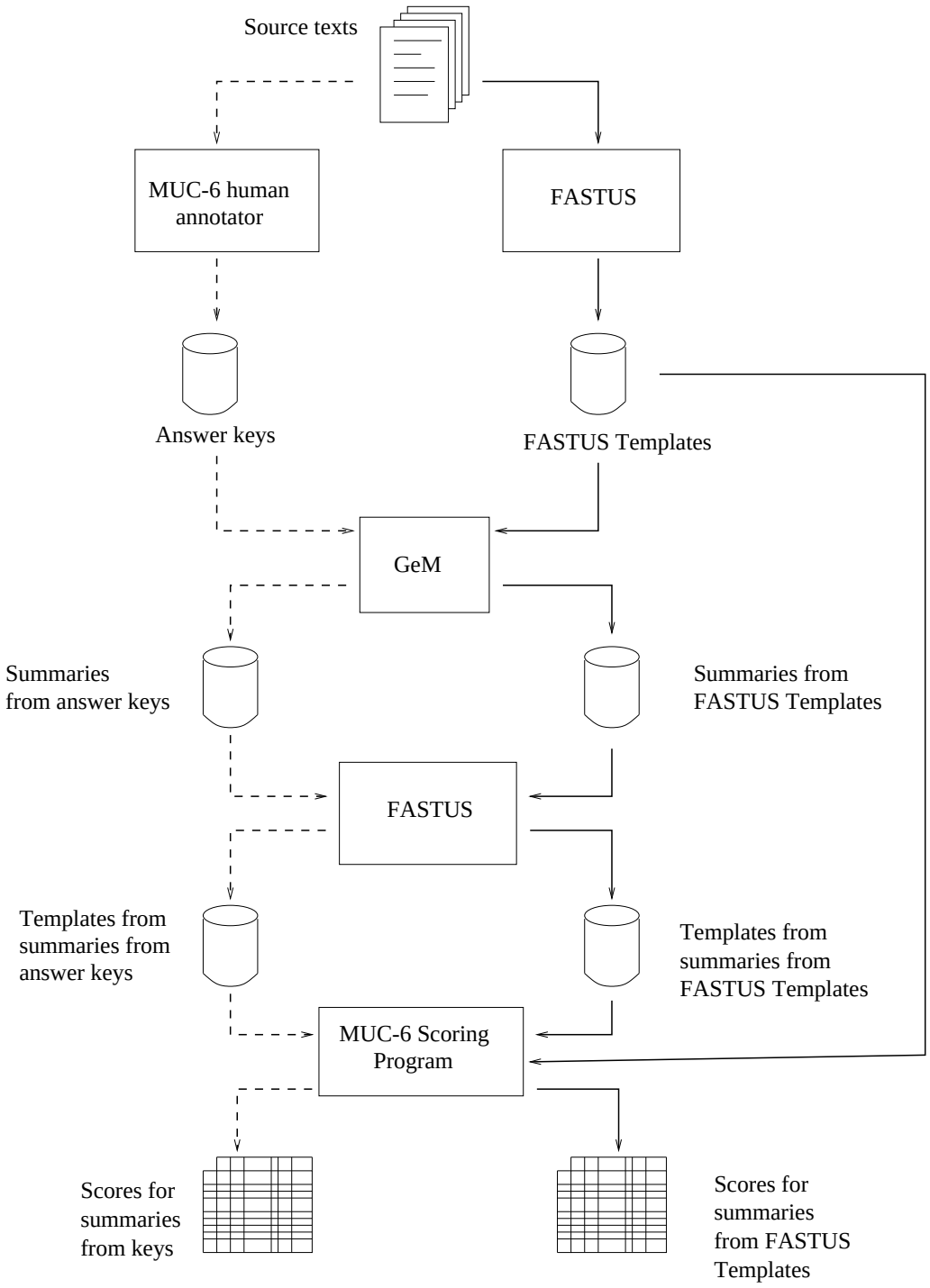


Figure 6.9: The recycling procedure for scoring summaries produced from FASTUS templates, considering FASTUS template themselves as "ground truth" instead of answer keys. Only flows drawn as solid lines, on the right side, were considered.

