Spanish generation in the NLP system 'LOLITA'

Miquel A. Fernández

University of Durham

Spanish Generation in the NLP System 'LOLITA'

Miguel A. Fernández

Laboratory for Natural Language Engineering,
Department of Computer Science,
University of Durham, U.K.

Submitted for the degree of Master of Science,
October 1995

Miguel Angel Fernández

Spanish Generation in the NLP System 'LOLITA'

Master of Science, 1995

Abstract

The aim of this research has been to modify the NLG module in the NLP system LOLITA to enable it to produce Spanish utterances. Natural Language Generation (NLG) is the production of text in a surface language by the computer in order to meet communicative goals. The NLG module of LOLITA is currently able to generate English utterances. It provides the generation capabilities required for the prototype applications built onto LOLITA. The module also aids in the development and debugging of the system as NL utterances are easier to understand than the semantic network representation. The LOLITA generator receives as input the whole LOLITA semantic network,'SemNet',(the system knowledge base) and adopts the traditional two components architecture. However, the distribution of task between the planner and plan-realiser (planner and realiser in other systems) differs from that in traditional systems as the plan-realiser can perform tasks such as the selection of content traditionally performed by a planner. The Spanish generator is based upon the same theoretical principles as the current English generator. SemNet forms the input of the generator and has been expanded for this purpose by the addition of Spanish lexical entries and information associated with them. The existing planner module has been used while the plan-realiser has been modified by developing new solutions where the existing ones were not adequate for producing correct Spanish utterances. The generator has been implemented in the pure functional language Haskell, taking advantage of several features of this language and, like LOLITA, it has been built following Natural Language Engineering principles. These two aspects influencing the research are also described.

Declaration

The material contained within this thesis has not previously been submitted for a degree at the University of Durham or any other university. The research reported within this thesis has been conducted by the author unless indicated otherwise.

The copyright of this thesis rests with the author. No quotation from it should be published without his prior written consent and information derived from it should be acknowledged.

Contents

List of Figures

Chapter 1

Introduction

1.1 Problem Outline

The aim of Natural Language Processing (NLP) is to develop a computer system capable of understanding human language. Therefore, the system must be capable of analysing, understanding and communicating in natural language.The last task means that a NLP system should be able to produce NL text and this is the aim of Natural Language Generation (NLG). NLG is the automatic generation of Natural Language by computer in order to meet communicative goals. However, there is not an unique natural language and a system, as well as humans, should be able to cope with more than one natural language. The differences and similarities of two natural languages affect the way a generator, capable of producing both of them, is built: a single process can be used where the languages coincide while language-dependent processes are necessary where they diverge. The aim of this project was to allow the generation module in the LOLITA system, which is currently able to produce English utterances, to generate Spanish utterances.

1.2 Generation Process

As stated above, the aim of NLG is to produce an utterance in a particular natural language. The definition of the task carried out by a generator differs amongst researchers depending on the input the generator receives and the activities it has to perform. The most common activities that a generation process carries out [Meteer, 1993] are:

- **Content Delimitation**: Choosing what information the utterance should express.
- **Unit Organisation**: Determining a coherent organisation of the units delimited by the chosen content by the above activity.
- **Lexical Selection**: Choosing the content words.
- **Syntactic Structures Selection**: Choosing the syntactic constituents of the text.
- **Morphology and Read Out**: Producing the actual text from a syntactic representation with lexical items inserted. The main function of this process is to produce the correct morphological forms of the words.
- **Focus**: Determining which entities are most relevant at a given point and how they affect the choices made in the generator.

Many researchers group these activities by adopting a distinction between two stages of natural language generation: deciding 'what to say' and deciding 'how to say it'. While there is general agreement on the activities and components described, there is less agreement on which activities go into which components, differing from system to system.

1.3 Thesis Structure

The thesis is composed of the following chapters:

Chapter 2: Context of the work, provides details of the LOLITA system and how it uses the generator module, together with an overview of the NLE principles and the main features of the functional programming language Haskell.

Chapter 3: Related Work, includes an overview of the different problems and approaches in the area of NLG. Systems receiving a similar input to the LOLITA generator are presented. Finally, the current LOLITA generation module is described.

Chapter 4: Generating Spanish with LOLITA, provides details of how the Spanish generator has been merged with the current English generator in the LOLITA system. Details are provided of the solutions adopted in the plan-realiser module for Spanish generation and heuristics and examples are provided. Finally, some implementation details are provided together with examples of how the features of the programming language Haskell have influenced the solution.

Chapter 5: Evaluation and Results, presents a simple experiment which evaluates the generator by comparing human generated descriptions with those produced by LOLITA.

Chapter 6: Conclusions, summarises the project's practical success and it also suggests possibilities for further work.

Chapter 2

Context of the work

This chapter will introduce some aspects which have influenced the development of the work presented in this paper. Firstly the Natural Language Processing system LOLITA, of which the generator is a component, will be described. LOLITA has been developed following the principles of Natural Language Engineering (NLE) [Garigliano and Tate, 1995]. This methodology will be introduced in the next section. The functional programming language Haskell [Hudak *et al.*, 1994] has been used in the implementation of the solution. The properties and features of Functional Programming, and more specifically Haskell, which are not found in other programming languages will be presented.

2.1 The LOLITA system

The LOLITA (Large-scale, Object-based, Linguistic Interactor, Translator and Analyser) system [Garigliano *et al.*, 1992][Long and Garigliano, 1994][Hazan et al, 1993] has been under development at the University of Durham since 1986. LOLITA is a NLP system based on NLE principles (see Section 2.2) able to parse complex texts, semantically and pragmatically analyse the resulting parse trees and add the information to the semantic network (SemNet). The system can also reason about and answer natural language queries about the knowledge stored by generating natural language from the network. The overall structure of the LOLITA system is showed in Figure 2.1.

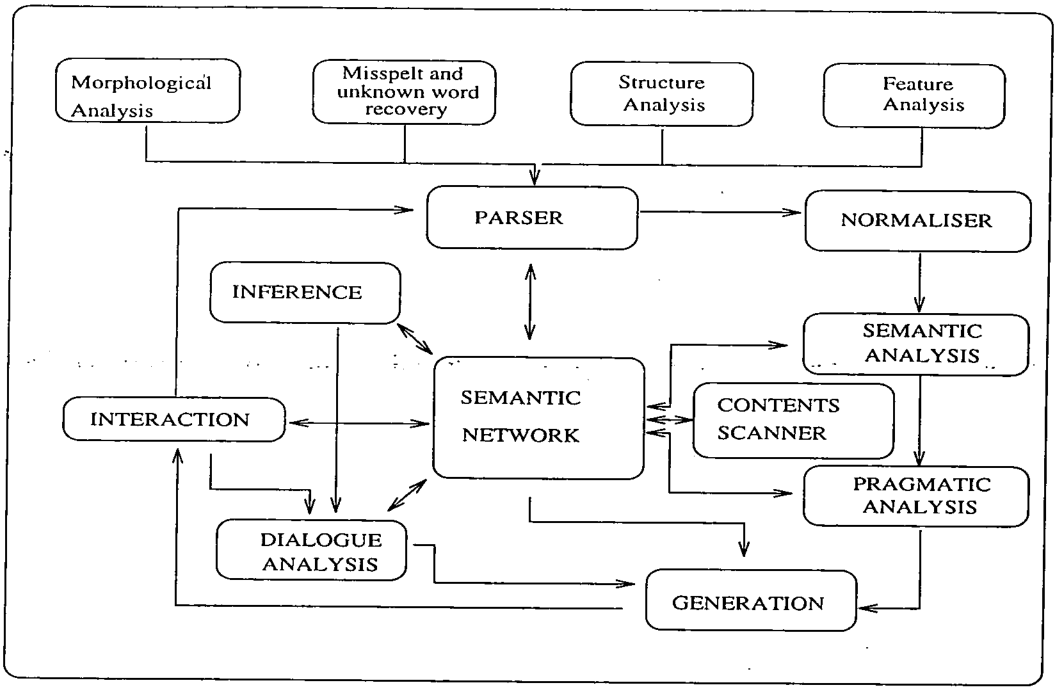


Figure 2.1: Structure of the LOLITA system.

LOLITA is defined as a *general purpose base system.* This definition is an extension of the terminology presented in [Galliers and Sparck Jones, 1993]. A *generic system* is defined as a system designed to perform a task in different domains. A *general purpose system* is categorised as one that can be used for any task in any domain without further modifications. It is at the intersection of these two types of systems that *general purpose base systems*, including LOLITA, belong.

2.1.1 LOLITA semantic network

The knowledge representation used in LOLITA is important as the whole system is built around it. It is particularly important for the generator, as the semantic network (SemNet) provides its input.

This representation forms a semantic network [Long and Garigliano, 1994] (similar to Conceptual Graph Theory [Sowa, 1984]) in which concepts are represented by nodes and relationships (links between concepts) by arcs. This structure holds world information and data, as well as some linguistic information. The concepts , for example entities or events, are arranged in hierarchies with entities and events lower in the hierarchy inheriting properties from those higher up. Figure 2.2 shows a simplified portion of SemNet representing the event 'the taxi burned fiercely'.

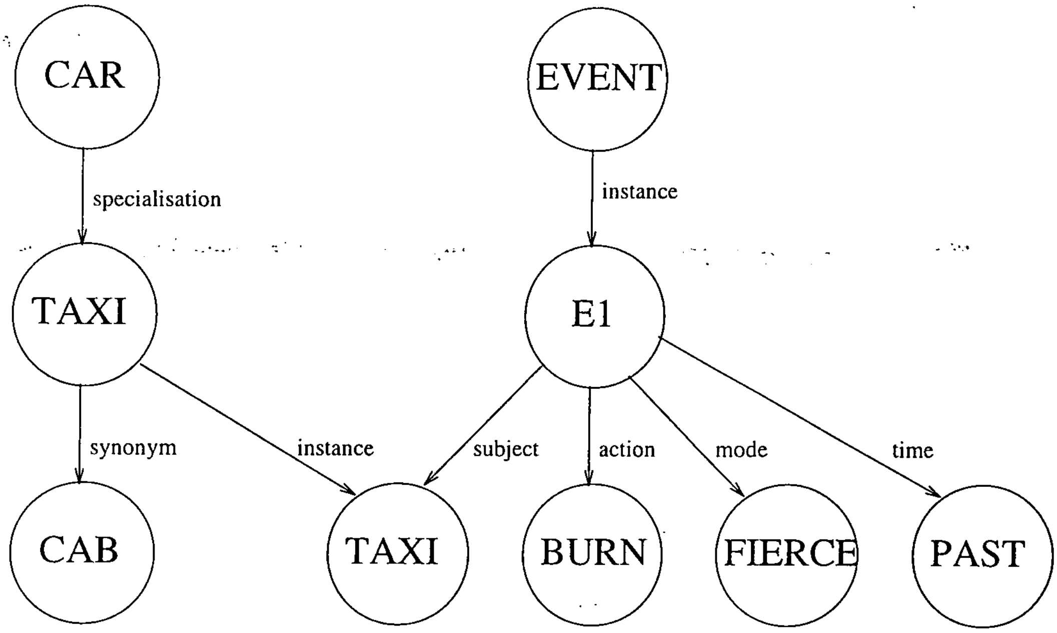


Figure 2.2: A portion of SemNet

Attached to each node is a set of control variables which contain basic information about the node. This information is essential for many components of the system (including generation) and it needs to be accessed often and quickly. Defining the *meaning* of a concept requires reference not only to the relevant node but also to the whole semantic network.

In this representation the concepts have a smaller 'grain size' than words (where words are any lexical entry in a surface language). For every word there is at least one different concept, more than one in the case of more than one meaning, but there are many concepts not corresponding to a particular word. How the generator can produce natural language from this representation is described in Section 3.4.3.

The LOLITA *semantic network* currently comprises around 100,000 nodes and it is continuously being extended (e.g by the addition of linguistic information to enable LOLITA to cope with Spanish).

2.1.2 LOLITA applications

Analysis of texts

The main operation of LOLITA is to analyse text in order to build a representation of its meaning. The information added from the input is identified in or added to SemNet. An example of parsed tree and resulted portion of semantic network is seen in Figure 2.3

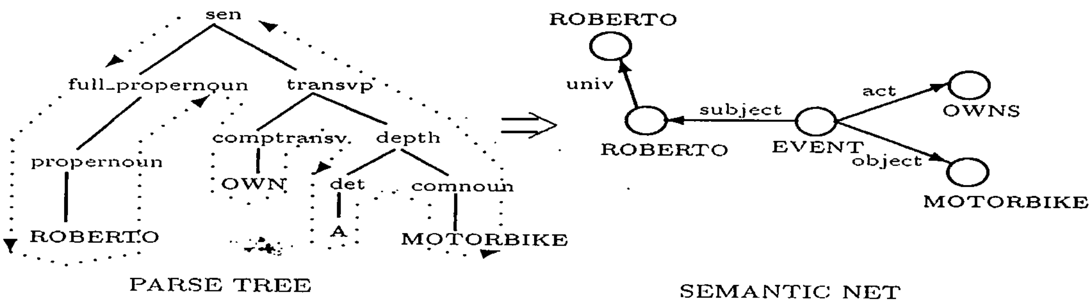


Figure 2.3: A fragment of SemNet.

Query

This application allows the users to interrogate LOLITA using NL utterances about the knowledge it holds.

Contents scanning

Contents scanning in LOLITA [Garigliano *et al.*, 1993] involves the analysis of texts and the completion of templates to summarise the information identified in the input texts. Contents scanning is a standard test for natural language systems [Long and Garigliano, 1994]. An example of contents scanning in LOLITA is presented in Figure 2.4.

The input is parsed and semantically analysed and then the representation of its meaning is stored in SemNet. A domain dependent module then searches the network in order to find the relevant information for each of the template slots.

A car bomb exploded outside the Cabinet Office in Whitehall last night, 100 yards from 10 Downing Street. Nobody was injured in the explosion which happened just after 9 am on the corner of Downing Street and Whitehall. Police evacuated the area. First reports suggested that the bomb went off in a black taxi after the driver had been forced to drive to Whitehall. The taxi was later reported to be burning fiercely.
(THE DAILY TELEGRAPH 31/10/92)

```
Template: Incident
Incident: A bomb explosion.
Where    : On the corner of Downing Street and Whitehall.
           Outside Cabinet Office and outside 10 Downing Street.
           In a black taxi.
When     : 9pm.
           Past.
           Night.
           When a forceful person forced a driver to drive a
           black taxi to Whitehall.
Responsible:
Target: Cabinet Office.
Damage: Human: Nobody.
        Thing: A black taxi.
Source: telegraph
Source_date: 31 October 1992
Certainty: Facts.
Relevant Information
         Police evacuated 10 Downing Street.
```

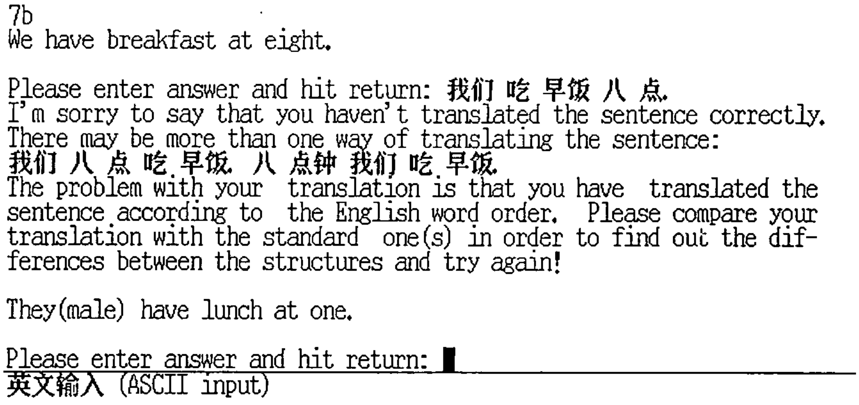
Figure 2.4: Example of the contents scanning application.

This information, in the form of SemNet nodes, is then passed to the generator which produces the output. Recent work has concerned the use of the LOLITA scanner with domain independent templates.

Chinese tutoring

This application [Wang and Garigliano, 1992][Wang, 1994] involves tutoring English-speaking students learning Chinese. One of the main problems in this learning process is the influence of the mother tongue, known as *negative transfer* [Selinker, 1969].

Chinese grammar rules have been added to the existing English parser to produce a mixed grammar. The tutoring module asks the student to translate an



```
7b
We have breakfast at eight.

Please enter answer and hit return: 我们 吃 早饭 八 点.
I'm sorry to say that you haven't translated the sentence correctly.
There may be more than one way of translating the sentence:
我们 八 点 吃 早饭. 八 点钟 我们 吃 早饭.
The problem with your  translation is that you have  translated the
sentence according to  the English word order.  Please compare your
translation with the standard  one(s) in order to find out the dif-
ferences between the structures and try again!

They(male) have lunch at one.

Please enter answer and hit return: █
英文输入 (ASCII input)
```

Figure 2.5: An example output from the Chinese Tutor

English sentence into Chinese. This translation is then parsed using the grammar rules. In the case of failure the English parser is used in order to identify transfer errors.

Translation

A small prototype has been developed by adding a few rules to the LOLITA parser to enable it to cope with Italian texts, enabling the system to add the information contained in the text to SemNet. Since the generator can produce English [Smith, 1995], translation from Italian to English is produced.

The completion of the work described in this thesis allows LOLITA to perform translation from Italian or English to Spanish.

Dialogue

A prototype has been developed which enable the user to carry out a dialogue with LOLITA [Jones, 1994]. The dialogue system takes into account factors such as the emotional state of the participants, motives, and the relationship of one participant to the other.

The dialogue module utilises a *Dialogue Structure Model* [Garigliano and Jones, 1992], which is a template-like schema containing the basic information about the dialogue situation. This information is used to structure the responses that LOLITA generates.

```
YOU:    I like you.
LOLITA: I know.
YOU:    You are beautiful.
LOLITA: I am glad because you think I am beautiful.
YOU:    Do you like me?
LOLITA: Yes, I like you.
YOU:    Why do you like me?
LOLITA: I do not know why I like you.
YOU:    I am tired.
LOLITA: I understand.
YOU:    Are you tired?
LOLITA: No.
YOU:    Tell me if you believe that I am tired.
LOLITA: Yes, you are tired.
```

Figure 2.6: Example of a dialogue with LOLITA.

2.1.3 Role of generation in LOLITA

Generation is involved in most of the applications built onto the LOLITA system. The basic operation of the generator is to rebuild surface language expressions from SemNet. As well as being used by most of the applications, these expressions are useful for the development of the semantic network and for debugging purposes as they are easier to understand than the SemNet representation.

- Query. The generator produces NL utterances for answering questions as well as utterances for the original questions.

