Chapter 7

Process Model Overview

1. Process Model Overview

In Part A of this thesis, I described the resource model behind a single-sentence processing system. I described the three strata of this model -- graphology, lexico-grammar and semantics, as well as the resources for mapping between these strata.

Part B will look at how these resources are used to process sentences -- both generation of sentences from a micro-semantic specification, and analysis of sentences up to micro-semantic form.

1.1 Sentence Processing as Staged Re-Representation

Figure 7.1 presents WAG's sentence processing architecture, allowing both generation and analysis. Each of the modules is implemented, except for the Speech Recogniser: at present, input to the parser is typed, or read from a text file. For Speech Synthesis, I use the Macintosh Speech Manager, text-to-speech software supplied with the system software.

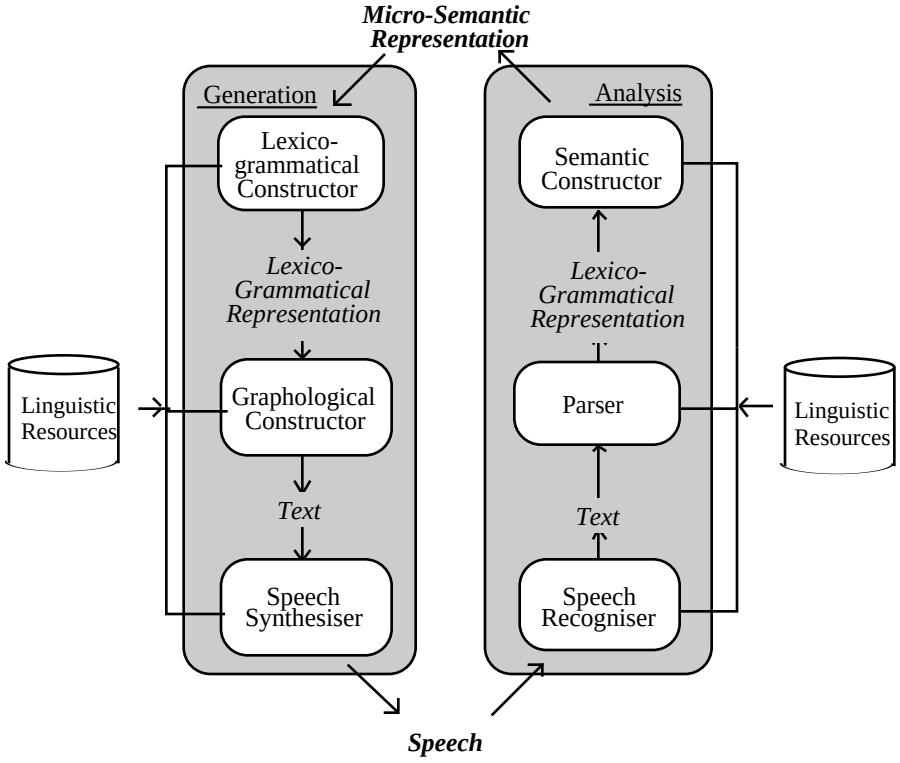

Figure 7.1: Sentence Processing

Both generation and analysis can be seen as a series of staged transformations from an input representation to an output representation. Each of the stages, or modules, is abstractly the same, involving the re-representation of a *source representation* into a *target representation*. For instance, the first step in the generation process in figure 7.1 involves a micro-semantic specification (source representation), being re-represented as a lexico-grammatical representation (target representation). Figure 7.2 represents a generic processing module, the basic element from which figure 7.1 is constructed.

While each module is abstractly the same as any other (a transformation of representations), the actual strategy used to perform the transformation may vary, since the nature of the source and target representations may vary. This chapter will look at various processing strategies: different means of controlling the translation from source to target

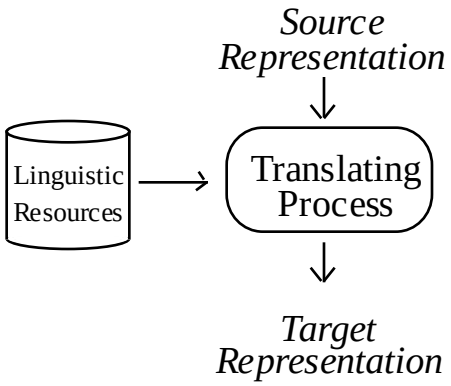

Figure 7.2: The Basic Component of Sentence Processing

I will also discuss how the individual modules are structured together to make the macro-process -- the sentence analysis or generation system. In this regard, several architectures for organising the flow of information between modules will be discussed.

1.2 Structure of Part B

Chapter 8 will look at processing at a formal level, the manipulation of Systemic features and Systemic structures:

a) Operations Involving Features: forward-chaining, backward-chaining (finding the path of a feature or feature-complex), path unification, and feature negation.

b) Operations Involving Systemic Structures: assertion of Systemic constraints, structure unification and structural negation.

The remaining chapters in part B will address specific types of processing -- generation and analysis.