	POS	ACT	COR	PAR	INC	MIS	SPU	NON	REC	PRE	UND	OVG	SUB	ERR
OBJ SCORES														
organization	98	47	40	0	0	58	7	0	41	85	59	15	0	62
person	116	71	48	0	5	63	18	0	41	68	54	25	9	64
in_and_out	241	106	86	0	0	155	20	0	36	81	64	19	0	67
succession_e	199	73	62	0	0	137	11	0	31	85	69	15	0	70
template	100	100	30	0	70	0	0	0	30	30	0	0	70	70
SLOT SCORES														
organization														
name	95	47	30	0	9	56	8	0	32	64	59	17	23	71
alias	68	0	0	0	0	68	0	0	0	0	100	0	0	100
descriptor	46	0	0	0	0	46	0	0	0	0	100	0	0	100
type	98	47	40	0	0	58	7	0	41	85	59	15	0	62
locale	16	0	0	0	0	16	0	0	0	0	100	0	0	100
country	13	0	0	0	0	13	0	0	0	0	100	0	0	100
comment	0	0	0	0	0	0	0	0	0	0	0	0	0	0
person														
name	116	71	48	0	5	63	18	0	41	68	54	25	9	64
alias	73	0	0	0	0	73	0	0	0	0	100	0	0	100
title	69	0	0	0	0	69	0	0	0	0	100	0	0	100
comment	0	0	0	0	0	0	0	0	0	0	0	0	0	0
in_and_out														
io_person	241	106	65	0	21	155	20	0	27	61	64	19	24	75
new_status	241	106	75	0	11	155	20	0	31	71	64	19	13	71
on_the_job	241	106	31	0	55	155	20	0	13	29	64	19	64	88
other_org	107	49	17	0	4	86	28	0	16	35	80	57	19	87
rel_oth_org	107	49	13	0	8	86	28	0	12	27	80	57	38	90
comment	0	0	0	0	0	0	0	0	0	0	0	0	0	0
succession_e														
success_org	165	65	35	0	16	114	14	0	21	54	69	22	31	80
post	199	73	47	0	15	137	11	0	24	64	69	15	24	78
in_and_out	241	106	59	0	10	172	37	0	24	56	71	35	14	79
vac_reason	199	73	41	0	21	137	11	0	21	56	69	15	34	80
comment	0	0	0	0	0	0	0	0	0	0	0	0	0	0
template														
doc-nr	0	0	0	0	0	0	0	100	0	0	0	0	0	0
content	199	73	62	0	0	137	11	0	31	85	69	15	0	70
comment	0	0	0	0	0	0	0	0	0	0	0	0	0	0
ALL SLOTS	2534	971	563	0	175	1796	233	100	22	58	71	24	24	80
												P&R	2P&R	P&2R
F-MEASURES												32.13	43.86	25.34

Figure 6.10: Score summary table for texts generated from FASTUS output, with FASTUS output considered as the ground truth.

	POS	ACT	COR	PAR	INC	MIS	SPU	NON	REC	PRE	UND	OVG	SUB	ERR
ALL SLOTS	1806	753	456	0	137	1213	160	904	25	61	67	21	23	77
												P&R	2P&R	P&2R
F-MEASURES												35.64	47.32	28.58

Figure 6.11: Score summary table for texts generated from FASTUS output, with FASTUS output considered as the ground truth and without the influence of slots ignored by GeM.

Type	Defects	Skills
Semantic	Poor message choice. Missing message class. Redundancy due to lack of temporal recognition. Redundancy due to lack of causal knowledge.	Interesting message choice. Appropriate message combination. Appropriate detail filtering.
Discourse	Sentences too long. Missing ellipsis. Overuse of pronouns. Lack of hyperonymy. Lack of parallelism.	Appropriate use of pronouns, ellipsis and hypernyms. Sentence length variety. Consistent verb tense use.
Grammar	Repetition of locative. Repetition of temporal.	Effective clause combining. Appropriate use of conjunctions.
Syntactic	Overuse of syntactic form.	Appropriate syntactic choice.
Lexical	Repetition of terms.	Lexical variety. Metaphorical usage.

Figure 6.12: Fluency defects and skills required to avoid them grouped by type of linguistic level (modified from[Kukich, 1991]).

is expressed, so the fact that it is expressed in less space is not surprising.

6.3.2 Results of the qualitative evaluation

Numeric indices correlated with our desiderata could not always be found. Readability, for example, is highly subjective and difficult to measure: the reader can make his or her personal judgment by looking at the example summaries in Figure 5.17 and 5.18.

Despite the highly subjective nature of readability, some specific text properties can be discussed, using as a guideline the table in Figure 4.1, modified from one in [Kukich, 1991], that is repeated here in Figure 6.12.

The rest of this section discusses how effectively the architecture addresses each of these potential fluency defects, and is concluded by some notes on portability to new domains and applications.

Semantic defects

Message choice, in a sense, is not carried out within the architecture: it is the user who forces it by specifying the template composition. In its current implementation, though, the architecture suffers from missing message classes: those for expressing several of the slot fillers. This limitation is not intrinsic to the architecture.

It's difficult to give a definitive judgment on the architecture concerning possible redundancies due to lack of causal and temporal knowledge, because in the domain studied so far temporal and causal information are missing, so that almost no inference at all can be drawn that could make part of the produced summary naively obvious. It seems, however, that sentence combination would allow to reflect causal and temporal relations through an appropriate lexical choice.

Discourse defects

Discourse skills seem those that would gain most from the introduction of multi-sentence combination rules. Sentences are never too long, as length can be controlled by setting a cut-off threshold on the number of phrases, and verb tense is carefully selected. Other rhetorical devices mentioned in the table, however, as ellipsis, pronominalization, hyperonymy and parallelism, are not currently implemented. The architecture itself, though, poses no conceptual obstacle to such a development.

Grammar and syntactic defects

Sentence Combination as it is already implemented in GeM is a powerful tool for combining clauses effectively and avoiding repetitions. A primary reason for developing combination rules was indeed allowing a consistent use of conjunctions without having the number of interrogation rules grow combinatorially. The resource decomposition that allows a syntactic variant to be chosen for a sentence as an almost independent parameter greatly helps in producing a variated text. All variants recognizable by FASTUS can be easily implemented in generation as well. The choice of syntactic variants is guided by focus-tracking considerations, so that it directly contributes to the overall text cohesiveness.

Lexical defects

GeM can import the whole FASTUS lexicon, thus easily achieving a satisfactory lexical variety. New phrase structures can be added simply by copying FASTUS domain-dependent patterns. FUF randomized alternatives, moreover, allow a good variety in lexical choice, giving the designer some control on the relative frequency of synonyms.