- Content scanning. The template filling module will require the generator to build NL utterances in order to fill the slots of the template.

- Translation. The generator will rebuild language expressions from a semantic representation corresponding to an input text in a language other than the one generated.

- Dialogue. The generator produces NL utterances. The dialogue module interfaces with it to generate appropriate responses.

2.2 Natural Language Engineering

The development of the LOLITA system is concerned with Natural Language Engineering (NLE) rather than the more traditional computational linguistics. The field of Natural Language Engineering is composed of a number of interconnecting disciplines. It is an engineering activity and is thus pragmatic by nature, though its scientific and technical background is based on Descriptive and Computational Linguistics, Lexicology and Terminology, Formal Languages, Computer Science, Software Engineering and other relevant subject areas.

There are only a small number of systems which have the properties of a large-scale system compared with the great number of smaller systems performing specific tasks in defined domains. Whereas the central ideas formulated by computational linguistics have been successfully applied to small systems, difficulties have been experienced in their application to large-scale systems (those not highly restricted in their task or domain).

The following subsection will describe important properties of NLE according to which LOLITA has been developed.

2.2.1 Aspects of NLE

- **Scale** - The size of the system must be sufficient for supporting realistic large-scale applications (i.e. vocabulary size, grammar coverage).
- **Integration** - The system components should be built so that they can be combined with the system as a whole. These components should not make unreasonable assumptions about other parts of the system.
- **Feasibility** - For example, hardware requirements must not be too great and the execution speed must be acceptable. This process includes making the system efficient.
- **Maintainability** - The usefulness of the system over a long period of time must be ensured.

- **Flexibility** - The system has to be able to perform different tasks in different domains. The applications described (section 2.1.2) give an example of the flexibility of LOLITA.

- **Usability** - The solution of the system must fulfil the requirements of the user. This solution should be *user-friendly.*

- **Robustness** - This is a critical aspect of large-scale systems. Robustness concerns not only linguistic scope but also the ability of the system to deal with incorrect input without crashing.

- **Use of a wide range of techniques** - Systems following the NLE approach should use a full range of AI techniques. This implies the use of long-standing, reliable, general or localised theories from computational linguistics and logic (i.e. set-based semantics), knowledge based approaches, individual heuristics and adaptative or evolutionary techniques.

2.3 Functional Languages

LOLITA is written in the functional language Haskell [Hudak *et al.*, 1994]. The system consists of over 40.000 lines of source code equivalent to about 400,000 lines of imperative code [Turner, 1982].

Functional languages are a subset of the Declarative programming languages. The main feature of a Declarative language is that it has no 'implicit state' (global variables, program counter, etc). Any information needed must be handled explicitly.

A program consists of *expressions*, instead of *sequencing of commands* as in imperative languages. It has often been argued that it is easier to write in a functional programming language than in an imperative language.

Declarative languages are subdivided into the following types:

- Specification: (i.e. Z,VDM) They are not used to program, but to specify the behaviour of a system.

- Logical: (i.e. Prolog) They use the concept of *relations (predicates)*.
- Functional: They use the concept of *functions*.

Haskell is a pure functional language with non strict semantics (i.e. lazy evaluation) and a polymorphic type-checking system. It was developed following a conference in 1987 as the definitive functional language. The next subsection will present the features of functional programming, and more specifically Haskell, which are different from other programming languages. The effect of these features on the implementation of the generator will be discussed in Chapter 4.

2.3.1 Features of Functional Languages

Referential Transparency:

Referential transparency rules out the use of assignment statements and the explicit concept of a program state based on the values of variables and constants avoiding side-effects. This means that the value of an expression depends solely on the values of its subexpressions and there are no hidden effects influencing its value. Also, different occurrences of the same variable always have the same value, unlike in imperative languages, where a variable may be assigned several different values within an expression.

This property makes functional programs easier to be understood and easier to be developed, therefore allowing a better integration of the system.

Function Application and Currying

A factor which improves readability is the syntax of function application in Haskell. The operation of function application is represented by simple juxtaposition of the function and its arguments. Thus a function `f` applied to two arguments `x` and `y`, represented in most imperative languages as `f(x,y)` is represented in Haskell as `f x y`. This enables a program to use far fewer brackets. Associated with this is a device known as *currying*. Currying involves the replacement of structured arguments with a list of simple ones. We shall take the example of the function `plus`. This function gives the sum of two numbers. Consider the two definitions:

```
plus' (x,y) = x + y z
```

and

```
plus x y = x + y
```

In an ordinary imperative language, the definition `plus'` would be used. However, Haskell also allows the definition `plus` to be written. The difference is that `plus'` takes the single, structured argument of a tuple of two numbers; the function `plus` takes two simple arguments. One can therefore write `plus 1 2` which is equivalent to the expression `1 + 2`. Function application in Haskell is left associative; `plus 1 2` is therefore interpreted as `((plus 1) 2)`. Thus the expression `(plus 1)` is a function in its own right—it takes a single argument and adds 1 to it. Without currying, the function to add 1 to a number would have to be written as a separate, new function. This simple but useful feature allows functions to be greatly simplified merely by leaving out arguments when they are not necessarily required, thus aiding readability. Currying therefore allows parameter hiding in abstract types.

Higher-order Functions:

Higher-order functions are functions which take other functions as part of their input or return functions as results.

The definition of the **map** function is shown below as an example of a higher-order function. 'map' takes as its arguments a function and a list of elements. It returns a list containing the results of applying the given function to each of the elements of the input list.

```
map f []     = []
map f (x:xs) = f x : map f xs
```

Abstract Types:

Abstract Data Types (ADTs) are data types whose representation is hidden to the rest of the system. They can only be accessed using a set of provided functions. The data types can be modified without affecting the parts of the system using them.

Lazy Evaluation:

Lazy Evaluation allows an expression to be evaluated when its value is actually needed. That is, the expression is evaluated on demand. So lazy evaluation allows unevaluated expressions to be passed to functions as parameters and if the value of a expression is not being used the expression will not be evaluated.

Lazy evaluation also allows the programmer to handle very large (even infinitely large) expressions when complete evaluation of them is not required. For example in a search problem it is possible to build all the possible solutions (even an infinite number of them) and then a set of functions to decide on the chosen solution.

2.4 Chapter Summary

This chapter has presented the main aspects influencing the development of the Spanish generator (the current LOLITA generator will be described in the next chapter).

The generator will be integrated in LOLITA. This NLP system has been described paying particular attention to the semantic network (which forms the knowledge representation of the system) as it is the core of the whole system and more especially the input for the generator. Prototype applications developed 'on top' of LOLITA have been introduced to show the capabilities of the system. Some of these applications (Query, Content scanning, Translation, Dialogue) make use of the generator to perform their operations.

The rest of the chapter has presented aspects affecting not only the generator but the whole system. The LOLITA system has been built within the domain of

Natural Language Engineering (NLE) so NLE principles have been adopted for the development of LOLITA.

Functional languages and particularly Haskell, together with its features, have been introduced. Chapter 4 will show how these features have been used in the implementation of the Spanish generator.

Chapter 3

Related Work

Natural Language Generation (NLG) is a subfield of Natural Language Processing (NLP) but its boundaries are not easy to define as researchers define the NLG task differently depending on the input received and the processes performed. Following these different views, diverse approaches have been adopted to cope with the different aspects and assumptions considered in the generation of natural language. However, two stages are commonly identified during the natural language generation process; planning and realisation, but the tasks carried out in each stage differ from system to system. This chapter discusses aspects such as the input a generation system assumes, different architectures adopted to build them and approaches to realisation, planning, and the interface between the modules (the problem of the generation gap). The chapter also pays special attention to those systems which receive the same type of input as the LOLITA generator, a semantic network or graph. Finally the generation module in LOLITA is introduced.

Some criticism applied to the field of NLG in general (this does not mean that all these criticisms apply to all the systems) are:

- Systems tied to limited domains
- Small scale of the systems
- Restriction to a particular natural language
- Lack of information in the form of example outputs

3.1 Input

One of the most important factors which determine a generator's characteristics is the input it assumes. The type of input delimits the tasks a generator must perform. Generation systems can be roughly split in two groups: systems assuming the content as a side effect of the application program (typical of 'active' programs such as simulations and expert systems) and systems taking on the responsibility of extracting the content from the application program (typical of 'static' programs such as databases).

The most common inputs can be classified in the following types:

- Input containing a knowledge base and a communication goal. The generator module must retrieve knowledge from the knowledge base according to the goal.

- Input in the form of clause-sized chunks. The generator must order the clauses into sentences.

- Input assuming a detailed specification of the utterance. The tasks to be performed by the generator differ from system to system depending on the level of detail of the specification. These tasks comprise grammatical and lexical choices.

- Input containing a semantic representation of the information to be generated in the form of a network or graph. This is the type of input assumed by the LOLITA's generator.

3.2 Architecture

Generation involves three main activities: *determining the information to communicate, ordering this information* and *realising the information in a surface NL.* These three activities have generally been divided in two processes:

- ‘what to say’: involving the two first activities. This part is commonly called the **Text Planning** component.
- ‘how to say it’: it involving the third activity. This component is commonly referred as the **Realisation** module.

Other terms used to refer to this division are: *Strategic* and *Tactical* levels, *Deep* and *Surface* generation, *Text planning* and *Plan execution*, *Message* and *Form* levels, *Functional* and *Positional* levels and *Conceptual* and *Grammatical* levels.

Following this distinction between planning and realisation modules, [Kantrowitz and Bates, 1991] claim that there are two types of generation architecture: *integrated* systems and *separated* systems, with the latest type subdivided into *pipelined* and *interleaved* systems.

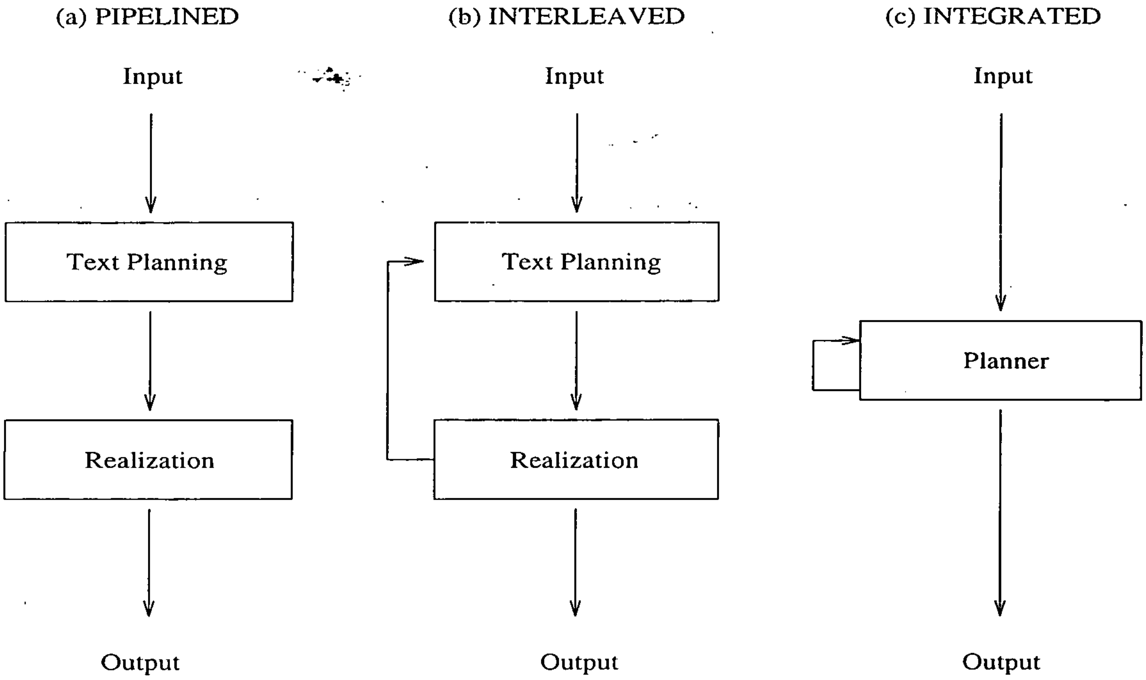


Figure 3.1: Generator architectures

Integrated Systems

Some researchers argue against the modularisation of a generator into text planning and realisation. Systems following an integrated approach try to overcome the problems derived from the interface between planner and realiser by integrating both modules in one component.

Examples of integrated generators are:

- KAMP [Appelt, 1985]
- DIOGENES [Nirenburg *et al.*, 1988]
- GLINDA [Kantrowitz and Bates, 1992]

Separated Systems

Most of the generation systems follow a separated components approach. The first module, the planner, selects and order the information to be generated in terms of the input received. The realiser determines which linguistic resources will be used for expressing the information. The problem arises when following this approach because semantic and syntactic structures are not isomorphic [Elhadad and Robin, 1992] so the interface between planner and realiser is not a trivial step. This problem is called 'generation gap' [Meteer, 1993].

There are two different approaches to the interface between the two components:

- Pipelined systems. In this systems the planner makes decisions indelibly. Therefore, it must assure that the decisions it makes can be realised in the surface language.
- Interleaved systems. These systems use a backtracking mechanism or constrain the process by passing information and control between the components.

3.2.1 Planning

Early computational systems, from the 1970's to the beginning of the 1980's, generating multi-sentence text ignored the issue of text structure [Hovy, 1993] and some of the more modern generators are still domain restricted and often rely on domain dependent organisations to plan their discourse.

However, two common methods for the structuring of text above the level of sentence are used in a variety of systems: *text schemas* and *rhetorical structure theory* (RST).

The Schema Approach

Text schemas describe conventional textual structures in terms of patterns which specify the overall structure of a text.

McKeown's TEXT system [McKeown, 1985] was developed to produce paragraph length texts in response to users' queries about an underlying knowledge base. This knowledge base contains information about military vehicles and weapons. TEXT uses its knowledge about 'discourse strategies', represented in predefined *schemas*, and follow the assumption that people use certain discourse patterns to express certain discourse goals. Therefore, for each 'communicative goal' the system might have, there is a corresponding schema representing a discourse strategy. The schemas are made up of 'rhetorical predicates' such as "identification", "attribution", "analogy". An example of a schema and the output generated from it by the TEXT system is shown in Figure 3.2

More examples using a schema based planner are ANA [Kukich, 1988], EDGE [Cawsey, 1990], WEIBER [Horacek, 1990] and TAILOR [Paris, 1993].

Rhetorical Structure Theory

Rhetorical Structure theory (RST) was initially developed for the descriptive analysis of relations in text [Mann and Thompson, 1987]. RST is a descriptive formalism that attempts to capture the organisation of natural text through the relations that hold between parts of the text. The relations in RST are embodied in schemas. Each schema consists of a NUCLEUS and zero or more SATELLITES whose function is to support the nucleus. Satellites are linked to the nucleus by a RELATION which indicates how the satellite supports the nucleus. A satellite can also be discomposed into a nucleus and satellites of its own. Each relation

SCHEMA

```
Constituency
Cause-effect*/Attributive*/
  {Depth-identification/Depth-attributive
    {Particular-illustration/evidence}
    {Comparison/analogy}}+
  {Amplification/Explanation/Attributive/Analogy}
```

'{}' indicates optionality,'/' indicates alternatives, '+' indicates that the item may appear 1-n times, and '*' indicates that the item may appear 0-n times.
EXAMPLE:

> "Steam and electric torpedoes. (1) Modern Torpedoes are of 2 general types. (2) Steam-propelled models have speeds of 27 to 45 knots and ranges of 4000 to 25,000 yds.(4,367-27,350 meters). (3) The electric powered models are similar (4) but do not leave the telltale wake created by the exhaust of a steam torpedo"

CLASSIFICATION OF EXAMPLE:

1. Constituency
2. Depth-identification; (depth-attributive)
3. Comparison
4. Depth-identification; (depth-attributive)

Figure 3.2: The constituency schema.

has constraints on the nucleus, constraints on the satellite(s), constraints on the combination of nucleus and satellite(s) and an effect. These constraints have to be satisfied before a relation can be applied to a text. The complete analysis of a text is a tree with a single schema at the top level.Figure 3.3 shows an example of a RST schema from the PENMAN system.

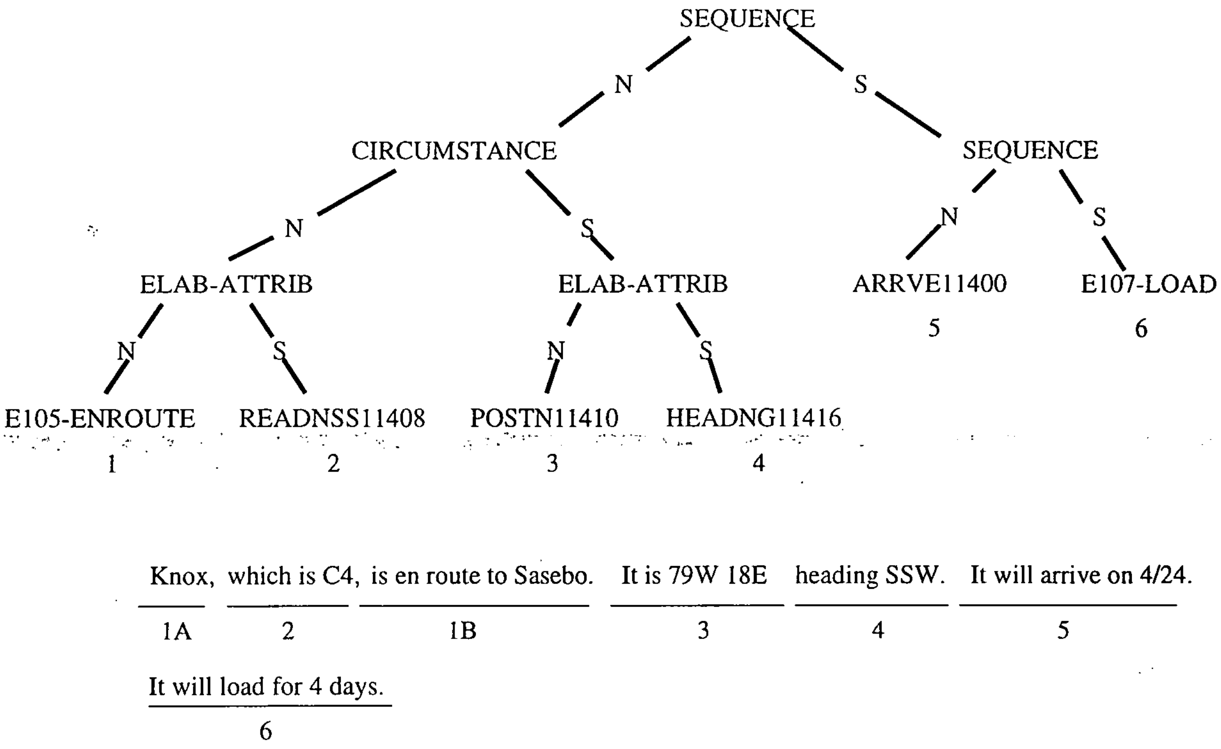


Figure 3.3: A RST schema from PENMAN.