- **Sentence Analysis**: the operations involved in the analysis of a sentence. This can be broken up into stages, for instance, graphological analysis, lexical (morphological) analysis, grammatical analysis and micro-semantic analysis. I deal with analysis in two chapters:

  a) General Parsing Strategies: a summary of parsing strategies such as top-down *vs.* bottom-up, depth-first *vs.* breadth-first, means of handling non-determinism, etc. (chapter 9).

  b) The WAG analyser: a description of the analysis algorithm used in the WAG system (chapter 10).

- **Sentence Generation**: the operations involved in the generation of a sentence (chapter 11):

a) General Generation Issues: an overview of the general issues involved in generation;

b) The WAG generator: a description of the sentence generation algorithm used in WAG, and also a description of the micro-semantic input specification to the generator.

2. Some Processing Issues

Viewing sentence processing as staged translation between representations, there are various issues which need to be addressed. These are summarised below.

2.1 Control Strategies

In this section, I outline different means of controlling the translation from a source representation to a target representation.

The resources which map between one representation and another can be considered as a set of *constraints* -- small fragments which map part of the source representation onto part of the target. The consecutive application of these constraints incrementally builds the target representation. For example, figure 7.3 shows a semantics-grammar mapping constraint, derived from Patten's mapping resource (see chapter 6, section 1.2.2), but shown in a direction-neutral form[1].

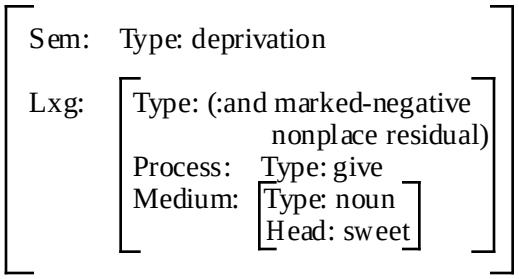

Figure 7.3: Part of Patten's Mapping Resource
Re-Expressed as a Mapping Constraint

The control strategy concerns the manner of choosing:

a) which constraint to apply next;

b) where in the source or target structure to apply it to.

2.1.1 Choosing the Constraint

The choice of which constraint to apply next can be determined either in relation to the resources, or to the representations being mapped:

a) **Resource-Driven:** A constraint is chosen from the resources, and applied to each node of the source or target representation. Some shift-reduce parsers, for instance, cycle through the list of structure rules, applying each in turn to the parse tree.

a) **Representation-Driven**: A node (e.g., a unit in a semantic structure) of the source or target representation is selected as the point of expansion. Constraints are selected on the basis of information contained in that node, and applied. For instance, we could select a constraint associated with one of the features of the current node.

---

[1]This direction-neutral form of representation is based on that used by Bateman *et al.* (1992).

2.1.2 Choosing the Node

A process will generally use either the source or the target structure to control the structural mapping:

1. **Source-Driven**: the process uses the source representation as the centre of constraint application. A sentence generator, for instance, could take each node of the semantic representation in turn, and apply relevant constraints.
2. **Target-Driven**: the process uses the target representation as the centre of constraint application. The Penman sentence generator, for instance, constructs the top-level unit of the lexico-grammatical representation (clause rank), before completing each of its constituents.

My use of the terms *source-driven* and *target-driven* has been adapted from Anwyl *et al.* (1991), where the terms *target-language-driven* and *source-language-driven* are used, in the context of machine translation. McDonald (1983) calls these *data-directed* and *goal-directed* control strategies.

Once we have chosen which of the source or target representations to use to control the mapping, we need to determine which node of the representation to expand first. Do we start at the top of the structure tree, and work down towards the leaves (top-down), or should we start at the leaves of the tree and work upwards?. Other issues in node selection involve the choice between breadth-first *vs.* depth-first strategies, and left-to-right *vs.* unordered strategies. These issues will be discussed in more detail in respect to parsing in chapter 9.

2.2 Flow of Information

Natural Language processes are typically broken down into modules, each module being responsible for one step in the translation from input to output. Graphological analysis, for instance, is a module in a sentence analysis process. Lexico-grammatical analysis comprises another module. An important issue in NLP involves the *flow of information* between these modules -- how information is passed between modules.

2.2.1 Conduit Architectures

The simplest information flow system is based on the *conduit* (or *pipeline*) metaphor:

> "The conduit metaphor refers to the treatment of language as a pipeline or conduit that relays information from the speaker to the hearer. The speaker, starting with some idea of what he wants to say, "packages" it in natural language and sends the package through the conduit to the hearer, who then unwraps the package and removes the contents." (Appelt 1985, p6).

Figure 7.4a shows a generic conduit architecture, where some input is processed by module 1 to produce an intermediate representation, which is itself the input to a second processing module, which produces the output of the process as a whole.