Portability

IE systems are typically difficult to customize for the end user, and this probably limited their diffusion even more than performance issues. Most groups working on IE are studying how this obstacle can be overcome, either by providing friendly interfaces to build and maintain resources or by applying machine learning algorithms, or both things together. A feature that places FASTUS in a good position in this sense is its neat separation between domain-dependent and domain-independent linguistic sources: only a small fraction of the work must be repeated when a new domain is approached. The generation architecture proposed in this thesis shares the benefits from this philosophy by sharing FASTUS resource decomposition. Moreover, if GeM is used in conjunction with FASTUS, resource development time is further reduced because domain-dependent patterns, the top portion of the grammar, and the lexicon, can be reused.

GeM was not ported to a domain other than that of management succession event, so no conclusive word can be said on how easy it is to port it to new applications. The author's own experience, however, is that only a fraction of the overall development time had to be dedicated to resources, and that the learning curve was definitely steep: the time required for writing new interrogation rules, for instance, dropped after the first two or three were tested. Among all resources necessary to GeM, those probably more time-consuming were combination rules, most of which are domain-independent, and those for generating texts in Italian, for which no corresponding FASTUS resource was available.

6.4 Conclusions

Summarization, once more, is difficult to evaluate. Evaluation methods proposed in literature are geared towards sentence extraction, and were of little help to this thesis work. To acquire some insight on what the proposed architecture is good at, and what it could do better, a new experiment had to be imagined: summaries generated by GeM both from MUC answer keys and from actual FASTUS output were fed to FASTUS again, and resulting templates were scored using the official MUC-6 scoring software. Moreover, compression ratios were computed for both sets of summaries. Scores thus obtained were examined, and showed that, though GeM can be improved, the architecture is well conceived for the task it is designed for. Generation times were measured, and shown to be perfectly compatible with times required by a state-of-the-art Information Extraction system such as FASTUS.

For two of our posted requirements no correlated numerical indices could be found: readability and portability to new domains and applications. Portability was a central issue in all the design process, and was reflected in the care for keeping resource development cost as low as possible. Readability was assessed, on the other side, looking in the produced text for several potential fluency defects. Whenever one of them was actually found, it was shown to depend on the limited extension of resources currently developed, and not on limitations intrinsic to the architecture.

Chapter 7

Conclusions

This dissertation presented a new knowledge-based architecture for generating text from the output of an Information Extraction system, and showed the feasibility of a truly AI approach to automatic summarization within reasonable practical constraints.

The choice of a knowledge-based approach was motivated by the author's conviction that the availability of a conceptual representation of the content of the text is an invaluable advantage, both in terms of output quality and in terms of possible extensions. Summarization by Information Extraction and Generation allows coherent and cohesive summaries to be produced, two features very difficult to achieve with methods based on passage extraction. The most interesting such extension, according to the author's opinion —cross-linguistic summarization— was studied as part of this thesis project.

As a matter of fact, knowledge-based approaches tend to cost more than their statistical counterparts in terms of required resources. This problem was given a special consideration all over the project, and resulted in the decision of sharing resources developed for a different purpose, namely for SRI's FASTUS system: the linguistic and semantic knowledge required for filling a MUC template from a text is reused to operate the inverse transformation. FASTUS resources themselves are conceived so to isolate in separate modules domain-dependent resources. Moreover, an expressive specification language —FASTSPEC— allows for a very limited development time when the system must be ported to a new application. Resource reuse lets all these advantages carry over to generation as well.

A proof-of-concept prototype implementing the architecture, GeM, was also presented in detail. GeM shows how most of the processing required to produce the final text can be performed efficiently and in a uniform way. Uniformity was accomplished by adopting a powerful graph unifier, FUF, that was relied upon extensively. GeM takes less then twelve seconds, as an average, to produce a text. GeM's behavior was assessed along different dimensions by means of experimental data gathered on the 100-text corpus used for the DARPA-sponsored MUC-6 evaluation. Evaluation criteria were clearly stated, and, whenever possible, meaningful numerical indices were computed from experimental data. In cases where no such indices were available, an analytic approach was adopted to provide a thorough insight on the prototype behavior. Results were shown to be absolutely encouraging.

GeM can be developed in several ways. With relatively little effort it could be possible to improve the generation of referring expressions by means of a mechanism analogous to sentence combination, so as to introduce pronouns and definite references. The distinction between initial references to entities and subsequent references would make it possible to write interrogation and combination rules for covering slots that are currently ignored. Different control algorithm for sentence combination, as the one proposed in 5.3.6, could reduce the overall generation time.

In the medium term, more basic evolutions can be considered. The generation architecture was

conceived to share resources with an IE system that was seen as unmodifiable. The idea of resource sharing, however, could be pushed in the direction of a common formalism for describing grammars and lexicons, from which resources specialized for IE and generation could be compiled.

Beside cross-linguistic summarization, there are other extensions for which the architecture presented in this dissertation is well suited. The first concerns generating a single summary from more than one text. Some very preliminary experiments were already carried out, as GeM was designed from the beginning to accept a set of templates as input. Multiple-text input, however, must be dealt with carefully, because incorrect or misleading text is more likely to be produced than in the single-text input case. In the management succession domain, for example, if no information were available about the chronological order of the events in the source texts, it would be perfectly possible to report on a person vacating a position and then acquiring it, or to produce a text from which it could be inferred that the same position was held by several persons at the same time.