Using RST for generation, an abstract specification of the utterance has to be provided via a discourse goal and the planner must choose what information to include. Some of the work using RST based planners include:

- Hovy [Hovy, 1991] was one of the first researchers to apply the descriptive RST formalism for building a text structure planner. The planner assumes as input one or more communicative goals and a set of clause-sized predicates. It proceeds by recursively applying RST relations, whose effects match the communicative goals, to units of the input and other RST relations in order to build a tree which represents the paragraph structure (non-terminals are RST operators, the leaves are the input predicates). The final tree is traversed from left to right forming the input for the sentence generator NIGEL.

- The Explainable Expert System (EES) [Paris *et al.*, 1991] uses an RST-based text planner to construct short explanatory dialogues of an expert system.

- Other examples of work which uses text planning based on RST can be found in [Scott and de Souza, 1990], the COMMUNAL project [Fawcett and

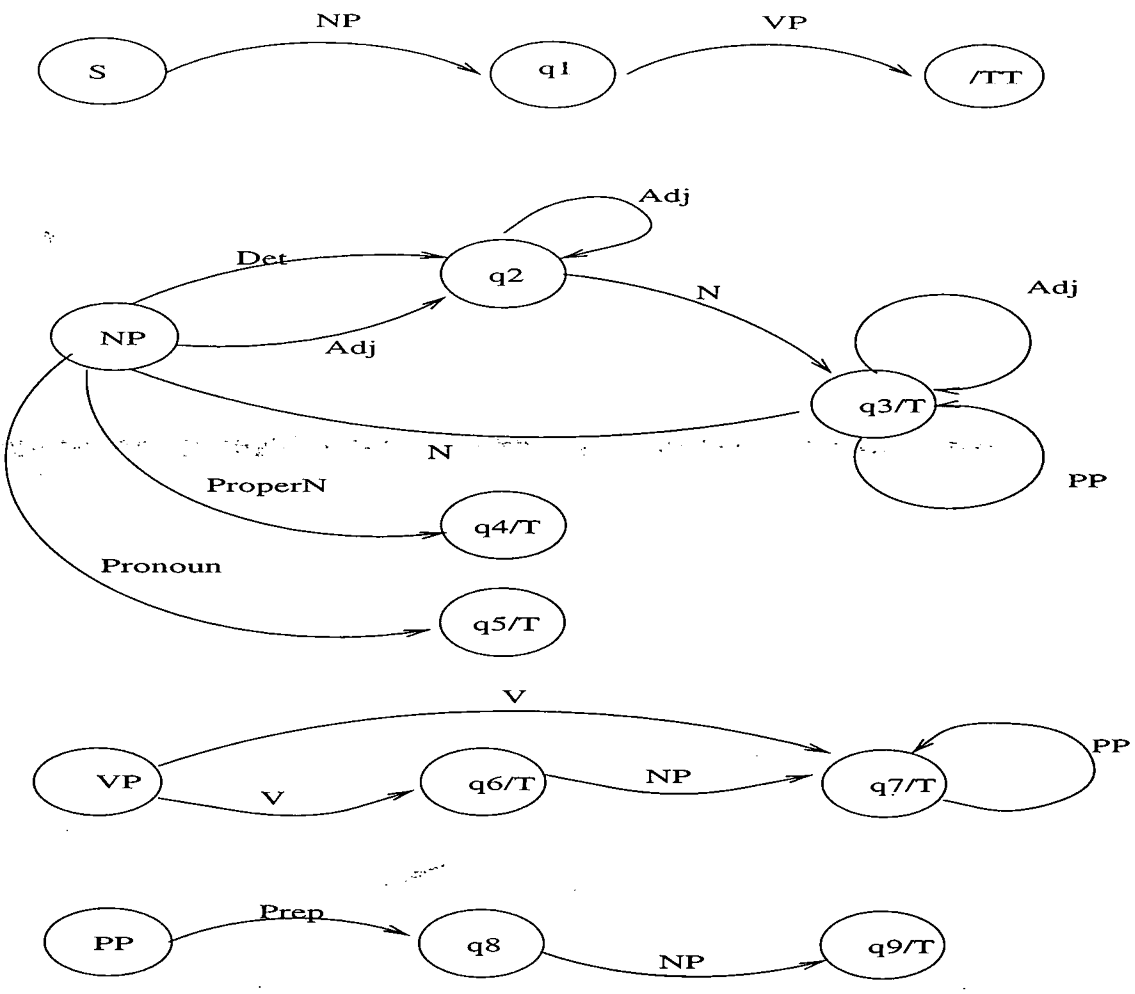
Tucker, 1992], TECHDOC [Rösner and Stede, 1992], IMAGENE [Vander-Linden et al, 1992], [Granville, 1994], [Delin *et al.*, 1994] and [Wanner, 1994].

3.2.2 Realisation

Internal representations are mapped into surface NL at the realisation stage. The module must contain linguistic information of the resources of the surface natural language. That is, grammatical rules of that language and a lexicon, as well as a mechanism to produce correct text from the input representation by applying the linguistic information. The type of input, the control of the process and the internal organisation of the module are some of the factors that determine the method to apply at the realisation stage. Some of them are presented below.

Augmented Transition Networks

The Augmented Transition Networks (ATNs) [Woods, 1970] is a formalism for writing parsing grammars. ATNs were originally developed for NL analysis but they have also been used in generation. Some of the researchers who use ATNs in generation are Simmons and Slocum [Simmons and Slocum, 1972], Goldman [Goldman, 1975], Shapiro [Shapiro, 1982] and McKeown [McKeown, 1985]. Figure 3.4 shows a context free grammar and its ATN representation.



S → NP + VP

NP → (DET) + (ADJ*) + N + ((ADJ*)+(PP*)*)

NP → ProperN

NP → Pronoun

VP → V + (NP) + (PP*)

PP → PREP + NP

notation: () :optional * :one or more occurences

Figure 3.4: An ATN and associated grammar.

Use of ATNs has been particularly common in systems which receive the input in the form of a semantic network or graph (Section 3.3).

Functional Unification

Realisation modules following a functional unification method combine a unification grammar approach ([Kay, 1979]) with a unification formalism.

A unification grammar contains the descriptions of particular linguistic objects in the form of *functional descriptions* (FDs) which associate values with 'objects' features. The input to the realisation process is another FD which specifies the content of the required utterance. The process is performed by 'unifying' the input FD with the grammar description. The unification of two FDs merges the features from both of them to produce a more specific FD. The main disadvantage of this approach is that the process of unification is non-deterministic and therefore inefficient. A simple example of unification is shown in Figure 3.5.

```
FD1 = {article:{definite:yes}, head:{lex:'cat'}}

FD2 = {article:{lex:'the'}, modifier:{lex:'black'}}

unify(FD1,FD2)=  {article:{definite:yes, lex:'the'},
                  head:{lex:'cat'},
                  modifier:{lex:'black'}}
```

Figure 3.5: A simple example of unification of two FDs.

Some of the systems which use a unification approach are:

- A functional unification grammar (FUG) was used for the realisation stage of the TEXT system [McKeown, 1985].
- Appelt's TELEGRAM [Appelt, 1983] modified the unification process by allowing the planner to be re-invoked at various choice points in the grammar in order to overcome the problem of efficiency caused by the non-determinism of this approach.
- McKeown *et al.* Functional Unification Formalism (FUF), used in their COMET system is an expansion of functional unification grammars. They

expanded the idea of FUG grammars to include a unification stage for lexical selection and for deciding when to explain information graphically or textually.

Systemic Grammars

Systemic functional linguistics [Halliday, 1985] divides language not just into syntax and semantics but on three functional lines of analysis : [Meteer, 1993]

- ideational: the content of the utterance and the organisation of the speakers experience in terms of processes, things, qualities, etc.
- interpersonal: The relation of the speaker and hearer.
- textual: The organisation and cohesion of text.

A systemic grammar has two basic components - a network of *systems* and realisation rules (related to the functional lines of reasoning listed above). The systems of the network represent a choice point where a feature must be selected from a set of alternatives. Realisation rules are used to decide the selection.

The main generative systemic grammars in existence are: NIGEL [Mann, 1983a] (the systemic grammar of the PENMAN project [Mann, 1983b]), GENESYS [Fawcett and Tucker, 1992] (part of the COMMUNAL project) and SLANG [Patten, 1988].

Use of a Formative Lexicon

Another approach to realisation is to contain formative information in the lexicon. In some systems the lexicon groups grammatical rules and lexical information together (for example PAULINE [Hovy, 1988b]) relating to the combination of particular words with others. In other systems information about semantic relations among the lexical entries is also included in the lexicon (i.e. MTM).

A problem adopting this approach is that of scale. If a system is limited to a domain with a large number of lexical entries then inclusion of formative and semantic information for every entry may prove unrealistic.

MUMBLE

McDonald's linguistic realisation component MUMBLE [McDonald and Bolc, 1988] uses the linguistic specification of the input message (MUMBLE is message directed, see Section 3.3.1) to build the surface structure of the text. McDonald also developed a specification language which allows interfacing MUMBLE to some underlying programs and planners (i.e. those using the SPOKESMAN representation). Another feature of the MUMBLE component is that it relies on indelible processes, so decisions cannot be retracted once they have been made.

MUMBLE's grammar is based on a Tree Adjoining Grammar (TAG, [Joshi, 1987]) defined in terms of elementary trees and rules for their composition. The generation process comprises three subprocesses:

- **Attachment**: assigns plan units to positions within the tree representing the surface structure. This surface structure has *attachment points* to which new structures can be added. The attachment is performed according to various grammatical constraints and stylistic rules.
- **Phase Structure Execution** (PSE). After a partial tree has been attached, the PSE takes over. PSE performs a depth first traversal of the tree performing transformations or invoking syntactic constraints indicated by tree labels. If plan units are found in the tree then realisation is invoked to determine how they should be realised. If an attachment point is encountered then Attachment is called to determine if there are additional plan units to be attached at this point.
- **Realisation**: Realisation selects appropriate words or phrases to 'realise' plan units.

3.3 Other Aspects of Generation

3.3.1 Control

Meeter [Meteer, 1993] distinguishes between two types of control:

- In a *grammar directed* (or *declarative* [Paris and McKeown, 1987]) system, control lies in the reference knowledge, that is, it is governed by some predetermined body of tests that gate and order actions (i.e. the NIGEL realiser).
- In a *message directed* (or *procedural*[Paris and McKeown, 1987]) system, control lies in the input itself and is interpreted by some general control loop within the process (i.e. SPOKESMAN).

Message directed processing is considered more efficient [McDonald *et al.*, 1987] as the action sequence is already implicitly determined by the process that built the input and no effort needs to be expended on control decision. Other researchers (i.e. [McKeown and Swartout, 1988]) arguing against this approach, say that the procedural control affects the clarity of a system and the absence of a explicit grammar makes the grammatical process more difficult to understand, judge and modify.

A declarative control is more typical in generators with static underlying process or those that receive inputs such as clause size predicates. On the other hand, a procedural control is commonly adopted by generators with active underlying programs or input containing rich information (i.e temporal or causal).

3.3.2 Generation Gap

As stated in Section 2.2, the separated components approach leads to the problem at the interface between components. This problem arises because semantic and syntactic levels are not isomorphic. It has been called the 'generation gap' problem. This section described how the generation gap has been tackled in the SPOKESMAN planner.

SPOKESMAN

Meeter [Meteer, 1993] first named this problem the 'generation gap' and she has been the researcher who has tackled the problem in most detail.

The SPOKESMAN planner is built on the MUMBLE realiser. It addresses the problem of the expressibility: as the system relies on indelible processes, once a decision is made it can not be withdrawn or modified. The text planner must assure that it will not compose an utterance that cannot be realised in the surface language. Meeter overcomes the problem of the generation gap and fills the expressibility requirements by designing an intermediate level of representation, the *Text Structure*, which is used by the planner in composing the utterance. The input objects drive the building of the text structure using mappings associated with their types. The text structure also takes part in its own construction by constraining further decisions depending on the decisions already made.

Text Structure is represented as a tree capturing the following kinds of information:

- **Constituency**: The nodes in the Text Structure represent the constituents of the utterance.
- **Structural relations among constituents**: The relations of each node with its parent and children.
- **Semantic Category of the constituents of the utterance.**

The Text Structure plays two important roles:

-It provides a bridge between the structures of the application program and the linguistic structures needed by the realisation component.

-It constrains the planning process to ensure that what is planned is *expressible* in language.

Once the text structure has been built using the input information, the tree is traversed to build the *linguistic specification* required by the MUMBLE realiser. Figure 3.6 (from [Meteer, 1993]) shows an example of Text Structure.

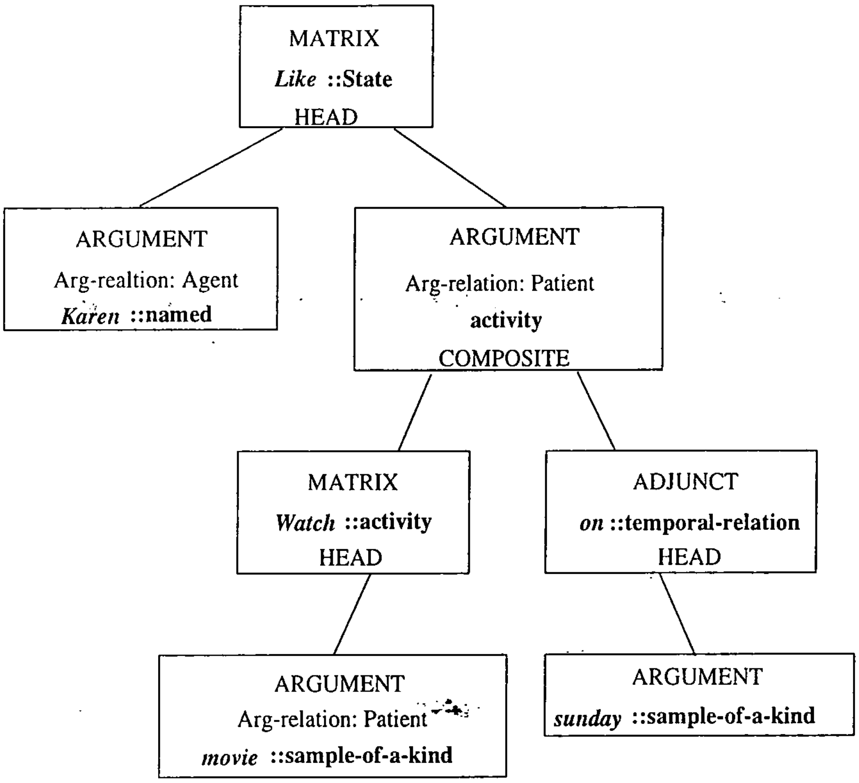


Figure 3.6: Text Structure for "Karen likes watching movies on Sundays"

3.4 Generating from Semantic Networks or Graphs

As stated in Section 2.1 the input is a critical aspect determining the further design of the system. Similar input often means similar design. Some of the advantages of taking a semantic network or graph as input are as follows:

- Such input can be a knowledge rich representation.
- The input allows for a message directed control approach which is often more efficient (see Section 2.2.3)
- The knowledge rich input can lead to the variation in the utterances which can be achieved separately from the realisation process.

This section describes some of the systems that take a simular input to the LOLITA generator.

3.4.1 Generating from CDT

Schank's Conceptual Dependency Theory (CDT) adopts a method of *knowledge representation* that intends to capture the meaning underlying NL utterances. CDT uses semantic primitives to compose meaning representations. In particular, actions are decomposed in a small set of *primitive acts* such as INGEST, ATRANS (movement of a physical object) and MTRANS (movement of a non-physical object) are three of the twenty four primitive acts. Schank later extended such a restricted set of primitives and defined higher level primitives which however was too restricted for a large scale system.

BABEL

The BABEL system [Goldman, 1975] produces English sentences from Schank's CDT. Goldman uses word-sense *discrimination networks* to make word choices. The discrimination networks are binary trees whose nodes comprise predicates which determine the child path to follow.

The first step is to find the main verb to express the CDT representation by applying discrimination networks. At the leaves of the trees are pointers to *concexion* entries (or to other nodes which will lead to more concexions) which are used to build a syntactic network of the utterance. This is realised into a NL utterance using an ATN (Section 1.2.2).

PAULINE

Hovy's PAULINE (Planning And Uttering Language In Natural Environments) [Hovy, 1988b] aims to produce different output text from the same semantic input (in the form of CDT representation) according to parameters which describe pragmatic settings. These include conversational settings, speaker and hearer's characteristics and the relationship between them. Depending on the value of these parameters the output text is constrained in different ways. However, Hovy considers that these factors are too general and describes some *rhetorical goals* in order to provide rules to constrain the generation. Examples of these rhetorical goals are **formality** (with associate values: highfalutin, normal, colloquial), **simplicity** (simple, normal, complex), **force** (forceful, normal, quiet), **colour** (facts only, with colour) and **respect** (arrogant, respectful, neutral, cajoling).

The values of the rhetorical goals are determined on the basis of supplied values of the conversational parameters. Figure 3.7 shows an example of highfalutin and informal texts generated by PAULINE(the rhetorical goals affect decisions such as the content selection, sentence organisation and word choice).

HIGHFALUTIN:

> "In early April, a shanty-town - named Winnie Mandela city - was erected by several students on Beinecke Plaza, so that Yale University would divest from companies doing business in South Africa. Later, at 5:30 AM on April 14, the shanty town was destroyed by officials; also at that time, the police arrested 76 students. Several local politicians and faculty members expressed criticism of Yale's action. Finally, Yale gave the students permission to reassemble the shanty town there and, concurrently, the university announced that a commission would go to South Africa in July to investigate the system of Apartheid."

INFORMAL:

> " Students put a shanty town, Winnie Mandela City, up on Beinecke Plaza in early April. The students wanted Yale university to pull their money out of companies doing business in South Africa. Officials tore it down at 5:30 on April 14, and police arrested 76 students. Several local politicians and faculty members criticised the action. Later, Yale allowed the students to put it up there again. The university said that a commission would go to South Africa in July to study the system of Apartheid."

Figure 3.7: Example of a formal and informal text produced by PAULINE

3.4.2 Generation from Conceptual Graphs

Conceptual Graph (CG) [Sowa, 1984] is a knowledge representation that was primarily developed by Sowa aiming to represent natural language semantics.