Another extension, already studied by other authors [Robin, 1994, Radev, 1997] consists in enabling the generator to access other resources in order to enrich the summary with background information. This could be done seamlessly by representing background information within the same knowledge base used for template interrogation. Both these extensions are indeed made possible by the adopted knowledge-based approach.

Bibliography

[Aberdeen *et al.*, 1995] John Aberdeen, John Burger, David Day, Lynette Hirschman, Patricia Robinson, and Marc Vilain. MITRE: Description of the ALEMBIC system used for MUC-6. In *Proceedings of the Sixth Message Understanding Conference (MUC-6)*. DARPA, Morgan Kaufmann, November 1995.

[Abney, 1996] Steven Abney. Partial parsing via finite-states cascades. In *Proceedings of the ESSLLI '96 Workshop on Robust Parsing*, 1996.

[Aone *et al.*, 1997] Chinatsu Aone, Mary Ellen Okurowski, James Gorlinsky, and Bjornar Larsen. A scalable summarization system using robust nlp. In *Proceedings of the 1997 ACL/EACL Workshop on Intelligent Scalable Text Summarization*, Madrid, Spain, July 1997.

[Appelt *et al.*, 1995] Douglas E. Appelt, Jerry R. Hobbs, John Bear, David Israel, Megumi Kameyama, Andy Kehler, David Martin, Karen Myers, and Mabry Tyson. SRI international FASTUS system MUC-6 test results and analysis. In *Proceedings of the Sixth Message Understanding Conference (MUC-6)*, pages 237–248. Morgan Kaufmann Publishers, 1995.

[Barzilay and Elhadad, 1997] Regina Barzilay and Michael Elhadad. Using lexical chains for text summarization. In *Proceedings of the 1997 ACL/EACL Workshop on Intelligent Scalable Text Summarization*, Madrid, Spain, July 1997.

[Bear *et al.*, 1997] John Bear, David Israel, Jeff Petit, and David Martin. Using information extraction to improve information retrieval. In *Proceedings of the sixth Text Retrieval Conference*, Garthiersburg, MD, November 1997.

[Boguraev and Kennedy, 1997] Branimir Boguraev and Christofer Kennedy. Salience-based content chaacterisation of text documents. In *Proceedings of the 1997 ACL/EACL Workshop on Intelligent Scalable Text Summarization*, Madrid, Spain, July 1997.

[Bourbeau *et al.*, 1990] L. Bourbeau, D. Carcagno, E. Goldberg, R. Kittredge, and A. Polguere. Bilingual generation of weather forecasts in an operations environment. In *Proceedings of the 13th International Conference on Computational Linguistics (COLING)*, Helsinki, Finland, 1990.

[Brill, 1993] Eric Brill. *A Corpus-Based Approach to Language Learning*. PhD thesis, University of Pennsilvanya, December 1993. IRCS Report 93-44.

[Buchanan *et al.*, 1995] B. Buchanan, J. Moore, D. Forsythe, G. Carenini, G. Banks, and S. Ohlsson. An intelligent interactive system for delivering individualized information to patients. *Artificial Intelligence in Medicine*, (7):117–154, 1995.

[Cancedda *et al.*, 1997] Nicola Cancedda, Gjertrud Kamstrup, Emanuele Pianta, and Ettore Pietrosanti. SAX: Generating hypertexts from SADT models. In *Proceedings of the International Workshop on Natural Language and Databases (NLDB'97)*, Vancouver, Canada, 1997.

[Carpenter, 1992] Robert Carpenter. *The Logic of Typed Feature Structures.* Number 32 in Cambridge Tracts in Theoretical Computer Science. Cambridge University Press, 1992.

[Cavazza and Zweigenbaum, 1994] Marc Cavazza and Pierre Zweigenbaum. Semantic analysis and in-depth understanding of technical texts. *Applied Artificial Intelligence*, 8(3):425–453, 1994.

[Cawsey *et al.*, 1995] A. Cawsey, K. Binsted, and R. Jones. Personalised explanations for patient education. In *Proceedings of the Fifth European Workshop on Natural Language Generation*, pages 59–74, 1995.

[Childs *et al.*, 1995] Lois Childs, Deb Brady, Louise Guthrie, Jose Franco, Dan Valdes-Dapena, Bill Reid, John Kielty, Glenn Dierkes, and Ira Sider. Lockheed martin: LOUELLA PARSING, an NLToolset for MUC-6. In *Proceedings of the Sixth Message Understanding Conference (MUC-6).* DARPA, Morgan Kaufmann, November 1995.

[Church, 1988] Kenneth Ward Church. A stochastic parts program and noun phrase parser for unrestricted text. In *Proceedings of the 2nd Conference on Applied Natural Language Processing*, Austin, Texas, 1988.

[Ciravegna, 1995] Fabio Ciravegna. Understanding messages in a diagnostic domain. *Information processing and managment.*, 31(5), 1995. (special issue on summarizing).

[Corston-Oliver, 1998] Simon Corston-Oliver. Beyond string matching and cue phrases: Improving efficiency and coverage in discourse analysis. In *Working Notes of the AAAI 1998 Spring Symposium on Intelligent Text Summarization*, Stanford University, California, March 1998.

[DARPA, 1995] DARPA, editor. *Proceedings of the Sixth Message Understanding Conference (MUC-6).* Morgan Kaufmann, November 1995.

[Ejerhed, 1988] Eva I. Ejerhed. Finding clauses in unrestricted text by finitary and stochastic methods. In *Proceedings of the 2nd Conference on Applied Natural Language Processing*, Austin, Texas, 1988.