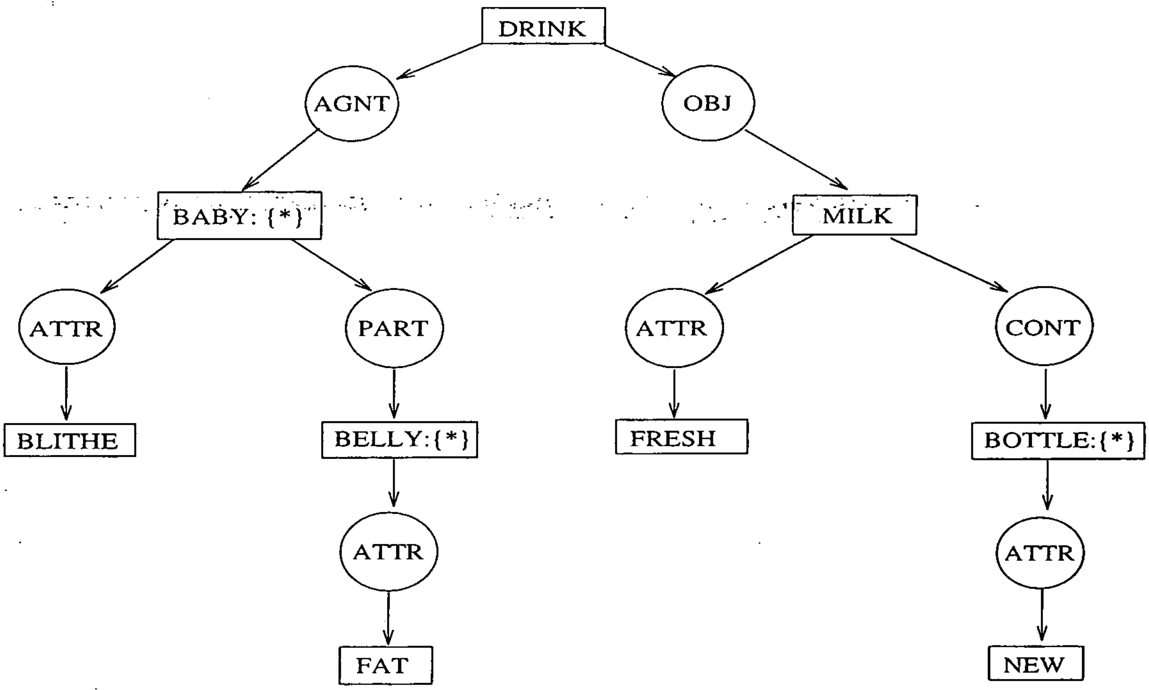


Figure 3.8: Example utterance graph input

A CG is a finite, connected graph with nodes representing either concepts or conceptual relations that relate two concepts. Figure 3.8 shows an example of CG. The concepts are represented by boxes and the relations by circles.

Sowa's work in generation

Sowa's generation process concerns the mapping of conceptual graphs into words. The sequence of nodes and arcs traversed in mapping a graph to a sentence is called the *utterance path.* For complex graphs, the utterance path may visit a concept more that once and depending on the type of language (pre-order, postorder or in-order language) words should be produced at the first, last or some intermediate visit to the node. However, independently of the number of visits, a concept can only be uttered at one of the visits.

The same graph can be expressed in many different sentences, depending on the starting point and direction of the utterance path. For example, the following sentences can be generated from the graph of Figure 3.8:

```
Blithe babies with fat bellies drink fresh milk in new bottles
Fresh milk in new bottles is drunk by blithe babies with fat bellies
Blithe babies that drink fresh milk in new bottles have fat bellies
Drinking fresh milk in new bottles is done by blithe babies with fat
bellies
```

As not all word orders are possible, Sowa defines six universal grammar rules for translating a CG into a sentence. These rules are claimed to be language independent and are complemented by language dependent rules. These rules decide which arc to follow when there is a choice and also insert function words and word inflections. These grammar rules are encoded in an *Augmented Phrase Structure Grammar* (APSG). APSG is an extension of a context-free grammar augmented with conditions to be tested and actions to be performed. The rules are applied in a top-down goal directed manner.

Other CG generation works are:

- Nogier and Zock's work [Nogier and Zock, 1992] on generation is used in the information retrieval system Kalipsos.
- Dogru and Slagle [Dogru and Slagle, 1992]
- Rijn [van Rijn, 1992] uses a special kind of graph called a conceptual dependency graph which contains low-level primitives.

A common problem in the systems cited above is that of the simplicity of the input graphs (i.e can be expressed as one sentence).

3.5 Generation from SNePS

The SNePS (Semantic Network Processing System) [Shapiro and the SNePS Implementation Group, 1993] is 'a knowledge representation and reasoning system that allows one to design, implement, and use specific knowledge representation constructs, and that easily supports nested beliefs, meta-knowledge, and meta-reasoning'.

With respect to generation, Shapiro describes a generalised ATN (see Section 2.2.2) that supplies consistent semantics for a combined parsing-generation grammar. This allows an ATN grammar to be constructed so that the 'parse' of a NL question is the NL statement that answers it. The goal of the generation part of this process is, given a node, to express the concept represented by that node as a NL surface string.

Another system based on SNePS is the KALOS system [Cline, 1994]. This system generates descriptions of the M68000 processor. It first produces very simple sentences and then a revisor component passes revisions suggestions back to the deep generator (for conceptual revisions) and the surface realiser (for stylistic revisions). The system uses SNePS representations to represent a Domain Knowledge, to encode a simple schema approach (see Section 2.2.1), to represent the grammar rules of a unification based approach (see Section 2.2.2) and to perform revision stages. A weakness of the system is that the domain and application are very restricted.

3.5.1 Generation from MTM

The Meaning Text Model (MTM), which is based on the Meaning Text Theory [Mel'čuk and Polguère, 1970], is a lexicon-based approach which describes the bidirectional mapping between linguistic meanings and texts which carry those meanings.

During the generation process, this approach assumes that nodes and arc labels in the semantic network input correspond directly to lexemas. A MTM model contains a MTM lexicon which has a very rich lexical information which aim to cover all possible linguistic knowledge about constrains on the combinations of words. Output utterances are produced in MTM by performing transformations that restructure the network using the information contained in the lexicon.

A limitation in this approach is the assumption that the generator receives sentence sized portions of semantic representation as input.

The GOSSIP system (Generation of Operating System Summaries in Prolog) [Iordanskaja *et al.*, 1991] is based on the MTM approach. This system makes the assumption that some other process has built a sentence-sized semantic network from which it will realize a sentence. It also receives as input a communicative structure which marks the theme and rhyme of the utterance to be produced.

Generation comprises four transformation stages: Semantic network reductions, Root lexical node choice, Deep syntactic structure paraphrasing using lexical functions and surface realisation (choosing alternatives syntactic structures).

Another system following this approach is the Joyce system [Kittredge et al, 1991].

3.6 Current Generation in LOLITA

This section presents a description of the LOLITA generator [Smith, 1995], [Garigliano *et al.*, 1992] and its components. The generator receives as input the whole LOLITA semantic network (SemNet) and like the majority of the NLG systems, adopts a 'separated architecture'. However the distribution of tasks between the two components, *the planner* and *the plan-realiser*, differs from other systems as the plan-realiser can undertake tasks more commonly performed by planners.

3.6.1 Input

The generator in LOLITA receives the whole SemNet as its input. There are other systems with similar input but they generally delimit the content by 'cutting out' a portion of the net. In LOLITA, SemNet is accessible throughout the whole generating process. This is based on the assumption that the meaning of a node is represented by the whole of the semantic network.

3.6.2 Planner

The role of the planner in the LOLITA generator is to provide the plan-realiser with instructions about the content and style of the utterance to be produced. The content will be given by passing one or more references to nodes in SemNet. The instructions refer to issues outside the scope of the surface language. The reference to at least one node should be passed to the plan-realiser.

A completed planner has not yet been implemented; instead the operation of the planner is simulated. For this purpose the system has been provided with:

- **Operation methods** *Node by node* is an operation method where the planner is not required. The plan-realiser is passed SemNet with a reference to a particular node inside it and generates an utterance corresponding to the content of the node. This operation has been useful in the debugging and development of LOLITA as natural language utterances are easier to understand than the SemNet representation itself.

- **Realisation Parameters** *Realisation parameters* are switches which affect the manner in which the plan-realiser produces the utterance. They can be set by the planner (or simulated planner) or the underlying application. There are four types of realisation parameters:

  - Grammatical: affect the grammatical style. i.e. Active/Passive, Dative/Non-Dative.

- Style: affect the generic style. i.e. length of sentences, number of adjectives, use of synonyms.

-Content: Affect 'what' should be said in the utterance.

-Abstract Transformations: affect which abstract transformations should be produced.

- **Commands** *The 'story' command* allows the user to play the role of the planner. The user inputs to the plan-realiser event node references and a list of realisation parameters to be applied to the nodes. The user also provides information about each node such as 'Must describe separately'/'Must describe'/ 'May describe'/'Do not describe'.

Smith claims that a real planner with the features described above is achievable because:

- The planner is not always needed for the generation system and the demands on it need not necessarily be high. Some applications could by-pass the planner even if it did exist.
- An intermediate planner already exists for the dialogue application. It comprises two elements, a *template* defining the current situation in terms of dialogue structure elements and a *based reactive* element which models the 'individuality' in the dialogue situation.
- The planner will not have to find the optimal solution.
- The planner does not have to know linguistic details. The planner only makes decisions on a conceptual level.
- Other components of the LOLITA system will aid the planner in its tasks.
- A planner is already being developed using state of the art hierarchical abstraction planning methods [Long and Fox, 1995][Fox and Long, 1995].

3.6.3 Plan-Realiser

The module has been called plan-realiser to differentiate it from realisers in more traditional approaches. It may perform tasks traditionally performed by the planner in other systems (i.e. content selection, sentence organisation).

The role of the plan-realiser is to produce utterances in a surface language (the current generator is currently able to produce English utterances) following the instructions passed to it by the planner. As seen in the previous section these instructions will at least contain references to one or more nodes in SemNet.

The plan-realiser generates an expression for that node taking into account the rest of the planner instructions. If this set of instructions is not detailed or does not exist, then there are some default instructions to apply to the input nodes. So the plan-realiser can perform tasks that in other approaches correspond to the planner. In case of conflict in applying all the instructions the plan-realiser takes decisions by itself. It decides which instructions have more priority and it could even decide not to apply some instructions.

The plan-realiser must relate concepts, contained in the nodes, to lexical items corresponding to the surface language. Concepts have a smaller grain-size than words in the approach followed by LOLITA (see Section 2.1) so only some concepts have a link to a single lexical entry; they are named 'language-isomorphic' concept nodes. For those concepts not corresponding to a lexical entry the plan-realiser must search for 'language isomorphic' concepts in SemNet in order to adequately describe them. This search depends on:

- The current content of the network. SemNet is the input of the generator so the most influential factor. The search depends on what is actually present in the network (in terms of arcs and nodes). This represents the procedural control (see Section 3.3.1).
- Grammar. The plan-realiser contains grammatical rules that constrain the search in order to produce correct utterances. Obviously, there are different grammatical rules depending on the surface language.

- The realisation parameters. These parameters represent planner instructions and affect the order of the search across the arcs.

A more detailed solution of the plan-realiser will be given in the next chapter when describing the solution for the generation of Spanish utterances. The Spanish plan-realiser has been integrated with the existing English one using some of its solutions for similar structures in the two surface languages.

3.6.4 Generation Gap

The problem of the 'generation gap', relating to the interface between planner and realiser, has been overcome in this approach as responsibility has been shifted from the planner to the plan-realiser.

As seen in Section 3.2.3 this can be a serious problem in other architectures, as there are complex interactions between planner and realiser (interleaved systems), or the planner responsibility is overloaded in order to make sure that its decision can be realised (pipelined systems).

In the architecture of the LOLITA generator the planner does not make decisions on the linguistic level so that simplifies the interface between the modules. Furthermore the realiser can make decisions by itself in the case of conflicting instructions or the lack of detailed ones to ensure the production of a correct utterance.

3.6.5 Abstract Transformations

Another aspect of the solution to NLG adopted in LOLITA is the use of *Abstract Transformations.* These transformations act on the SemNet input before it is passed to the plan-realiser. Abstract transformations change the SemNet representation from *normal forms* to alternative forms which represent the same or a very similar meaning. They lead to changes in the utterance produced on surface language; Abstract transformations can produce variation which apart from being more natural, can satisfy stylistic constraints.

The normal forms used in the SemNet representation are not as restricted as other normalised forms (for example Schank's CDT). They have been chosen in such a way as to allow the generator to produce more than one utterance to express an event.

The transformations described below will also be available when generating Spanish.

Substitution of an Antonym Action

This abstract transformation can be performed when the action concept of an event is deemed by the semantics to have an antonym. The event can be negated and the action replaced by its antonym. Events which have actions rather than non-actions have been chosen as normalised forms.

Transformations on Copula Actions

Copula actions are those which take complements. If the complement has an antonym a transformation can be performed replacing the complement by its antonym and negating the action of the event.

That man is tall → That man is not small

Velvet feels smooth → Velvet does not feel rough

Transformations on Complemented Verb Pairs

Some actions which describe a transfer from an origin to a destination have a complement which can be used to describe the same event in the opposite direction. A transformation can be carried out by changing the action to its complement and swapping the roles of the origin and destination of the event.

John bought a car from the salesman → *The salesman sold a car to John.*

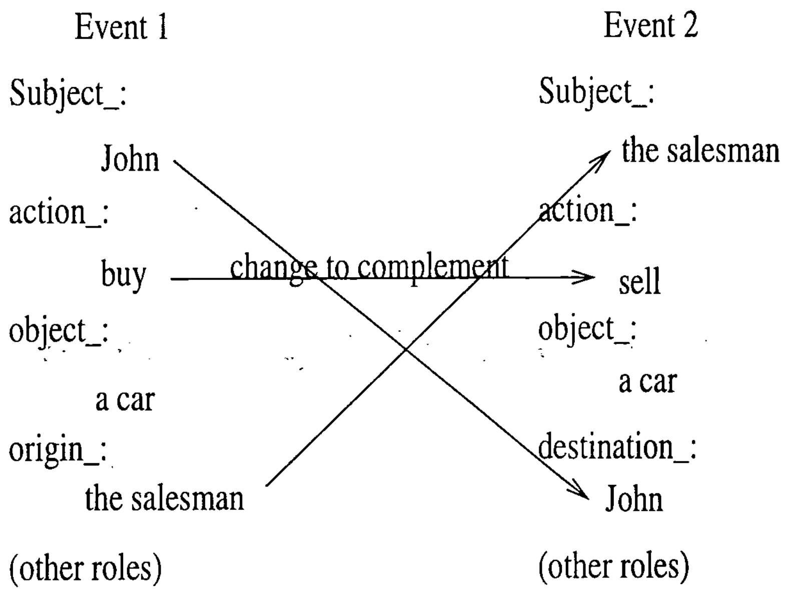


Figure 3.9: Example of a complemented action pair transformation

This transformation is only valid if the concepts representing the origin and the destination are of a compatible 'family' type (family is one of the controls attached to the nodes in SemNet, see Section 2.1.1). An example of an **incorrect transformation** is:

I bought some fruit from the shop → *The shop sold me some fruit*

Transformations on Multi-subject Events

Events with more than one subject can be transformed changing some of these subject links to co-subject links. The normalised form of the semantic network contains multiple subjects rather than co-subjects.

Steven and Paul went to play football →

Steven went to play football with Paul

De-lexical Transformations

De-lexical verbs are those that add very little meaning to a sentence (i.e. 'to have', 'to make', 'to give'); most of the meaning is given by the noun which is the object of the verb. The normal form in the semantic network doesn't contain actions which are realised using de-lexical verbs.

John kissed Sally → *John gave Sally a kiss*

You showered → *You had a shower*

Steven arranged to meet Paul in London → *Steven made an arrangement to meet Paul in London*

Figure 3.10: Examples of uses of de-lexical verbs

However, those de-lexical verbs have another meaning which is not de-lexical. For example *Martin has a nice car*, *The bomb made a big noise.*

Generalisation or Specialisation of concepts

- **Generalisation of concepts.** The plan-realiser already has the ability to find paraphrases therefore a transformation can be performed by removing the relevant language link from a node. As the plan-realiser will not be able to find a lexical entry for the concept represented in that node it must search for a more general concept which is LI and realise that concept together with other information which differentiates the meaning of the original and the more general concept.

  remove LI link to 'motorbike' → *'motor vehicle with two wheels'*

- **Specialisation of concepts** These transformations work in the opposite direction to the previous ones. An event can contain enough information to allow a move further down the event hierarchy to find a more specialised action. This information may then be dropped. The example involves an instrument role as the relevant information to allow the substitution of the action.

  I wounded you with a gun → *I shot you*

3.7 Chapter Summary

This chapter has presented an overview of the state of the art in Natural Language generation. It has concentrated on the areas of research that are the most relevant to that which is involved in the LOLITA generator. The chapter has also concentrated on systems which take semantic rich information similar to the SemNet representation used in the LOLITA system.

The latter part of the chapter has discussed the architecture of the LOLITA NL generator.

Chapter 4

Generating Spanish with LOLITA

This chapter presents the solutions adopted to generate Spanish within the LOLITA system. The goal of this research was not to build a new specific Spanish generator, but aimed instead at modifying the existing generator in order to enable it to generate both English and Spanish. Therefore, the Spanish generator is based upon the same theoretical principles as the current English generator. The same input, SemNet, and planner module (see Section 3.3.2) are considered for both generation processes. Spanish linguistic information and data have been added to SemNet to enable LOLITA to cope with Spanish. The plan-realiser module has been modified by developing new solutions for those grammatical features where English and Spanish languages differ. This chapter discusses the solutions adopted to produce correct Spanish utterances and describes implementation details to show how the features of Haskell have been utilised in the development of the Spanish Generator.

4.1 Adding information to the Semantic Network

The LOLITA semantic network, SemNet, is formed by a hierarchy of concepts linked by arcs. Concepts have a smaller 'grain-size' than words (lexical entries in any surface language) (see Section 2.1) and those concepts that can be described by a lexical entry are called 'language-isomorphic' (LI) concepts (see Section 3.4.3). LI concepts in Spanish are linked to the appropriate Spanish word by the link *Spanish_*.

Two new commands were added to the LOLITA's interface to input these lexical entries. These commands allow either the input of a new word linking it to a concept of SemNet or the linking of an existing word to more than one concept. The interface also allows the input of lexical information related to the new words in the form of controls (see Section 2.1). The next section describes the new controls added to the existing set because of the nature of the Spanish grammar.