[Elhadad and Robin,] Michael Elhadad and Jacques Robin. SURGE: a comprehesive plug-in realization component for text generation. Submitted. Available via anonymous ftp as ftp://ftp.cs.bgu.ac.il/pub/people/elhadad/surge2.ps.gz.

[Elhadad, 1993] Michael Elhadad. *FUF: the Universal Unifier – User Manual Version 5.2.* Department of Computer Science, Ben Gurion University of the Negev., 1993.

[Endres-Niggemeyer, 1998a] Birgitte Endres-Niggemeyer. *Summarizing Information.* Springer, Heidelberg, 1998.

[Endres-Niggemeyer, 1998b] Brigitte Endres-Niggemeyer. A grounded theory approach to expert summarization. In *Working notes of the AAAI 1998 Spring Symposium on Intelligent Text Summarization*, Stanford University, California, March 1998.

[Fisher *et al.*, 1995] David Fisher, Stephen Soderland, Joseph McCarthy, Fangfang Feng, and Wendy Lehnert. Description of the UMass system as used for the MUC-6. In *Proceedings of the Sixth Message Understanding Conference (MUC-6).* DARPA, Morgan Kaufmann, November 1995.

[Friedman, 1986] Carol Friedman. Automatic structuring of sublanguage information. In *Analyzing Language in Restricted Domains: Sublanguage Description and Processing*, pages 85–102. LEA, Hillsdale, New Jersey, 1986.

[Gabriel, 1991] Richard P. Gabriel. Deliberate writing. In David D. McDonald and Leonard Bolc, editors, *Natural Language Generation Systems*. Springer Verlag, 1991.

[Gaizauskas *et al.*,] Rob Gaizauskas, Pete Rodgers, Hamish Cunningham, Kevin Humphreys, and Sandy Robertson. *GATE User Guide*. Department of Computer Science and Institule for Language, Speech and Hearing (ILASH). University of Sheffield, UK.

[Gaizauskas *et al.*, 1995] R. Gaizauskas, T. Wakao, K. Humphreys, H. Cunningham, and Y. Wilks. University of sheffield: Description of the LaSIE system as used for MUC-6. In *Proceedings of the Sixth Message Understanding Conference (MUC-6)*. DARPA, Morgan Kaufmann, November 1995.

[Garey and Johnson, 1979] Michael R. Garey and David S. Johnson. *Computers and Intractability: a Guide to the Theory of NP-Completeness*. W.H.Freeman and Company, New York, NY, 1979.

[Geldof and Van de Velde, 1997] S. Geldof and W. Van de Velde. An architecture for template-based (hyper)text generation. In *Proceedings of the Sixth European Workshop on Natural Language Generation*, pages 28–37, 1997.

[Gilardoni *et al.*, 1994] Luca Gilardoni, Paola Prunotto, and Gianluigi Rocca. Hierarchical pattern matching for knowledge based news categorization. In *Proceedings of the RIAO '94*, New York, New York, October 1994.

[Goldberg *et al.*, 1994] E. Goldberg, N. Driedgar, and R. Kittredge. Using natural-language processing to produce weather forecasts. *IEEE Expert*, (9):45–53, 1994.

[Grefenstette, 1998] Gregory Grefenstette. Producing intelligent telegraphic text reduction to provide an audio scanning service for the blind. In *Working notes of the AAAI 1998 Spring Symposium on Intelligent Text Summarization*, Stanford University, California, March 1998.

[Grishman, 1995] Ralph Grishman. The NYU system for MUC-6 or where's the syntax. In *Proceedings of the Sixth Message Understanding Conference (MUC-6)*. DARPA, Morgan Kaufmann, November 1995.

[Grosz and Sidner, 1986] Barbara J. Grosz and Candace L. Sidner. Attention, intentions, and the structure of discourse. *Computational Linguistics*, 12(6), July-September 1986.

[Halliday and Hasan, 1976] Michael Halliday and Ruqaiya Hasan. *Cohesion in English*. Longman, London, 1976.

[Hand, 1997] Thérèse Firmin Hand. A proposal for task-based evaluation of text summarization systems. In *Proceedings of the 1997 ACL/EACL Workshop on Intelligent Scalable Text Summarization*, Madrid, Spain, July 1997.

[Hayes *et al.*, 1988] Philip J. Hayes, Laura E. Knecht, and Monica J. Cellio. A news story categorization system. In *Proceedings of the 2nd Conference on Applied Natural Language Processing*, Austin, Texas, 1988.

[Hobbs *et al.*, 1997] Jerry Hobbs, Douglas Appelt, John Bear, David Israel, Megumi Kameyama, Mark Stickel, and Mabry Tyson. FASTUS: A cascaded finite-state transducer for extracting information from natural-language text. In Emmanuel Roche and Yves Schabes, editors, *Finite-State Language Processing*, chapter 13, pages 383–406. The MIT Press, Cambridge, Massachussets, 1997.

[Hobbs, 1985] Jerry R. Hobbs. On the coherence and structure of discourse. Technical Report CSLI-85-37, CSLI, Stanford, CA, 1985.

[Hobbs, 1993] Jerry R. Hobbs. Summaries from structure. In *Working Notes of the Dagsthul Seminar on Summarizing Text for Intelligent Communication*, 1993.

[Hovy and Meier, 1994] Eduard H. Hovy and Elisabeth Meier. Parsimonious or profligate: how many and which discourse structure relations? *Discourse Processes*, 1994.

[Hovy, 1993] Eduard Hovy. Automated discourse generation using discourse structure relations. *Artificial Intelligence*, 63(1-2):341–386, October 1993.

[Iordanskaja *et al.*, 1992] L. Iordanskaja, M. Kim, R. Kittredge, B. Lavoie, and A. Polguère. Generation of extended bilingual statistical reports. In *Proceedings of the Fifth International Conference on Computational Linguistics (COLING-92)*, volume 3, pages 1019–1023, 1992.

[Jacobs, 1991] Paul S. Jacobs. PHRED: A generator for natural language interfaces. In David D. McDonald and Leonard Bolc, editors, *Natural Language Generation Systems*, pages 312–351. Springer Verlag, 1991.

[Jacobs, 1992] Paul S. Jacobs. Joining statistics with nlp for text categorization. In *Proceedings of the third Conference on Applied Natural Language Processing*, Trento, Italy, 1992.

[Jacobs, 1993] Paul S. Jacobs. Using statistical methods to improve knowledge-based news categorization. *IEEE Expert*, pages 13–23, April 1993.

[Jing *et al.*, 1998] Hongyan Jing, Kathleen McKeown, Regina Barzilay, and Michael Elhadad. Summarizing evaluation methods: Experiments and analysis. In *Working notes of the AAAI 1998 Spring Symposium on Intelligent Text Summarization*, Stanford University, California, March 1998.

[Johnson *et al.*, 1993] F. C. Johnson, C. D. Paice, W. J. Black, and A. P. Neal. The application of linguistic processing to automatic abstract generation. *Journal of Document and Text Management*, 1(3):215–241, 1993.

[Kameyama, 1997] Megumi Kameyama. Information extraction across linguistic barriers. In AAAI, editor, *Electronic Working Notes: AAAI Spring Symposium on Cross-Language Text and Speech Retrieval*, AAAI Spring Symposium Series, Menlo Park, CA, March 1997.

[Kaplan and Bresnan, 1982] Ronald M. Kaplan and Joan Bresnan. Lexical-functional grammar: A formal system for grammatical representation. In Joan Bresnan, editor, *The mental representation of grammatical relation*, chapter 4. The MIT Press., 1982.

[Kasper, 1989] R. Kasper. A flexible interface for linking applications to penman's sentence generator. In *Proceedings of the 1989 DARPA Speech and Natural Language Workshop*, pages 153–158, 1989.

[Kay, 1979] Martin Kay. Functional unification grammar. In *Proceedings of the 5th Annual Meeting of the Berkeley Linguistic Society*, 1979.

[Krupka, 1995] George R. Krupka. SRA: Description of the SRA system used for MUC-6. In *Proceedings of the Sixth Message Understanding Conference (MUC-6)*. DARPA, Morgan Kaufmann, November 1995.

[Kukich, 1991] Karen Kukich. Fluency in natural language reports. In David McDonald and Leonard Bolc, editors, *Natural Language Generation Systems*. Springer Verlag, 1991.

[Lee, 1995] Richard Lee. Sterling software: an NLToolset-based system for MUC-6. In *Proceedings of the Sixth Message Understanding Conference (MUC-6)*. DARPA, Morgan Kaufmann, November 1995.

[Luhn, 1968] H.P. Luhn. The automatic creation of literature abstracts. In Schultz, editor, *H.P.Luhn: Pioneer of Information Science*. Spartan, 1968.

[MacGregor and Bates, 1987] R. M. MacGregor and R. Bates. The loom knowledge representation language. Technical Report ISI/RS-87-188, USC/ISI, 1987.

[Mani *et al.*, 1998] Inderjeet Mani, Eric Bloedorn, and Barbara Gates. Using cohesion and coherence models for text summarization. In *Working notes of the AAAI 1998 Spring Symposium on Intelligent Text Summarization*, Stanford University, California, March 1998.

[Mann and Thompson, 1988] William C. Mann and Sandra A. Thompson. Rhetorical structure theory: Towards a functional theory of text organization. *Text*, 8(3):243–281, 1988.

[Marcu, 1997a] Daniel Marcu. From discourse structures to text summaries. In *Proceedings of the 1997 ACL/EACL Workshop on Intelligent Scalable Text Summarization*, Madrid, Spain, July 1997.

[Marcu, 1997b] Daniel Marcu. From local to global coherence, a bottom-up approach to text planning. In *Proceedings of the Fourteenth National Conference on Artificiali Intelligence*, pages 629–635, Providence, Rhode Island, 1997.

[Marcu, 1997c] Daniel Marcu. The rethorical parsing of natural language texts. In *Proceedings of the 35th Annual Meeting of the Association for Computational Linguistics*, pages 97–103, Santa Cruz, California, 1997.

[Marcu, 1998] Daniel Marcu. To build text summaries of high quality, nuclearity is not sufficient. In *Working Notes of the AAAI 1998 Spring Symposium on Intelligent Text Summarization*, Stanford University, California, March 1998.

[Marsh, 1986] Elaine Marsh. General semantic patterns in different sublanguages. In Ralph Grishman and Richard Kittredge, editors, *Analyzing Language in Restricted Domains: Sublanguage Description and Processing*, pages 103–127. Lawrence Erlbaum Associates, Hillsdale, New Jersey, 1986.

[McDonald, 1992] David. D. McDonald. An efficient chart-based algorithm for partial-parsing of unrestricted texts. In *proceedings of the 3rd Conference on Applied Natural Language Processing*, Trento, Italy, 1992.