4.1.1 New Syntactic Features

The Spanish grammar contains some features which are not found in the English grammar. New controls have been added to SemNet to cope with these features. They are as follows:

- **Gender.** All nouns in Spanish are either masculine or feminine in gender[1]and adjectives agree with nouns and pronouns in number and, if possible, gender.
- **Adjective position.** The position of adjectives in relation to nouns or pronouns is not fixed like in English. Some of them can only either precede or follow the noun.
- **Verb forms.** The Spanish verb system contains more tenses than the English system. A control indicates the tense of a particular verb form.

[1] there are one or two nouns of undecided gender.

- **Irregular Forms.** The irregular plural form of nouns, irregular forms of verbs and other irregular forms in the Spanish grammar are now present in the linguistic part of SemNet.

4.1.2 Irregular Forms

Exceptions to the normal transformations can be found in cases such as forming the plural of a noun, changing the gender of an adjective and conjugating verb forms. These irregular forms are present in the linguistic zone of SemNet linked to the root form of the corresponding word by the link *root_*. The infinitive form has been chosen as root form for verbs so the nodes containing the infinitive of a verb are linked to the concept representing its meaning. Masculine for the gender control and singular for the number control have also been chosen as root forms for those words that can change their gender or number. Figure 4.1 shows an example of a simplified portion of SemNet for the demonstrative adjective *aquel* (that) which has irregular feminine and plural forms.

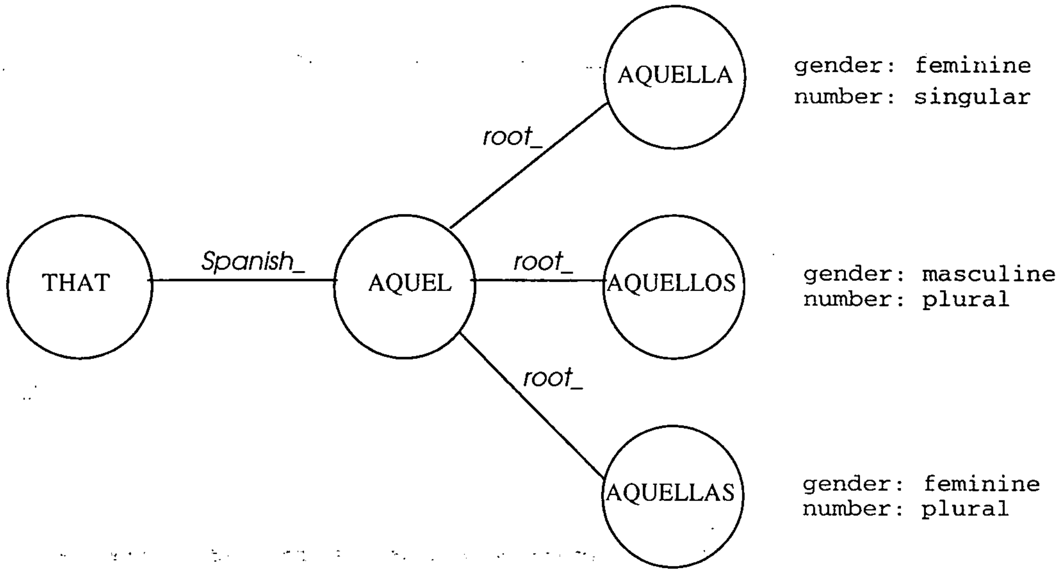


Figure 4.1: A simplified portion of SemNet.

Regular transformations are produced by the plan-realiser from the SemNet root forms (see Section 4.3.1 for nouns and adjectives and Section 4.3.3 for verbs) by applying morphological rules.

4.2 Integration with the Current Plan-realiser

The Spanish generator has been merged with the English generator in the plan-realiser module. Both generation processes receive the same input, the whole Sem-Net, so linguistic information about both languages is accessible for the generation module.

The same planner is also used by both generation processes. A completed planner has not yet been implemented but the operations provided to simulate it (see Section 3.4.2) are valid for Spanish generation as the planner makes decisions according to issues which are not surface language specific.

The work in this project has dealt mainly with the integration of the Spanish generation process in the current plan-realiser module. The Spanish generator has been built based upon the same theoretical principles as the English generator, the same approaches have been followed and a good deal of the same code has been utilised.

The plan-realiser must either relate concepts to lexical items corresponding to a particular surface language or search for LI concepts to describe adequately non-LI concepts. This search depends on three factors: *the SemNet content*, *realisation parameters* and *the grammar* (Section 3.4.3). The two first aspects are common to both generation processes as they correspond to the input and planner module and Spanish and English grammatical rules are merged in the plan-realiser module composing the third factor. New grammatical rules are applied to produce correct Spanish utterances when the existing rules are not valid for this purpose.

For example when generating the sentence *'Roberto wants a motorbike'* the same word order[2] can be applied in English and Spanish.

Roberto wants a motorbike (SVO)

Roberto quiere una moto (SVO)

[2] Although compared with English word order in Spanish is free, the Subject-Verb-Object (SVO) order is usually considered as the 'normal' order.

Instead, if the entity *motorbike* has already been mentioned and can be pronominalised then different rules are applied for each language as the pronoun is shifted before the verb in Spanish.

Roberto wants it (SVO), 'it' ⇒ 'motorbike'

Roberto la quiere (SOV), 'la' ⇒ 'moto'

The next section provides details about the solutions adopted to produce Spanish besides relevant examples generated by the LOLITA generator.

4.3 Generation in Spanish

4.3.1 Generation of Entities

Theoretically every concept in SemNet is defined by the whole of the network (Section 2.1.1). However, it is impractical and unnecessary to realise the whole network each time an entity is to be expressed. What is required is to generate an expression which defines a particular entity in sufficient detail and the uniqueness of the entity generated must be assured. The default in the LOLITA generator is to produce the most specific realisation of the concept (i.e. if the concept is LI then the lexical entry linked to it will be generated).

The problem of deciding when to generate an alternative paraphrase to express a concept has not yet been tackled in the LOLITA generator. The decision will be further distributed between the planner and plan-realiser [Smith, 1995].

If an entity concept node is LI (it has a link to a lexical entry) then the plan-realiser can generate this lexical item with the correct quantification (given by the **rank** control). For example if a concept represents a set of entities then the root of the lexical item has to be pluralised. Morphological rules produce regular pluralisations. For words with irregular plurals, these are present in the linguistic part of SemNet.

If a concept is not LI then the plan-realiser has to search for an alternative expression. If the node has more than one universal concept then the plan-realiser has to move across the universal links (this process may be recursive if any of the universals is not LI) to find the lexical items for each of the universal concepts of the original concept. Heuristics can be used to order these concepts. In particular, if any of the universal is linked to an adjective, its position in relation with the noun of the entity (which will be represented by another universal concept) has to be controlled as some adjectives are restricted in their position. The next examples show how the adjectives can be positioned before or after the noun.

un gran coche (a great car)
suelos limpios y brillantes (clean and bright floors)
'el poderoso ejército romano' (the powerful Roman army)

Another feature of the Spanish grammar is that the adjectives must agree in gender and number with the noun involved in the same entity.

un **buen** *cocinero* (a good cook)
una **buena** *cerradura* (a good locker)
buenos *amigos* (good friends)
buenas *comidas* (good meals)

Determiners

	Definite articles		**Indefinite articles**	
	Masculine	**Feminine**	**Masculine**	**Feminine**
Singular	*el*	*la*	*un*	*una*
Plural	*los*	*las*	*unos*	*unas*

Figure 4.2: Articles in Spanish.

A determiner may also be required to correctly realise an entity. The plan-realiser must ensure that a correct determiner is used (although in some cases more than one determiner could be correctly used).

There seems to be little discussion in the NLG literature about rules for the generation of articles. The uses of **the definite article** treated in the current LOLITA's English generator ([Smith, 1995], [Garigliano, 1992]) are valid for the generation of Spanish utterances:

- The definite article is used to refer to an unique element in the external world or at least to an unique element in the common knowledge of the writer and reader. For example, *'la luna'* (The Moon), *'el gobierno'* (The Government). It will be up to the planner to mark concepts as being uniquely defined by the context.
- The definite article is used to show the uniqueness of a concept when it is defined in the sentence. For example, *'la moto que yo guardo en mi garage'* (the motorbike that I keep in my garage).
- The definite article is used to refer to something that has been introduced before and is unique in the focus of discourse. For example, *'Yo encontré un perro, el perro me mordió'* (I met a dog, the dog bit me).
- The definite article is used to refer to something that is implicitly unique in the focus of discourse. For example, *'Yo fui a un restaurante. La camarera era bonita'* (I went to a restaurant. The waitress was pretty).
- The definite article is used as a determiner for universal sets. For example, *'El caballo es un animal precioso'* (The horse is a beautiful animal).
- The definite article can be used in a situation where there is no 'script' but it is used to trigger one. For example, *'Estuve buscando a Russell. La oficina estaba vacía'* (I was looking for Russell. The office was empty). This complex use of the definite article has not been considered in the LOLITA's generation proccess.

Additionally, the Spanish generator uses the definite article in other different grammatical cases. Some examples are as follow:

- The definite article is required before generic nouns, i.e. nouns than refer to something in general. These are typically

  - Abstract nouns: *la democracia* (democracy).
  - Substances in general *La sangre no tiene precio* (Blood has no price)
  - Countable nouns which refer to all the members of their class:

    'Tom ama los coches' (Tom loves cars)
    'Amo las flores' (I love flowers)

  The article is omitted before nouns that refer not to the whole but only to part of something. **Yo quiero cerveza** (I want beer), **Ellos nos dieron lapiceros** (they gave us some pencils).

- If two or more nouns appear together, each has its own article if they are individually particularised:

  'el padre y la madre' (the father and mother)
  'el agua y la leche' (the water and milk).

There are some special transformations when using articles:

- *El* and *un* are always used immediately before singular feminine nouns beginning with stressed *a-* or *ha-*, despite the fact that all the adjectives and pronouns that modify these nouns must be in the feminine form. The feminine forms of the article are used with the plural forms of these words.

  el agua (water)
  el/un alma humana (the human soul)
  el/un águila (eagle)

- *De* plus *el* is shortened to *del* (of the). *A* plus *el* is shortened to *al* (to the). This rule is not used when the article is part of a proper name.

  'Puedo ver al elefante' (I can see the elephant)
  'No cojas cosas del suelo' (Do not take things from the floor)

Proper Nouns

Entities with a **rank** control of value **Named Individual** are realised as proper nouns. Currently the plan-realiser assumes that a named individual is adequately specified generating its name. In the future the planner will decide whether the name is sufficient or more information is required.

Entities with Relative Clauses

An entity may be involved in events which may define that entity more fully. Relative clauses or 'special' relative clauses may be used to describe those events. The decision of when a relative clause is needed to define a concept will be a job for the planner [Smith, 1995]. In the absence of instructions from the planner, the plan-realiser generates relative clauses for entities depending on the information of SemNet, the grammar and the value of the rhythm realisation parameter. The rhythm parameter limits the number of nested clauses to zero, one or two (a greater number leads to complex and incomprehensible utterances) and a separate sentence can be used if an event has to be mentioned and can not be produced as a relative clause. Details about the generation of relative clause events are provided in the next section.

Some events can not be expressed using normal relative clauses and need special rules for their realisation. Some examples are as follow:

- **Possessive clauses.** If an entity is the object of an event with the action *'poseer'* (to own) or the internal action **poss_relate** then the plan-realiser can link the object and subject (the owner) with *de* to express the event or can generate a possessive adjective if the owner has been mentioned. The event can also be realised as a normal event:

  event: 'Paul posee un libro' (Paul owns a book)

  'El libro de Paul' (Paul's book)

  'El libro que Paul posee' (The book that Paul owns)

event: 'Posees dos relojes' (You own two watches)

'Quiero tus relojes' (I want your watches)

'Quiero los dos relojes que posees' (I want the two watches that you own)

A possessive article agrees with the owner in the person and with the object in the number.

- **Noun co-locations.** Some internal events can be realised using a collocation of nouns. For example:

  - action **is_part_of**: If an entity is the subject of an event with such a action the object can be used as a collocation before the entity: For example *'coche bomba'* (car bomb).
  - action **controls_**: For example *'mecánico de coches'* (car mechanic), *conductor de trenes* (train driver).
  - action **is_a**: if the object of the event is an attribute (marked by the control **type**) then the object can be expressed as an adjective. For example *'Yo cogí los tomates verdes'* (I took the green tomatoes) instead of *'Yo cogí los tomates que estaban verdes'* (I took the tomatoes that were green)

4.3.2 Generation of Events

The relationships between entities are expressed as events in SemNet. These events can be generated by realising the entities involved in them together, assigning each entity a different role (i.e. subject, verb, object).

Word order can be considered as free in Spanish. The factors that call for a particular word order depend on considerations such as style, context and rhythm.

The planner may provide instructions (with these considerations) which limit the number or order of clauses (a role in the event is realised by a clause) to be expressed. In the absence of such instructions, and depending on the length and rhythm realisation parameters, the plan-realiser will normally generate all clauses (although it is unlikely that every clause will be present in a particular event) in the following order (It should be noted that this is the same order already followed by the current generator which has been found valid for the generation of Spanish utterances): certainty, time, subject, verb, object, co-subject, origin, destination, instrument, location and goal. However, different word orders will be considered in questions, passive sentences, some relative clause events, events with certain actions ('gustar' (to like), 'disgustar' (to dislike), 'importar' (to care), etc) and when using pronouns to express some of the clauses.

Generating Event Roles

Since the same clauses are considered by the Spanish and English generator and both processes receive the same input, the existing algorithm has also been applied in the process of generating Spanish. This algorithm, that realises the roles associated with an event, is an example of procedural control within the plan-realiser. The plan-realiser will only attempt to realise clauses corresponding to roles if these roles are actually present in the input event. There may be cases, however, when a clause for a role that is not explicit in a particular event is required. In this case the role has to be inherited from an event higher in the hierarchy.

- Subject clause (**subject_**). The subject can be omitted in some cases where pronominalisation is allowed; in particular, the pronouns corresponding to the first and second persons (yo/tú) are always omitted (except for the purpose of emphasis). More details about pronouns are provided in Section 4.3.4.

- Object clause (**object_**). If a pronoun is used to realise the object clause the word order is altered as object pronouns are shifted before the verb or attached to the verb if the infinitive or gerund forms are required, '**Me** *amas'* (you love me), *'Estabas gritándo*me*'* (you were shouting at me), (more details about pronouns are provided in Section 4.3.4). In some cases where the object is an event a pronoun representing the subject of this event can also be shifted before the verb of the main event (i.e. '**Te** *forcé a golpear el balón'* (I forced you to hit the ball).

The preposition 'a' is used before objects denoting human beings or animals. For example,

'Llevó a las niñas al zoo' (He took the girls to the zoo)

'Odias a tu jefe' (You hate your boss)

*Regué las flores** (I watered the flowers) ('a' not required)

- Time clause (**time_**). If the time clause is not an event then simple heuristics are used to produce the correct linking phrases for explicit times; for example '**El** *martes ..'*, **A las** *9pm'*, '**En** *1995*' etc) [3]. If an event appears in a time slot, then depending on whether the event is to be 'opened' or 'closed' (see events in events), the phrases 'cuando' or 'al tiempo de' will be used (e.g.,*'Cuando la bomba explotó, el taxi fue destruido'* or *'Al tiempo de la explosión de la bomba, el taxi fue destruido'* ('When the bomb exploded, the taxi was destroyed' or 'At the time of the bomb explosion, the taxi was destroyed')). The plan-realiser is recursively called to generate the clause.

- Certainty clause (**certainty_**). The certainty link can be added to events by the LOLITA's analysis process (e.g., using inference methods such as analogy [Long and Garigliano, 1994], or source control [Bokma and Garigliano, 1992]) and is a measure of LOLITA's acceptability of the truth of an event. The plan-realiser uses phrases (dependent on their value of certainty and the planning instructions) such as *'hay una ligera posibilidad de que', 'es probable que..'* etc.

- Co-subject clause (**co_subject_**). Co-subject clauses are realised using the preposition 'con' and a recursive call to the plan-realiser.

  'Fui con Mike a Londres' (I went with Mike to London).

- Origin clause (**origin_**). Origin clauses are realised using 'de' (or 'del' if the article 'el' is to be generated after the preposition. See Section 4.3.1) and a recursive call to the plan-realiser. For example:

[3] When the time representation of LOLITA is improved these heuristics will be more precise.

'Recibí un regalo de Carol' (I received a present from Carol)

'Ayer vine de Madrid' (Yesterday I came from Madrid)

If the origin represents a human being and a pronoun can be used to stand for it (i.e. the origin has already been mentioned) then depending on the rest of roles to be generated a 'prepositional pronoun' or an 'object pronoun' is realised (Section 4.3.4). For example, *Ella* me *compró un libro* (she bought a book from me).

- Destination clause (**destination_**). Destination clauses are realised using 'a' or ('al' if the article 'el' is to be generated after the preposition. See Section 4.3.1) and a recursive call to the plan-realiser. For example:

  'Compré una flor a Mary' (I bought a flower to Mary)

  'Conté un cuento a los niños' (I told a tale to the children)

  If the destination represents a human being and a pronoun can be used to stand for it (i.e. the destination has already been mentioned) then depending on the rest of roles to be generated a 'prepositional pronoun' or an 'object pronoun' are realised (Section 4.3.4). For example, 'Te *dió una moneda'* (she/he gave you a coin). If the destination represents a location and pronominalisation is allowed then 'allí' can be used.

- Instrument clause (**instrument_**). If the instrument clause is not an event then it can be realised using 'con' and a recursive call to the realiser. For example *'Paul mató a Steven con una daga'* (Paul killed Steven with a dagger). If the instrument is an event then it can be realised using the gerund form in the action, *'la gente puede reservar habitaciones* **llamando al hotel**' (people can book rooms by calling the hotel).

- Location clause (**location_**). Location clauses are realised by generating the correct preposition before the location. More details about the location clause are further provided in this chapter (see Positions).

- Goal clause (**goal_**). If the goal clause is not an event then it can be realised with 'por' and a recursive call to the plan-realiser. For example

```
    * event: 28949 *
 universal_:
   event - 7688 - rank: universal  (happen_)  - definition_
 cause_:
   event - 28946 - rank: universal  (is_a)
   event - 28944 - rank: universal  (is_a)
 subject_:
   roberto - 19845 - rank: named individual
 action_:
   give - 3936 -
 object_:
   tip - 28948 - rank: individual
 destination_:
   driver - 28945 - rank: individual
 time_:
   past_ - 20991 -
 date:
   12 June 1993
 source_:
   roberto - 19845 - rank: named individual
*******************************************************
```

Diste una gran propina al conductor porque el taxi que llamaste era acogedor. Estabas cansado asi que te fuiste a tu casa.

Figure 4.3: Example of an event in the LOLITA representation

'Robé el banco por dinero' (I robbed the bank for money)

'Mataría por ella' (I would kill for her/it)

If the goal is an event and the subjects in both events are different then 'para que' and a recursive call to the plan-realiser can be used. For example:

'John venderá la bicicleta para que Steven pueda comprar una moto' (John will sell the bicycle so that Steven can buy a motorbike)

If the subjects are the same 'para' can be used and the event generated with a forced infinitive. For example:

'Escribí la carta para felicitarla' (I wrote the letter to congratulate her)

Passive

Sentences in passive voice can be generated when the action of the event is transitive (and it is not a sentential action with an 'open' event required), there is an explicit object and the event is not marked as a command.

There is more than one construction for the passive:

- Using the appropriate tense and person of 'ser' ('to be') and the past participle of the original verb, which agrees in number and gender with the subject of *'ser'* (the object of the original event). The subject can be generated by using the preposition 'por' (by) for each subject involved in the event.

  'Sally es amada por John y por David' (Sally is loved by John and David).

- When there is no an explicit subject and the verb is in third person the passive can be produced with the 'passive se'

  'Los cangrejos se cuecen en vino blanco' (The crabs are cooked in white wine).

Questions

Questions without a question pronoun are represented as normal events with the **status_** of *question.* They are produced by generating the verb before the subject.

¿Vino el chico a mi casa? (Does the boy come to my house?)

Questions with a question pronoun are represented using normal events with the role to which the question relates marked. A question pronoun is realised ('qué, quiénes, quién, cuál, cuáles, dónde, cuándo, por qué') followed by the question event omitting the role denoted by the pronoun. For example:

'¿Dónde está el gato?' (Where is the cat?) for location
'¿Por qué no comes el pollo?' (Why do you not eat the chicken?) for cause
'¿Quiénes fueron a la fiesta?' (Who went to the party?) for animate subject.

Relative Clause Events

As described in Section 4.3.1 an entity can be defined in more detail by describing events in which the entity is involved. Relative clauses are attached to the entity to realise these events. The plan realiser must first generate a relative pronoun according to the role the entity plays in the event to be expressed:

- The entity is the subject of the event: the pronoun **que** is generated.

  'El chico que rompió el cristal' (The boy who broke the glass)
  'El coche que ganó la carrera' (The car which won the race)

- The entity is the object of the event: the pronoun **que** or **a quien/quienes** is generated when the entity is animated (marked by a control) and the pronoun **que** when the entity is inanimate. The verb is generated before the subject to keep the verb close to the relative pronoun:

  'La mesa que usaste' (the table that you used)
  'El hombre que ama Sally' (the man whom Sally loves)
  'El chico a quien diste una camisa' (the boy whom you gave a shirt)

  If the entity is the object of an 'is_a' event then the plan-realiser produces the following: 'de quienes' or 'de los/las cuales' (animate or not) followed by the subject, followed by the correct present form of 'ser' (is/are), followed by 'uno/una' (singular) or 'miembros' (plural):

  'Hombres locos de quienes Rasputin es uno' (Mad men of whom Rasputin is one)

- If the entity plays other roles the preposition corresponding to those roles (see generation of event roles) followed by 'quien/quienes' (depending on the number of the entity) when the entity is marked animate or followed by 'el/la/los/las' 'cual/cuales' (depending on the gender and number of the entity) are realised.

  'La chica a quien compré un caramelo' (the girl for whom I bought a sweet)
  'El cuchillo con el cual cortó la cebolla' (the knife with which he/she chopped

the onion)

After the relative pronoun the plan-realiser generates the relative clause event similarly to a normal event (although as described above in some cases the order can be altered). The entity being defined by the relative clause does not have to be expressed again.

Events within Events

Events can appear playing any of the roles of other events. Sometimes those embedded events have to be expressed with a noun phrase ('close' events). Some heuristics are adopted to decide when to generate an 'open' or a 'close' event.

- Close events: Closed events may be realised depending on:
  - The context of the utterance: when the event to be realised is part of a dialogue, an answer or the slot of a template are situations where expressing a close event may be more natural.
  - The verb of the main event: verbs like 'describir' (to describe) require the event objects to closed: *'Describí la explosión de la bomba'* (I described the bomb explosion)
  - Events inheriting most of its roles from the prototypical event: *'la explosión'* instead of *'dispositivos explosivos explotaron'* (the explosion, explosive devices exploded).
- Open events:
  - The verb of the main event: verbs like 'pensar' (to think), 'sugerir' (to suggest) require the event objects to be open: *'Pienso que el hombre murió'* (I think that the man died)
  - Infinitive verbs as 'forzar' (to force), 'querer' (to want) which require the object event to be open and the verb of the embedded event to be in the infinitive or any subjunctive form: *'Te forcé a comprar el periódico'*

(I forced you to buy the newspaper), *'Quiero que vengas'* (I want you to come).

- Events not inheriting most of its roles from the prototypical event: *'En 1963, Oswald asesinó a Kennedy en Dallas'* is better than *'El asesinato de kennedy por Oswald en Dallas'* (In 1963, Oswald murdered Kennedy in Dallas, The 1963 murder of Kennedy by Oswald in Dallas)

Some heuristics are adopted to 'open' a close event and to 'close' an open event when this is required because of the nature of any of the events:

- A close embedded event can be 'opened' by using the correct form of the verb 'suceder' (to happen)

  'Un hombre sugirió que una explosión sucedió' (A man suggested that an explosion happened)

- An open embedded event can be 'closed' by using the gerund form of its verb

  'Ví la bomba explotando' (I watched the bomb exploding)

Causal events

Events being the cause of other events are linked in SemNet by the arc **cause_**. These events are realised as follow:

- **Event1 has a *cause_* link to Event2**: if event2 is **hypothetical** then the structure 'event1 si event2' is used to realise the events with the verbs of the actions in the following forms:

  - If the action of event1 is in the future tense then the action of event1 is realised in the future form and the action of event2 in the present form: *'Me quedaré si viene'* (I will stay if he comes).

- If the tenses are in the present tense then the action of event1 is realised in the conditional form and the action of event2 in the imperfect subjunctive form: *'Trabajaría si no estuviera cansado'* (I would work if I was not tired).
- If the tenses are in the past tense then the action of event1 is realised in the perfect conditional form and the action of event2 in the pluperfect subjunctive form: *'Habrías tenido dinero si hubieras vendido tu casa'* (you would have had money if you had sold your house).

If event2 is not hypothetical then the events are realised using the structure 'event1 porque event2':

'Eres feliz porque te amo' (you are happy because I love you)

- **Event2 has a link** *cause_of* **to Event1**: if event2 is hypothetical then the structure 'si event2, event1' is used to realise the events with the verbs of the actions following the same conditions than the previous case:

  'Si duermes, te sentirás mejor' (if you sleep then you will feel better)
  'Si poseyeras una moto, irías a París' (if you had a motorbike then you would go to Paris)
  'Si hubieras visto el ratón, habrías gritado' (if you had seen the mouse then you would have shouted)

  If event2 is not hypothetical then the events are realised using the structure 'event2 así que event1':

  'El vendrá así que lo verás' (he will come so you will see him)

Other Aspects of Events

Internal events

Some events represented in SemNet are not directly expressible in any language. They usually represent input text which has not been fully disambiguated. Some of these internal events can be realised as follow:

- action **is_a**: internal events with this action can be expressed using the verb 'ser' if the object is an entity or an adjective referring to identity or nature, and using the verb 'estar' for the rest of attributes. Subject and object must agree in gender and number.

  'Paul es cocinero' (Paul is a cook)

  'Louise es alta' (Louise is tall)

  'Las carreteras estaban obstaculizadas' (The roads were obstructed)

- action **poss_relate**: Internal events with this action can be realised using the verb 'tener' (to have). This verb will convey the ambiguity contained in the event.

Positions

SemNet builds explicit position nodes [Short and Garigliano, 1993] which can be realised in isolation, as a relative clause or as a location role in a event. The plan-realiser generates an appropriate preposition relating the entity associated with the position. Examples of preposition used are 'entre, en, sobre, alrededor de, próximo a, cerca de, encima, debajo, de, por, detrá, enfrente, dentro', etc. The plan-realiser can also realise more complex positions including distances ('kilómetros, metros, millas', etc)

'Entre tu casa y el hotel' (between your house and the hotel)

'El plato sobre la mesa que Mike cogió' (The plate on the table that Mike took)

'El gato se sienta en un felpudo' (The cat sat on a mat)

'Roberto pone una piedra cada 100 metros' (Roberto places a stone every 100 metres)

Negation with 'no'

No usually precedes the verb that it negates, but when object pronouns (Section 4.3.4) are realised, these pronouns are generated between 'no' and the verb as these pronouns are never separated from the verb.

'No intentó verla' (He did not try to see her)

'No te lo dije' (I did not tell it to you)

If a verb is generated in the infinitive form and requires the preposition 'a' before it, 'no' will be realised between the preposition and the verb:

'Te forzaré a no beber cerveza' (I will force you to not drink beer)

Events with verbs such as 'gustar' (to like), 'disgustar' (to dislike), 'importar' (to care)

Events containing one of these actions are realised differently from the normal events. The object is realised as if it was the subject and the subject is realised as if it was the object using an object pronoun (Section 4.3.4).

'La miel les gusta a los osos' (Bears like honey)

'Me gustas' (I like you)

If the object (realised as the subject) is an event or the subject is being realised (i.e. the event with the special verb is a relative clause) then the order of realisation will be changed as it will be realised after the verb.

'No les importa que no tengan dinero' (They don‘t care that they don‘t have money)

4.3.3 Generation of Actions

The action in an event is realised as a verb and it usually conveys most of the meaning of the event. If an action is non LI (it is not connected to a lexical entry in the surface language), the plan-realiser will have to find the prototypical event for that action and then find the first prototypical event above the event that has a LI action. The plan-realiser can generate this LI action but it will also have to generate the roles in the original prototypical event which differ from those in the event with the LI action in order to express the meaning of the original action.

Time and tense representation in the LOLITA system is currently under development [Short, forthcoming 1995]. However, by using the explicit tense of the

time_ slot, or by using the tense determined by looking at the relative times of events (i.e. causal events may require to realise a different tense than the explicit one), an adequate tense can be realised.

Once the correct tense for the verb expressing the action has been found, morphological rules are applied to ensure that this verb agrees with the subject in person (first person, second person, third person) and in number (singular/plural). The plan-realiser then checks whether the verb is irregular for the chosen tense and person. Rules for realising regular verbs are found within the plan-realiser while irregular verb forms are present in the linguistic part of SemNet.

The following tenses can be realised:

present indicative: *bebo cerveza* (I drink beer)
imperfect indicative: *bebía cerveza* (I drank beer)
preterite: *bebí cerveza* (I drank beer)
future: *beberé cerveza* (I will drink beer)
conditional: *bebería cerveza si* ... (I would drink beer if ...)
present subjunctive: *quieres que* **beba** *cerveza* (you want me to drink beer)
imperfect subjunctive: *si bebiera cerveza* ... (if I drank beer ...)
perfect indicative: *he bebido cerveza* (I have drunk beer)
pluperfect indicative: *había bebido cerveza* (I had drunk beer)
perfect conditional: *habría bebido cerveza* (I would have drunk beer)
perfect future: *habré bebido cerveza* (I will have drunk beer)
perfect subjunctive: *que haya bebido cerveza*
pluperfect subjunctive: *hubiera bebido cerveza*
gerund: *estoy* **bebiendo** *cerveza* (I am drinking beer)
infinitive: *beber* (to drink)
past participle: *he* **bebido** *cerveza* (I have drunk beer)

The rules applied to regular verbs are different depending on the conjugation to which the verbs belong. The conjugation is distinguished by the final vowel of the infinitive: (1)-*ar*, (2)-*er*, (3) *-ir (or -ír)*.

The realisation of some tenses can lead to a change in the word order of the utterance being generated:

- reflexive or pronominal verbs: An object pronoun is generated before the verb.

  'Me estoy lavando' (I am washing (myself))
  'Los prisioneros se escaparon' (The prisoners escaped)
  'No te conoces' (You do not know yourself)

- infinitive or gerund forms of the verbs may need the object pronouns to be realised attached to them

  *'Cogì la piedra para guardar*la*'* (I took the stone in order to keep it)
  *'Estoy estudiándo*lo *'* (I am studying it)

4.3.4 Generation of personal pronouns

When realising an entity which has already been mentioned either explicitly or implicitly (it is in the context), the plan-realiser applies different rules to adequately describe the entity. Sometimes a simple pronoun will be adequate and in other cases a shorter noun phrase will be required. Complex handling of anaphora using context, scripts, etc. will be the responsibility of the planner. In the absence of the planner instructions the plan-realiser produces anaphora by keeping a record of all the entities and events that have already been referred to. If an entity or event is to be referred and it is the only one of its pronoun class (there are not more entities with the same person, number and gender) from the referred record of entities then it can be pronominalised. In some cases even the pronoun can be omitted (see Subject pronouns).

Personal pronouns can be split in three different groups depending on the role they play in the event and the position in which they are produced: *subject pronouns*, object pronouns and prepositional pronouns.

PERSON	EMPHATIC	OBJECT	PREPOSITIONAL	
SINGULAR	SUBJECT			
1	yo	me	mí	I
2	tú	te	ti	you
3	él	lo/la/le	el	he, it
	ella	la/le	ella	she, it
	ello	lo/le	ello	it (neuter)
	se		sí	'reflexive'
PLURAL				
1	nosotros	nos	nosotros	we(masc)
	nosotras	nos	nosotras	we(fem)
2	vosotros	os	vosotros	you(masc)
	vosotras	os	vosotras	you(fem)
3	ellos	los/les	ellos	they(masc)
	ellas	las/les	ellas	they(fem)
	se		sí	'reflexive'

Figure 4.4: Table of the Spanish personal pronouns.

Use of subject pronouns

Subject pronouns are those that stand for an entity that is the subject of an event. The default in the plan-realiser is not to generate the pronoun (nor the entity) as the person of the subject is expressed by the verb ending. However, they may be optionally generated to emphasise the subject of a verb but only if they stand for an entity representing a human being. It will be up to the planner to decide whether a subject pronoun is to be expressed.

*'Ir*é *a casa con un amigo'* (I will go home with a friend)

*'Ir*ás *a casa con un amigo'* (You will go home with a friend)

*'John y Maria no vinieron porque estab*an *cansados'* (John and Maria did not come because they were tired)

There are some exceptions to the use of the subject pronouns of Figure 4.4. For example, if the event plays a role in another event and the verb is realised in the infinitive form an object pronoun is used to stand for the subject of the embedded event. This pronoun is realised before the action of the main event.

'Sally te *forzó a estudiar la grámatica'* (Sally forced you to study the grammar)

Use of object pronouns

Object pronouns are used to express some of the roles of an event such as the object, the destination and the origin.

First and second person object pronouns

Forms of first and second person object pronouns are invariably independent of which is their role in the event (see Figure 4.4):

'Me han visto' (They have seen me)

'Te quitaron a tus hijos' (They take your children from you)

'Nos dejó una manta' (He/she left a blanket to us)

Third person object pronouns

Different forms of the third person object pronouns are used depending on the role in the event:

- Object role. LOLITA produces the following pronouns:
  - **se** is used when the subject and object are represented by the same entity or entities.

    'Se lava' (he is washing (himself))

    'Se cortaron con una lata' (they cut themselves with a tin)
  - **la** is used when the pronoun stands for a feminine entity (except with some verbs, see use of 'le' below).

    'Maria estaba en la oficina. Yo la vi' (Maria was in the office. I saw her)

    'La mesa no es nueva, la he pintado' (The table is not new, I have painted it)
  - **le** is used when the pronoun stands for a singular, human and male entity[4].

[4] lo is acceptable to most of the native speakers of Spanish in this case

'le vi' (I saw him)

'le golpeé con un palo' (I hit him with a stick)

'le' (and 'les') is also used with verbs such as 'gustar' (to like), 'disgustar' (to dislike) and 'importar' (to care)

'les gusta el vino' (They like wine)

'No les importa ir con Rick' (They do not care to go with Rick)

- **lo** stands for the singular entities where 'la' and 'le' are not used.

  'Era un árbol extraõ. La gente lo miraba' (It was a strange tree. People looked at it)

  'lo compré en Londres' (I bought it in London)

- **las** and **los** stand for plural feminine and plural masculine entities respectively.

  'Los oí desde el río' (I heard them from the river)

  'las bombas son peligrosas. Las odio' (The bombs are dangerous. I hate them)

- Other roles different than the object. **le** and **les** respectively are produced standing for singular and plural entities.

  'Le compré una muñeca' (I bought him/her a doll)

  'Les quereis por dinero' (You want them for money)

Oder of object pronouns

Some events are generated using more than one object pronoun. The invariable order of object pronouns when two or more are generated is:

```
se          te/os          me/nos          le/lo/la/les/los/las
```

'Maria te lo dijo' (Maria told it to you)

'Me la robó' (He stole it from me)

Substitution of le/les by se

If **le** or **les** have to be generated but they are immediately followed by an object pronoun beginning with 'l' (i.e 'lo/la/los/las), then the pronoun **se** is produced instead of 'le' or 'les'.

'Se lo doy' (I give it to him/her)

'Se lo dije' (I told it to them)

Position of object pronouns

The word order produced by the plan-realiser when object pronouns are generated is different from the default order.

- **Pronouns with verbs in finite tenses**

  If the action of the event is generated with a verb in a form other than infinitive, gerund or past participle then the pronoun is generated immediately before the verb. In compound tenses the pronouns are generated before the auxiliary verb.

  'Te los enviaré luego' (I will send them to you later)

  'Os las guardo' (I keep them for you)

  'La has visto' (You have seen her)

- **Positions with infinitives.** The object pronouns are generated suffixed to the verb.

  *'Vine para decir***selo***'* (I came to tell it to him)

  *'Me hizo abrir***la***'* (He made me open it)

- **Position with gerunds.** The object pronouns are generated suffixed to the verb.

  *'La vi golpeándo***la***'* (I saw her hitting it)

  'Estuvieron esperándonos' (They were waiting for us)

Personal pronouns after prepositions

The forms of the personal pronouns produced after prepositions are the same as the forms of subject pronouns except 'yo' and 'tú' which have different prepositional forms 'mí' and 'ti'.

'Compraron la camisa para ti' (They bought the shirt for you)

'Fue con ellas' (He went with them)

There are two special forms produced **conmigo** and **consigo** which correspond to 'con + mí' and 'con + ti'.