[McKeown and Radev, 1995] Kathleen McKeown and Dragomir Radev. Generating summaries of multiple news articles. In *Proceedings of the 18[18] Annual International ACM SIGIR Conference on Research and Develpment in Information Retrieval*, pages 74–82, Seattle, Washington, 1995.

[McKeown *et al.*, 1994] Kathleen McKeown, Karen Kukich, and James Shaw. Practical issues in automatic document generation. In *Proceedings of the Fourth Conference on Applied Natural-Language Processing*, pages 7–14, 1994.

[McKeown, 1985] Kathleen McKeown. *Text Generation*. Cambridge University Press, 1985.

[Mellish *et al.*, 1992] Chris Mellish, David Allport, Anthony F. Hartley, Roger Evans, Lynne J. Cahill, Robert Gaizauskas, and John Walker. The TIC message analyser. Draft, April 1992.

[Miller *et al.*, 1990] George A. Miller, Richard Beckwith, Christiane Fellbaum, Derek Gross, and Katherine J. Miller. Introduction to wordnet: An on-line lexical database. *International Journal of Lexicography*, 3(4):235–312, 1990.

[Minel *et al.*, 1997] Jean-Luc Minel, Sylvaine Nugier, and Gérald Piat. How to appreciate the quality of automatic text summarization? example of FAN and MLUCE protocols and their results on SERAPHINE. In *Proceedings of the 1997 ACL/EACL Workshop on Intelligent Scalable Text Summarization*, Madrid, Spain, July 1997.

[Moore and Paris, 1993] Johanna Moore and Cécile Paris. Planning text for advisory dialogues: Capturing intentional and rhetorical information. *Computational Linguistics*, (19):651–694, 1993.

[muc7proc, 1998] Working notes of the seventh message understanding conference (MUC-7)., April 1998.

[Not and Pianta, 1995] Elena Not and Emanuele Pianta. Issues of multilinguality in the automatic generation of administrative instructional texts. In Marco Gori and Giovanni Soda, editors, *Topics in Artificial Intelligence, Proceedings of the Fourth Conference of the Italian Association for Artificial Intelligence (AI*IA '97)*, Firenze, 1995. Springer. LNAI 992.

[Ono *et al.*, 1994] K. Ono, K. Sumita, and S. Miike. Abstract generation based on rhetorical structure. In *Proceedings of the International Conference on Computational Linguistics (COLING-94)*, pages 344–348, 1994.

[Paice, 1990] Chris D. Paice. Constructing literary abstract by computer: Techniques and prospects. *Information Processing & Management*, 26(1):171–186, 1990.

[Radev, 1997] Dragomir Radev. Generating natural language summaries from multiple on-line sources. PhD Thesis proposal, Columbia University, 1997.

[Reiter and Dale, 1997] Ehud Reiter and Robert Dale. Building applied natural language generation systems. *Natural Language Engineering*, 3:57–87, 1997.

[Reiter *et al.*, 1995] E. Reiter, C. Mellish, and J. Levine. Automatic generation of technical documentation. *Applied Artificial Intelligence*, (9):259–287, 1995.

[Resnick *et al.*, 1993] Lori Alperin Resnick, Alex Borgida, Ronald J. Brachman, Deborah L. McGuinness, Peter F. Patel-Schneider, and Kevin C. Zalondel. *CLASSIC Description and Reference Manual*, 1993.

[Robin, 1994] Jacques Robin. Revision-based generation of natural language summaries providing historical background. Technical Report CUCS-034-94, Columbia University, New York, December 1994.

[Roche and Schabes, 1997] Emmanuel Roche and Yves Schabes. *Finite-State Language Processing*, chapter Introduction. The MIT Press, 1997.

[Salton *et al.*, 1997] G. Salton, A. Singhal, M. Mitra, and C. Buckley. Automatic text structuring and summarization. *Information Processing & Management*, 33(2):193–208, 1997.

[Shieber *et al.*, 1990] Stuart M. Shieber, Gertjan van Noord, Fernando C. N. Pereira, and Robert C. Moore. Semantic-head-driven generation. *Computational Linguistics*, 16(1), 1990.

[Strzalkowski *et al.*, 1998] Tomek Strzalkowski, Jin Wang, and Bowden Wise. A robust practical text summarization. In *Working notes of the AAAI 1998 Spring Symposium on Intelligent Text Summarization*, Stanford University, California, March 1998.

[Sundheim, 1991] Beth Sundheim, editor. *Proceedings of the Third Message Understanding Conference (MUC-3)*, San Diego, California, May 1991. Morgan Kaufmann Publishers.

[Sundheim, 1992] Beth Sundheim, editor. *Proceedings of the Fourth Message Understanding Conference (MUC-4)*, McLean, Virginia, June 1992. Morgan Kaufmann Publishers.

[Swartout, 1983] William Swartout. XPLAIN: a system for creating and explaining expert consulting systems. *Artificial Intelligence*, (21):285–325, 1983.

[Weischedel, 1995] Ralph Weischedel. BBN: Description of the PLUM system as used for MUC-6. In *Proceedings of the Sixth Message Understanding Conference (MUC-6)*. DARPA, Morgan Kaufmann, November 1995.

[Woods, 1970] William A. Woods. Transition networks grammars for natural language analisys. *Communications of the Association for Computing Machinery*, 13(10):591–606, 1970.