'Paul vino conmigo' (Paul came with me)

'Lo vió contigo' (She saw it with you)

Redundant object pronouns are produced when generating the first and second person plural forms to clarify which entities compound the plural.

'Nos lo dió a mí y a Mark' (He gave it to me and Mark)

'Os amo a ti y a tu hermana' (I love you and your sister)

4.4 Implementation overview

The use of the functional language Haskell in the development of LOLITA eases program comprehension and makes the integration of new code into the system easier [Hazan *et al.*, 1993]. This proved right for the integration of the Spanish generator with the current LOLITA generator. A great deal of the existing code has been used during the implementation of the Spanish generator and, for example, the use of ADTs (Section 2.3.1) simplified this integration by allowing the modification of the data types without affecting the parts of the English generator using them.

4.4.1 Data Types

This section briefly introduces the important types used in the implementation of the Spanish generator.

- **Global**: The **Global** data type is perhaps the most important used in the LOLITA system. Functional programming languages do not allow *side effects* (see section 2.3.1) so the 'state' of the system has to be made explicit and passed round between functions rather than leaving it implicit (as in imperative languages). The **Global** data type corresponds to this overall system state. It holds, among other things, the whole of LOLITA's SemNet representation together with information on how the SemNet has been most recently changed and the planning instructions that the plan-realiser receives form the planner or an underlying application.

- **Noderef**: A **Noderef** is simply a reference to a particular unique node within the SemNet representation.

- **Meaning**: The **Meaning** data type is simply a 'repackaging' of information held in the Global and Noderef data-types to make it more suitable for generation. A **Meaning** holds the meaning of a particular node in the SemNet by combining a starting point node and the complete SemNet representation.

- **Generator**: The **Generator** is a data type which acts as a 'building block' during the generation process. As an utterance is built, generators representing different parts of the utterance are composed together to form a more complete generator. This generator is then applied to the input instructions from the planner to produce a NL utterance. The generator comprises the utterance generated so far as well as planning instructions and switches set by the planner.

- **GenVals**: The **GenVals** data type is a collection of flags set by the generator as it goes along which can affect the future choices that the generator can make. Examples are a flag to force embedded events to be open or closed and a flag to force a verb to be in a particular tense.

GenVals has been modified (i.e. more tenses were required) for use by the Spanish generator but the functions of the existing generator using this data type have not been altered proving the suitability of the use of ADTs.

- The **SpVals** data type is a collection of flags, as is the GenVals data type, used only by the Spanish generator. Examples are a flag to force an adjective to be produced in its feminine form and a flag indicating the person of a verb.

Generation Process

Figure 4.5 shows an oversimplified portion of the code. This code is included to give an idea of how the generation process is implemented in addition to showing how some of the Haskell features have been used.

Each function is given next to its type declaration. For example, the function 'say_event' (a simplified version of the function is shown) takes as input parameters values of the type **GenVals** and **Meaning** and returns a result of type **Generator**. Assuming that a node containing an event is to be generated and the function **say_event** is called the process is as follow: The function decides if the event is to be expressed in the passive or active voice by using the query function **if_gen** (it is a function and not a Haskell constructor). This function queries the hidden parameter of Generator type (which contains the utterance so far and the planning instructions) by using the function **is_style**. If for example the active voice is required the function **say_active_event** is called. At this point the generation process differs for English and Spanish so the function again queries the hidden parameter of Generator type to determine which language is active. For Spanish, the function **say_active_event_sp** is called. This function calls other functions to generate each of the roles in the event. Each of these functions return a **Generator** and the **before_** function is used to compose the generators together. Some of the functions called to express the roles of the event can again be utilised for both languages (i.e. say_subject).

The use of ADTs has been described in the previous section and following this

```
say_event:: GenVals -> Meaning -> Generator
say_event gv e
  = if_gen (is_style Active)
       say_active_event gv e
    ‘or_else‘
    if_gen (is_style Passive)
       say_passive_event gv e
         .
         .
         .

say_active_event:: GenVals -> Meaning -> Generator
say_active_event gv e
  = if_gen (is_language Spanish)
        (say_active_event_sp gv e)
    ‘or_else’
        ( say_subject gv e
          ‘before_‘
          say_action gv e e
          ‘before_‘
          say_object gv e
          ‘before_‘
          say_origin gv e
          ‘before_’
          say_rest_event gv e)

say_active_event_sp :: GenVals -> Meaning -> Generator
say_active_event_sp gv e
  = say_subject gv e
    ‘before_’
    (if_ (negative_action)
           say_no
    ‘or_else’
           say_nothing
     )
    ‘before_’
    say_pronouns gv e
    ‘before_’
    say_action_sp gv e
    ‘before_’
    say_object_sp gv e
    ‘before_’
    say_rest_event gv e
```

Figure 4.5: Simplified portion of the NLG Haskell code

simplified example some other of the Haskell features are found:

- Higher-order functions. **if_** and **if_gen** are two examples of higher-order functions as they both take as first parameter a function:
  if_gen (is_style Active) ...

- Currying. The functions return a result of type **Generator** which is in itself a function. The data type could be replaced by the explicit reference to the function's type. Currying allows to make only two parameters explicit when the function receives three. For example, (1) would be replaced by (2)

```
say_event:: GenVals -> Meaning -> GenTypeA -> GentTypeB      (1)
say_event:: GenVals -> Meaning -> Generator             (2)
```

- Lazy Evaluation. For example **if_** and **if_gen** rely on lazy evaluation as only one of their branches is required to be evaluated. For example:

```
if_gen (is_language Spanish)
        (say_active_event_sp gv e)
'or_else'
        (...) -- say_active_event for English
```

  If the language is set to be Spanish only the branch calling the function 'say_active_event_sp gv' is evaluated.

Figure 4.6 shows a simplified and expanded portion of the code generating the adequate verb form. **say_action_sp** decides which is the person and number of the verb (**first_p_sing_m** and the rest of functions from the conditions will check which are the gender and the number of the subject of the event as subject and verb must agree in these two aspects). **say_verb_sp** controls whether the form to be produced is irregular or not (if so, the form will be taken from the SemNet). If the form is regular then a function for the particular tense ('present' in the example) is called, in this case **get_tense_present_indicative**. This function decides to which conjugation (see Section /actions) the verb belongs and calls the corresponding function, **get_p_i_ar** for the first conjugation. Finally this function produces the adequate form of the verb depending on the person and number.

4.5 Chapter Summary

This chapter has discussed the integration of the Spanish generator in the LOLITA generation module and given details of some of the heuristics involved in the plan-realiser component. The implementation of these heuristics, together with many more that have not been detailed have resulted in a plan-realiser component that can successfully produce Spanish utterances from the SemNet representation.

Finally, the chapter has given a brief discussion of the operation of the LOLITA generator and discussed the effects of the chosen implementation language Haskell on the generator's development.

```
say_action_sp :: Tense_gen -> Meaning -> Word
say_action_sp tense e
  = if_ (first_p_sing_m e)
          (say_verb_sp tense verb v is_first_singular)
    `or_else`
    if_ (first_p_plur_m e)
          (say_verb_sp tense verb v is_first_plural)
   .
   .
where
 v = action_spanish e
 verb = spanish_word v


say_verb_sp :: Tense_gen -> Word -> Meaning -> SpVal -> Word
say_verb_sp  Present_ verb v spv
      | (is_irreg_present_m v)  = get_irregular_sp Present_ spv v
      | otherwise = get_tense_present_indicative verb spv


get_tense_present_indicative :: Word -> SpVal -> Word
get_tense_present_indicative verb sp
      | (form_ar verb) = get_p_i_ar w sp
      |   ...
      |   ...
       where
        w = lastn2 verb


get_p_i_ar :: Word -> SpVal -> Word
get_p_i_ar w sp
      | first_sing = w ++ "o"
      | sec_sing = w ++ "as"
      | ...
      |
```

Figure 4.6: Simplified portion of the NLG Haskell code

Chapter 5

Evaluation and Results

5.1 Introduction

There has been little work on NLG evaluation as researchers have concentrated more on development. Evaluation of NLG systems is not an easy task. [Galliers and Sparck Jones, 1993] state in their report on NLP evaluation:

> " Evaluation for NLG remains at the discussion stage. Evaluating generation is difficult; it is hard to define what the input to a generator should be and it is hard to objectively judge the output." (page 98).

Evaluation has to be performed taking into consideration the particular setup and task defined for each generation system. The environment differs from system to system making it infeasible for a comparative evaluation. Furthermore, papers on generation systems are not usually littered with example output so the systems cannot be easily evaluated.

The aim of the generator is to produce correct utterances in Spanish conveying the information contained in the nodes of SemNet. The examples provided in Chapter 4 can be considered as a proof of the success of the Spanish generator. These examples show how the Spanish generator is able to produce utterances for any kind of information (in the form of nodes and links between the nodes) found in the SemNet representation.

Additionally, an experiment has been carried out for the evaluation of the Spanish generator in a similar way to the experiment made to evaluate the LOLITA English generator ([Smith, 1995], Chapter 8). The idea of the evaluation is to compare the utterances generated by LOLITA with those produced by humans. The evaluation will be judged by humans and the results can be considered as a suggestion (the experiment is not an exhaustive evaluation) of how the generator works.

5.2 The evaluation experiment

This experiment intends to evaluate the Spanish generator not the whole LOLITA system or other modules of the system so examples have to be chosen which are correct for other areas of the system (i.e. an output utterance can be incorrect because of a mistake in the analysis of the input text).

The experiment involved two different tests. For both of them two simple input texts which comprised information on twelve events and entities were provided.

First test: Human or Computer

A group of ten people were asked to produce Spanish utterances describing the events and entities mentioned in the given texts. They were asked to write utterances in different levels of detail and styles.

The LOLITA system also did the same task. LOLITA analysed the input texts and built an internal representation (in SemNet) of the meaning of each of the events and entities described. Then, the Spanish generation module produced different utterances from these representations (different utterances about the same entity or event were produced by different setting of the realisation parameters)

Five describing utterances were collected for each entity and event from the set of utterances provided by humans and LOLITA. There were thus sixty utterances, twenty produced by LOLITA and forty produced by humans. They were given to another group of ten people who evaluated them. The judges had to mark each utterance as:

- **H**: The utterance has been produced by a human.
- C: The utterance has been produced by a computer.
- ?: if they cannot tell who has produced the utterance.

Second Test: Acceptability

All the output utterances produced by the Spanish generator from the representation of the two input texts were used for the second test. As stated before different utterances describing a same event or entity are produced by setting different the realisation parameters (Passive/Active, Short/Long generation, Short/Long rhythm, Abstract transformations, etc). The utterances were given to the judges who marked them as **Acceptable** or **Unacceptable**. Comments as to why utterances were marked as unacceptable were also collected.

5.3 Evaluation Results

5.3.1 Human or Computer

From a total of 600 marked utterances the results obtained were as follows:

446 (74%) utterances were marked as being humans or computer.
154 (26%) utterances were not marked, the judges could not tell if the utterance was human or computer generated.
208 (87%) of the utterances marked as human generated were correctly assigned.
99 (48%) of the utterances marked as computer generated were correctly assigned.

These results indicate that the judges were very good at deciding if an utterance was produced by a human but poor at identifying utterances produced by the computer.

5.3.2 Acceptability

From a total of 350 marked utterances the results obtained were as follows:

269 (77%) utterances were marked as acceptable.

Comments were collected to why certain utterances were marked as being unacceptable. One common comment was the dislike of the use of passive sentences. Some people found sentences in passive voice unacceptable as they are not commonly used. Other typical occurring comment was the use of neither the imperfect form or the preterite form of the verb to generate past sentences. As stated in Section 4.3.3, time and tense representation in the LOLITA system is currently under development so simple heuristics are used to generate the tense of the verbs.

All the collected comments can be used for future generator improvements as for example, selecting carefully when to use the passive voice or implementing more complex heuristics to produce the tense of the verbs.

Chapter 6

Conclusion

The overall aim of the project was to enable the LOLITA system to produce Spanish utterances. The results of the evaluation described in Chapter 5, comparing the generator's capabilities with that of humans doing the same task, plus the examples provided alongside of the solutions adopted indicate that this goal has been met.

It was not the aim to build a separate new generator, instead the current generator was modified to generate either English and Spanish. The integration of both generators allows the abstraction of some of the generation principles in order to facilitate the generation of other target languages.

The procedural approach followed by the generator meant that the grammar was not made explicit and was therefore more difficult to understand and modify. However, the methodological principles of Natural Language Engineering such as *scale, robustness, maintainability, flexibility, integration, feasibility and usability* followed in the development of the English generator eased the integration of the new generator in the system. Furthermore, the programming features which are provided by the functional programming language Haskell were of particular help in the implementation of the Spanish generator. The use of Abstract Data Types (ADTs) and the purity of Haskell made it easier to integrate new data structures and algorithms into the existing system. The use of lazy evaluation also proved to be a very useful feature when diverse solutions were adopted depending on the different languages as only the solution corresponding to the language being generated needed to be evaluated.

The generator has been developed alongside SemNet and each has influenced the development of the other. This means that SemNet has to be able to cope with Spanish. For this purpose work has been undertaken in two aspects. Firstly new controls have been added to cope with Spanish linguistic features. These controls refer to aspects such as the gender of the words, the position of the adjectives with respect to the noun they modified and persons, numbers and tenses in the Spanish verbs systems. The second aspect refers to the addition of lexical entries for those concepts which are language isomorphic in Spanish. It should be noted that they are not exclusive for the generator. As part of SemNet, they are accessible for other modules of the system and they can be useful for example for a further Spanish parser.

Scope for future work.

Future work can be undertaken in order to improve the operation of the generator in the following aspects:

- Addition of more Spanish lexical entries to SemNet. The lexical coverage of Spanish is currently underpopulated for a large-scale system.
- Coverage of the generation grammar. The coverage of the grammar is poor compared to some generation systems. However, it is enough to express the meaning represented in the input but following the development of the SemNet representation, the generation grammar has to be extended to cope with possible new SemNet structures.
- Development of a planner. This aspect comprises the whole generation module, not only the Spanish generator.
- Abstraction of some generation principles from the integration of the English and Spanish generator. This abstraction will aid to generate other target languages.

Appendix A

Utterances used for the evaluation

The following utterances were given to a set of people to evaluate them. They were asking to mark each utterance as follow:

- **H**: The utterance has been produced for a human.
- **C**: The utterance has been produced by a computer.
- ?: if they cannot tell who has produced the utterance.

> ‘I was very tired, so I called a taxi and went home. The cab was warm and the driver was nice, so I gave him a big tip”

TAXI:

1. Un taxi que te reconfortó tu cansancio camino de casa.

2. Un taxi que fue llamado por ti para que te llevara a casa descansado.

3. Cogiste un taxi para ir a casa.

4. Un taxi fue llamado por ti porque estabas cansado.

5. El taxi acogedor que llamaste.

DRIVER:

1. El taxista que te llevó a casa el día que estabas cansado y llamaste a un taxi para no andar.

2. El conductor amable a quien le diste una propina grande.

3. Al conductor le diste una gran propina.

4. Un conductor fue amable.

5. El taxista que se portó muy bien y al que le diste una buena propina.

TIP:

1. La propina grande que diste a un conductor.

2. La propina que dejaste al taxista que te llevó a casa.

3. La propina que se ganó el taxista por amable.

4. Una gran propina se la diste al conductor.

5. Una propina grande fue dada por ti a un conductor porque el taxi que llamaste era confortable.

TIREDNESS:

1. Estabas cansado así que te fuiste a tu casa.

2. Fuiste a casa en taxi por el cansancio que tenías.

3. Era mucho lo cansado que estabas por lo que decidiste ir en taxi a casa.

4. Muy cansado te sentías.

5. Comno estabas cansado querías volver al hogar.

THE GOING event:

1. Te fuiste a casa en taxi porque estabas muy cansado para ir andando.

2. Fuiste a casa en taxi por el cansancio.

3. Te fuiste a tu casa porque estabas cansado.

4. En taxi fuiste a casa.

5. Fuiste a llamar a un taxi que te llevaría a casa.

THE CALLING event:

1. Llamaste a un taxi para ir a casa. 2. Llamaste a un taxi porque estabas cansado y querías volver a casa.

3. Fuiste a llamar a un taxi que te llevaría a casa.

4. Llamaste a un taxi. Estabas cansado. Te fuiste a tu casa.

5. El taxi que llamaste era confortable así que diste una propina grande a un conductor. Estabas cansado así que te fuiste a tu casa.

'if I had known the big and fast motorbike you gave me was owned by her I would have liked it, because I do love her.'

MOTORBIKE:

1. Una moto grande y rápida de una propietaria fue dada por mí a ti.

2. La moto que era de la mujer.

3. Una moto grande y rápida de una propietaria que te di y que te habría gustado si lo hubieras sabido que la tuvo.

4. Su moto era grande y rápida.

5. La moto grande y rápida que pertenecí a la propietaria que amas.

SHE:

1. La mujer poseía una moto que era grande y rápida y que ahora amas.

2. Una propietaria tuvo la moto grande y rápida. 3. Ella era la propietaria de la moto que te di.

4. La dueña de la moto que te di.

5. La propietaria que tuvo la moto grande y rápida que te di y a quien amas.

THE LIKING event:

1. Si hubieras sabido que era de ella te hubiese gustado.

2. Si hubieras sabido que una propietaria tuvo la moto grande y rápida, te habría gustado. Te la di. La amas.

3. Como la amas a ella, si hubieses sabido que la moto le pertenecía, te habría gustado.

4. Tú no sabías que la moto era de ella. En caso contrario te hubiera gustado.

5. Te habría gustado la moto que te di de haber sabido que antes fue de la chica a quien amas.

THE GIVING event:

1. Te di una moto que era de su propiedad.

2. Te di una moto que te habría gustado un montón si llegas a saber que la tuvo antes la chica que amas.

3. Te di una moto que pertenecía a la mujer que es el amor de tu vida.

4. Te di la moto de tu amada.

5. Te di una moto grande y rápida de una propietaria que te habría gustado si lo hubieras sabido que la tuvo.

THE KNOWING event:

1. Tú no sabías que la moto era de ella. En caso contrario te hubiera gustado.

2. Saber que la moto que te di pertenecía a ella, hubiese cambiado tu actitud porque la amas.

3. Si hubieras sabido que la moto grande y rápida que te di fue tenida por la propietaria a quien amas, sería gustada por ti.

4. Si hubieras sabido que era de ella te hubiera gustado.

5. Si hubieras sabido que una propietaria tuvo la moto grande y rápida que te di, te habría gustado.

THE LOVING event:

1. Amas a la propietaria que tuvo la moto grande y rápida que te di. Te habría gustado si lo hubieras sabido que la tuvo.