Università *La Sapienza*
Dottorato di Ricerca in Ingegneria Informatica
Collana delle tesi
Collection of Theses

V-93-1 Marco Cadoli. *Two Methods for Tractable Reasoning in Artificial Intelligence: Language Restriction and Theory Approximation.* June 1993.

V-93-2 Fabrizio d'Amore. *Algorithms and Data Structures for Partitioning and Management of Sets of Hyperrectangles.* June 1993.

V-93-3 Miriam Di Ianni. *On the complexity of flow control problems in Store-and-Forward networks.* June 1993.

V-93-4 Carla Limongelli. *The Integration of Symbolic and Numeric Computation by p-adic Construction Methods.* June 1993.

V-93-5 Annalisa Massini. *High efficiency self-routing interconnection networks.* June 1993.

V-93-6 Paola Vocca. *Space-time trade-offs in directed graphs reachability problem.* June 1993.

VI-94-1 Roberto Baldoni. *Mutual Exclusion in Distributed Systems.* June 1994.

VI-94-2 Andrea Clementi. *On the Complexity of Cellular Automata.* June 1994.

VI-94-3 Paolo Giulio Franciosa. *Adaptive Spatial Data Handling.* June 1994.

VI-94-4 Andrea Schaerf. *Query Answering in Concept-Based Knowledge Representation Systems: Algorithms, Complexity, and Semantic Issues.* June 1994.

VI-94-5 Andrea Sterbini. *2-Thresholdness and its Implications: from the Synchronization with PVchunk to the Ibaraki-Peled Conjecture.* June 1994.

VII-95-1 Piera Barcaccia. *On the Complexity of Some Time Slot Assignment Problems in Switching Systems.* June 1995.

VII-95-2 Michele Boreale. *Process Algebraic Theories for Mobile Systems.* June 1995.

VII-95-3 Antonella Cresti. *Unconditionally Secure Key Distribution Protocols.* June 1995.

VII-95-4 Vincenzo Ferrucci. *Dimension-Independent Solid Modeling.* June 1995.

VII-95-5 Esteban Feuerstein. *On-line Paging of Structured Data and Multi-threaded Paging.* June 1995.

VII-95-6 Michele Flammini. *Compact Routing Models: Some Complexity Results and Extensions.* June 1995.

VII-95-7 Giuseppe Liotta. *Computing Proximity Drawings of Graphs.* June 1995.

VIII-96-1 Luca Cabibbo. *Querying and Updating Complex-Object Databases.* May 1996.

VIII-96-2 Diego Calvanese. *Unrestricted and Finite Model Reasoning in Class-Based Representation Formalisms.* May 1996.

VIII-96-3 Marco Cesati. *Structural Aspects of Parameterized Complexity.* May 1996.

VIII-96-4 Flavio Corradini. *Space, Time and Nondeterminism in Process Algebras.* May 1996.

VIII-96-5 Stefano Leonardi. *On-line Resource Management with Application to Routing and Scheduling.* May 1996.

VIII-96-6 Rosario Pugliese. *Semantic Theories for Asynchronous Languages.* May 1996.

IX-97-1 Paola Alimonti. *Local search and approximability of MAX SNP problems.* May 1997.

IX-97-2 Tiziana Calamoneri. *Does Cubicity Help to Solve Problems?.* May 1997.

IX-97-3 Paolo Di Blasio. *A Calculus for Concurrent Objects: Design and Control Flow Analysis.* May 1997.

IX-97-4 Bruno Errico. *Intelligent Agents and User Modelling.* May 1997.

IX-97-5 Roberta Mancini. *Modelling Interactive Computing by exploiting the Undo.* May 1997.

IX-97-6 Riccardo Rosati. *Autoepistemic Description Logics.* May 1997.

IX-97-7 Luca Trevisan. *Reductions and (Non-)Approximability.* May 1997.

X-98-1 Gianluca Battaglini. *Analysis of Manufacturing Yield Evaluation of VLSI/WSI Systems: Methods and Methodologies.* April 1998.

X-98-2 Piergiorgio Bertoli. *Using OMRS in practice: a case study with Acl-2.* April 1998.

X-98-3 Chiara Ghidini. *A semantics for contextual reasoning: theory and two relevant applications.* April 1998.

X-98-4 Roberto Giaccio. *Visiting complex structures.* April 1998.

X-98-5 Giampaolo Greco. *Dimension and structure in Combinatorics.* April 1998.

X-98-6 Paolo Liberatore. *Compilation of intractable problems and its application to artificial intelligence.* April 1998.

X-98-7 Fabio Massacci. *Efficient approximate tableaux and an application to computer security.* April 1998.

X-98-8 Chiara Petrioli. *Energy-Conserving Protocols for Wireless Communications.* April 1998.

X-98-9 Giulio Balestreri. *An Algebraic Semantics for the Shared Spaces Coordination Languages.* April 1999.

XI-99-1 Luca Becchetti. *Efficient Resource Management in High Bandwidth Networks.* April 1999.

XI-99-2 Nicola Cancedda. *Text Generation from Message Understanding Conference Templates.* April 1999.

XI-99-3 Luca Iocchi. *Design and Development of Cognitive Robots.* April 1999.

XI-99-4 Francesco Quaglia. *Consistent checkpointing in distributed computations: theoretical results and protocols.* April 1999.

XI-99-5 Milton Romero. *Disparity/Motion Estimation For Stereoscopic Video Processing.* April 1999.