2. Ella es amada por ti.

3. La quieres. Desconocías que la moto grande y rápida que te di era de ella.

4. Amas a una propietaria.

5. Amas a una mujer de la cual posees ahora la moto que ella tuvo hace tiempo.

Appendix B

Variations for generator output

The following utterances produced from the giving input texts show some of the different styles of output LOLITA is able to produce. These utterances were used for the second part of the evaluation experiment as they were given to a group of people who marked them as *acceptable* or *Unacceptable* utterances.

> ‘I was very tired, so I called a taxi and went home. The cab was warm and the driver was nice, so I gave him a big tip”

Diste una propina grande a un conductor porque el taxi que llamaste era confortable. Estabas cansado así que te fuiste a tu casa.

Una propina grande fue dada por ti a un conductor porque el taxi que llamaste era confortable. Estabas cansado así que te fuiste a tu casa.

Diste una propina grande a un conductor. Un taxi era confortable. Lo llamaste. Te fuiste a tu casa porque estabas cansado.

El taxi que llamaste era confortable así que diste una propina grande a un conductor. Estabas cansado así que te fuiste a tu casa.

El conductor amable a quien le diste una propina grande.

Un conductor fue amable así que le diste una propina grande.

Un conductor fue amable.

Un conductor fue amable así que una propina grande fue dada por ti.

La propina grande que diste a un conductor.

Llamaste un taxi porque estabas cansado.

Una propina grande fue dada por ti a un conductor porque el taxi que llamaste era confortable. Estabas cansado así que te fuiste a tu casa.

Diste una propina grande a un conductor. Un taxi era confortable. Lo llamaste. Estabas cansado. Te fuiste a tu casa.

Un taxi era confortable. Lo llamaste. Estabas cansado. Te fuiste a tu casa. Diste una propina grande a un conductor.

Estabas cansado así que te fuiste a tu casa.

Llamaste a un taxi. Estabas cansado. Te fuiste a casa.

El taxi acogedor que llamaste.

Un taxi fue llamado por ti porque estabas cansado.

Un conductor fue amable. Le diste una propina grande. Un taxi era confortable . Lo llamaste. Estabas cansado. Te fuiste a tu casa.

Un conductor te recibi una propina grande porque el taxi que llamaste era confortable. Estabas cansado así que te fuiste a tu casa.

> 'if I had known the big and fast motorbike you gave me was owned by her I would have liked it, because I do love her.'

Amas a la propietaria que tuvo la moto grande y rápida que te di. Te habría gustado si lo hubieras sabido que la tuvo.

Amas a una propietaria. Tuvo una moto grande y rápida. Te la di. Te habría gustado si lo hubieras sabido que la tuvo.

Amas a una propietaria.

La propietaria a quien amas tuvo la moto grande y rápida que te di y que te habría gustado si lo hubieras sabido que la tuvo.

Te di una moto grande y rápida de una propietaria.

Te di una moto grande y rápida de una propietaria y que te habría gustado si lo hubieras sabido que la tuvo.

La propietaria que tuvo la moto grande y rápida que te di y a quien amas.

Una moto grande y rápida de una propietaria que te di.

Si hubieras sabido que la propietaria a quien amas tuvo la moto grande y rápida que te di, te habría gustado.

Una moto grande y rápida de una propietaria fue dada por mí a ti.

Me recibiste una moto grande y rápida de una propietaria que te habría gustado si lo hubieras sabido que la tuvo.

Si hubieras sabido que una propietaria tuvo una moto grande y rapida, te habría gustado.

Una moto grande y rápida de una propietaria que te di te habría gustado si lo hubieras sabido que la tuvo.

La propietaria que tuvo la moto grande y rápida que te di es amada por ti. Sería gustada por ti si lo hubieras sabido que la tuvo.

Una propietaria tuvo la moto grande y rápida. Te habría gustado si lo hubieras sabido que la tuvo. Te la di. La amas.

Bibliography

[ANLP-92, 1992] *Proceedings of the Third ACL Conference on Applied Natural Language Processing*, Trento, Italy, 1992.

[Appelt, 1983] D. E. Appelt, "TELEGRAM: A Grammar Formalism for Language Planning", in *Proceedings of the Eighth International Joint Conference on Artificial Intelligence (IJCAI-83)*, pages 595–599, Karlsruhe, West Germany, August 8-12, 1983.

[Appelt, 1985] D. E. Appelt, *Planning English Sentences*, Cambridge University Press, Cambridge, UK, 1985.

[Bird and Wadler, 1988] P. Bird and P. Wadler, *Introduction to Functional Programming*, Prentice Hall International (UK) Ltd., 1988.

[Bokma and Garigliano, 1992] A. F. Bokma and R. Garigliano, "Uncertainty Management through Source Control: A Heuristic Approach", in *Proceedings, International Conference on Information Processing and Management of Uncertainty in Knowledge-Based Systems*, Mallorca, Spain, July 1992.

[Butt and Benjamin, 1994] J. Butt and C. Benjamin, *A New Reference Grammar of Moderm Spanish*, Edward Arnold, London, 1994.

[Cawsey, 1990] A. Cawsey, "Generating Explanatory Discourse", in Dale et al. [1990], pages 75–101.

[Cline, 1994] B. E. Cline, *Knowledge Intensive Natural Language Generation with Revision*, PhD thesis, Virginia Polytechnic Institute and State University, Blacksburg, Virginia, May 1994.

[Dale *et al.*, 1990] R. Dale, C. Mellish, and M. Zock, editors, *Current Research in Natural Language Generation*, Academic Press, New York, 1990.

[Dale *et al.*, 1992] R. Dale, E. H. Hovy, D. Rösner, and O. Stock, *Aspects of Automated Natural Language Generation*, Lecture Notes in Artificial Intelligence, 587, Springer-Verlag, Berlin, April 1992.

[Delin *et al.*, 1994] J. Delin, A. Hartley, C. Paris, D. Scott, and K. V. Linden, "Expressing Procedural Relationships in Multilingual Instructions", in NLG94 [1994], pages 61–70.

[Dogru and Slagle, 1992] S. Dogru and J. R. Slagle, "A System That Translates Conceptual Structures Into English", in Nagle et al. [1992].

[Elhadad and Robin, 1992] M. Elhadad and J. Robin, "Controlling Content Realization with Functional Unification Grammars", in *Aspects of Automated Natural Language Generation* [1992], pages 89–104.

[Fawcett and Davies, 1992] R. P. Fawcett and B. L. Davies, "Monologue as a Turn in Dialogue: Towards an Integration of Exchange Structure and Rhetorical Structure Theory", in *Aspects of Automated Natural Language Generation* [1992], pages 151–166.

[Fawcett and Tucker, 1990] R. P. Fawcett and G. H. Tucker, "Demonstration of GENESYS: A very large, semantically based systemic functional generator", in *Proceedings of the 13th International Conference on Computational Linguistics (COLING-90)*, volume 1, pages 47–49, Helsinki, 1990.

[Fernandez, 1986] S. Fernandez, *Gramática Española. 4. El Verbo y La Oración*, ARCO/LIBROS,S.A., Madrid, 1986.

[Fox and Long, 1995] M. Fox and D. Long, "Hierarchical Planning using Abstraction", *to appear in IEE Procs. Control Theory and Applications*, May 1995.

[Galliers and Sparck Jones, 1993] J. Galliers and K. Sparck Jones, "Evaluating Natural Language Processing Systems.", Technical Report 291, Computer Laboratory, University of Cambridge, 1993.

[Garigliano and Jones, 1992] R. Garigliano and C. Jones, "Dialogue Structure Models: An approach to dialogue analysis and generation by computer", Technical Report 1/92, School of Engineering and Computer Science, University of Durham, UK, 1992

[Garigliano *et al.*, 1992] R. Garigliano, R. G. Morgan, and M. H. Smith, "LOLITA : Progress Report 1.", Unpublished Research Report 12/92, Department of Computer Science, University of Durham, 1992.

[Garigliano *et al.*, 1993] R. Garigliano, R. G. Morgan, and M. H. Smith, "The LOLITA System as a Contents Scanning Tool", in *Proceedings of the 13th International Conference on Artificial Intelligence, Expert Systems and Natural Language Processing*, Avignon, France, May 1993.

[Garigliano, 1992] R. Garigliano, "A Computational Semantics for 'The'", Technical report, Department of Computer Science, Durham University, 1992.

[Garigliano and Tate, 1995] R. Garigliano and J. Tate, editors, *Journal of Natural Language Engineering, Volume 1*, Cambridge University Press, 1995

[Goldman, 1975] N. M. Goldman, "Conceptual Generation", in R. C. Schank and C. K. Riesbeck, editors, *Conceptual Information Processing*, American Elsevier, New York, NY, 1975.

[Granville, 1994] R. Granville, "Building Underlying Structures for Multiparagraph Texts", in NLG94 [1994], pages 21–28.

[Halliday, 1985] M. A. K. Halliday, *An Introduction to Functional Grammar*, Edward Arnold, London, 1985.

[Hazan *et al.*, 1993] J. Hazan, S. Jarvis, and R. Morgan, "Understanding LOLITA: Program Comprehension in Functional Languages", in *Proceedings of IEEE Conference on Program Comprehension*, Capri, Italy, June 1993.

[Horacek, 1990] H. Horacek, "The Architecture of a Generation Component in a Complete Natural Language Dialog System", in Dale et al. [1990], pages 193–227.

[Hovy, 1988b] E. H. Hovy, *Generating Natural Language under Pragmatic Constraints*, Lawrence Erlbaum Associates, Hillsdale, NJ, 1988, Based on PhD thesis, Yale University.

[Hovy, 1991] E. H. Hovy, "Approaches to the Planning of Coherent Text", in Paris et al. [1991], pages 83–102.

[Hovy, 1993] E. H. Hovy, "Automated Discourse Generation Using Discourse Structure Relations.", *Artificial Intelligence*, 63:341–385, 1993.

[Hudak *et al.*, 1994] P. Hudak, S. P. Jones, and P. Wadler, "Report on the Functional Programming Language Haskell, Version 1.2", May 1994, ACM SIGPLAN Notices 27.

[Iordanskaja *et al.*, 1991] L. Iordanskaja, R. Kittredge, and A. Polguère, "Lexical Selection and Paraphrase in a Meaning-Text Generation Model", in Paris et al. [1991], pages 293–312.

[Jones, 1994] C. Jones, *Dialogue Structure Models : An Engineering Approach to the Analysis and Generation of Natural English Dialogues*, PhD thesis, Department of Computer Science, Durham University, 1994.

[Joshi, 1987] A. K. Joshi, "The Relevance of Tree Adjoining Grammar to Generation", in Kempen [1987], pages 233–252.

[Kantrowitz and Bates, 1992] M. Kantrowitz and J. Bates, "Integrated Natural Language Generation Systems", in *Aspects of Automated Natural Language Generation* [1992], pages 13–28.

[Kay, 1979] M. Kay, "Functional Grammar", in *Proceedings of the 5th Annual Meeting of the Berkeley Linguistic Society*, pages 142–158, Berkeley, CA, February 17-19, 1979.

[Kempen, 1987] G. Kempen, editor, *Natural Language Generation: New Results in Artificial Intelligence, Psychology and Linguistics*, NATO ASI Series – 135, Martinus Nijhoff Publishers, Boston, Dordrecht, 1987.

[Kittredge et al, 1991] R. Kittredge, T. Korelsky and O. Rambow, "On the need for Domain Communication Knowledge", in Computational Intelligence 7(4).

[Kukich, 1988] K. Kukich, "Fluency in Natural Language Reports", in McDonald and Bolc [1988], pages 280–311.

[Long and Fox, 1995] D. Long and M. Fox, "A Hybrid Architecture for Rational Agents", in C.Thornton and S.Torrance, editors, *Hybrid Models of Cognition*, AISB, 1995.

[Long and Garigliano, 1994] D. Long and R. Garigliano, *Reasoning by Analogy And Causality: A Model and Application*, Ellis Horwood, 1994.

[Mann and Thompson, 1987] W. C. Mann and S. A. Thompson, "Rhetorical Structure Theory: Description and Construction of Text Structures", in Kempen [1987], pages 85–96, Also appears as USC/Information Sciences Institute Tech Report RS-86-174, October 1986.

[Mann, 1983a] W. C. Mann, "An Overview of the NIGEL Text Generation Grammar", in *Proceedings of the 21st Annual Meeting of the ACL*, pages 79–84, Massachusetts Institute of Technology, Cambridge, MA, June 15-17, 1983.

[Mann, 1983b] W. C. Mann, "An Overview of the Penman Text Generation System", in *Proceedings of the Third National Conference on Artificial Intelligence (AAAI-83)*, pages 261–265, Washington, DC, August 22-26, 1983.

[McDonald and Bolc, 1988] D. D. McDonald and L. Bolc, *Natural Language Generation Systems*, Springer-Verlag, New York, NY, 1988.

[McDonald *et al.*, 1987] D. D. McDonald, M. M. Meteer, and J. D. Pustejovsky, "Factors Contributing to Efficiency in Natural Language Generation", in Kempen [1987], pages 159–182.

[McKeown and Swartout, 1988] K. R. McKeown and W. R. Swartout, "Language Generation and Explanation", in Zock and Sabah [1988], chapter 1, pages 1–52.

[McKeown *et al.*, 1990] K. R. McKeown, M. Elhadad, Y. Fukumoto, J. Lim, C. Lombardi, J. Robin, and F. A. Smadja, "Natural Language Generation in COMET", in Dale et al. [1990], pages 103–139.

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

[Mel'čuk and Polguère, 1970] I. Mel'čuk and A. Polguère, "Towards a Functioning Meaning-Text Model of Language", *Linguistics*, 57:10–47, 1970.

[Meteer, 1993] M. Meteer, *Expressibility and the Problem of Efficient Text Planning*, Francis Pinter Publishers, London, 1993.

[Nagle *et al.*, 1992] T. Nagle, J. Nagle, L. Gerholz, and P. Elklund, editors, *Conceptual Structures: Current Research and Practice*, Ellis Horwood, New York, NY, 1992.

[Nirenburg *et al.*, 1988] S. Nirenburg, R. McCardell, E. Nyberg, P. Werner, E. Kenschaft, S. Huffman, and I. Nirenburg, "DIOGENES-88", Technical Report CMU-CMT-88-107, Center for Machine Translation, Carnegie Mellon University, 1988.

[NLG94, 1994] *Proceedings of the Seventh International Workshop on Natural Language Generation*, Nonantum Inn, Kennebunkport, Maine, June 21-24 1994.

[Nogier and Zock, 1992] J. Nogier and M. Zock, "Lexical Choice as Pattern Matching", in Nagle et al. [1992].

[Paris and McKeown, 1987] C. L. Paris and K. R. McKeown, "Discourse Strategies for Describing Complex Physical Objects", in Kempen [1987], pages 97–116.

[Paris *et al.*, 1991] C. L. Paris, W. R. Swartout, and W. C. Mann, editors, *Natural Language Generation in Artificial Intelligence and Computational Linguistics*, Kluwer Academic Publishers, Boston, 1991.

[Paris, 1993] C. L. Paris, *User Modelling in Text Generation*, Francis Pinter Publishers, London, 1993.

[Patten, 1986] T. Patten, *Interpreting Systemic Grammar as a Computational Representation: A Problem Solving Approach to Text Generation*, PhD thesis, Edinburgh University, Department of Artificial Intelligence, 1986.

[Patten, 1988] T. Patten, *Systemic text generation as problem solving*, Cambridge University Press, New York, 1988, Based on PhD Thesis [Patten, 1986].

[Rösner and Stede, 1992] D. Rösner and M. Stede, "Customizing RST for the Automatic Production of Technical Manuals", in *Aspects of Automated Natural Language Generation* [1992], pages 199–214.

[Scott and de Souza, 1990] D. R. Scott and C. S. de Souza, "Getting the Message Across in RST-based Text Generation", in Dale et al. [1990], pages 47–73.

[Selinker, 1969] L. Selinker, "Language Transfer", General Linguistics 9, pp. 69-92, 1969

[Shapiro and the SNePS Implementation Group, 1993] S. C. Shapiro and the SNePS Implementation Group, "SNePS 2.1 User's Manual", Technical report, State University of New York at Buffalo, Buffalo, NY, 1993.

[Shapiro, 1982] S. C. Shapiro, "Generalized ATN Grammars for Generation from Semantic Networks", *Computational Linguistics*, 8:12–26, 1982.

[Short and Garigliano, 1993] S. Short and R. Garigliano, "The Representation of Location in LOLITA", Unpublished research report, Department of Computer Science, University of Durham, March 1993.

[Short, forthcoming 1995] S. Short, *Semantic Representation and Analysis in the LOLITA System*, PhD thesis, Department of Computer Science, University of Durham, forthcoming, 1995.

[Simmons and Slocum, 1972] R. F. Simmons and J. Slocum, "Generating English Discourse from Semantic Networks", *Communications of the ACM*, 15(10):891–903, October 1972.

[Smith, 1995] M. H. Smith, *Natural Language Generation in the LOLITA System: An Engineering Approach.*, PhD thesis, Department of Computer Science, Durham University, 1995.

[Sowa, 1984] J. F. Sowa, *Conceptual Structures (Information Processing in Mind and Machine)*, Addison-Wesley, 1984.

[Turner, 1982] D. Turner, "Recusion Equations as a Programming Language", in Darlington, editor, *Functional Programming and Its Applications*, Cambridge University Press, 1982.

[van Rijn, 1992] A. van Rijn, "Generating Language from Conceptual Dependency Graphs", in Nagle et al. [1992].

[Vander-Linden *et al.*, 1992] K. Vander-Linden, S. Cumming, and J. Martin, "Using System Networks to Build Rhetorical Structure", in Vander-Linden [1992], pages 183–198.

[Wang and Garigliano, 1992] Y. Wang and R. Garigliano, *An Intelligent Tutoring System for Handling Errors Caused by Transfer*, Lecture Notes in Artificial Intelligence, 608, Springer-Verlag, Montreal, Canada, 1992.

[Wang, 1994] Y. Wang, *An Intelligent Computer-based Tutoring Approach for the Management of Negative Transfer*, PhD thesis, Department of Computer Science, Durham University, 1994.

[Wanner, 1994] L. Wanner, "Building Another Bridge over the Generation Gap", in NLG94 [1994], pages 137–144.

[Woods, 1970] W. Woods, "Transistion Network Grammars for Natural Language Analysis", *Communications of the ACM*, 13(10):591–606, October 1970.

[Zock and Sabah, 1988] M. Zock and G. Sabah, editors, *Advances in Natural Language Generation: An Interdisciplinary Perspective*, volume 1, Ablex Publishing Corporation, Norwood, NJ, 1988.