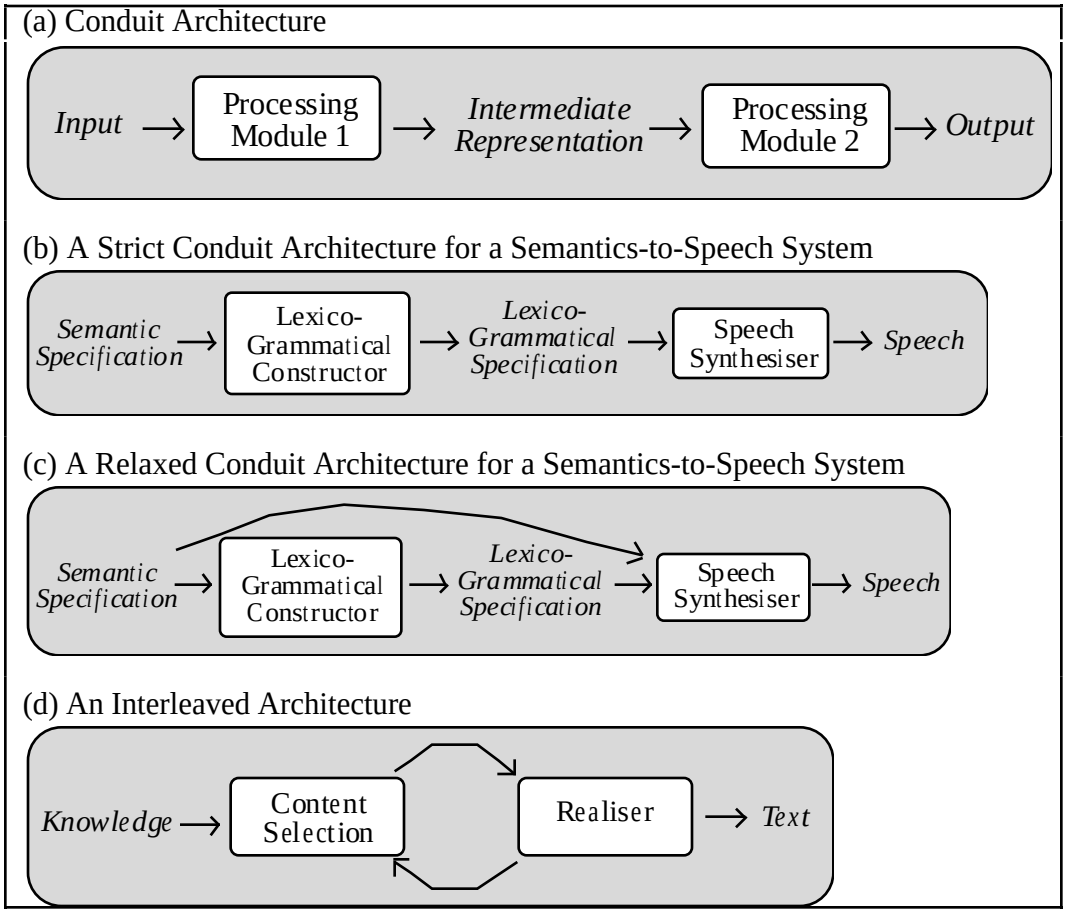

Figure 7.4: Various Information-Flow Architectures

Kantrowitz & Bates (1992) discuss several assumptions made in the conduit metaphor:

1) **The problem is divided into a number of modules, or stages**: Each module performs some processing. Modules may correspond to one or more levels of linguistic representation;
2) **The modules are related to each other only through the sharing of the products of each module:** The results of processing in one module needs to be made available to the other modules in some way. A module cannot affect or audit the internal workings of another module directly. Modules communicate only through their output;
3) **Information flow cannot loop:** A module cannot take its input from a module which has access to the output of that module. In other words, information can move in one direction only.

There are two variants of the conduit architecture, which I will label *strict* and *relaxed* conduit architectures:

1. **Strict Conduit Architecture**: Each module can access only the output of the preceding module (the first module is provided some input by the user). Such systems require information to be passed through successive modules until needed. Figure 7.4b shows a strict conduit architecture for a Semantics-to-Speech system. The speech synthesizer must base all of its decisions on the information contained in the output of the lexico-grammatical constructor. Halliday (1973), for instance, advocates such a system for the communication between semantics and phonology, which is mediated through the lexico-

grammar. Speech-function is encoded as grammatical 'key', which is then expressed through intonational contour.

2. **Relaxed Conduit Architectures**: Each module can access any existing representation, including the output of any preceding modules, and the initial input specification. Figure 7.4c shows a relaxed version of figure 7.4b, where the speech-synthesiser can access not only the lexico-grammatical structure, but also the semantic specification which fed into the lexico-grammatical constructor. It might be useful, for instance, for the speech-synthesiser to appeal directly to the semantic specification to determine the speech-function of the utterance, which is relevant when choosing the intonational contour. Also, the tonic stress of the utterance might be determined by reference to the relevance and recoverability status of the content, perhaps available in the semantic specification.

2.2.2 Interleaved Architectures

A second information-flow architecture drops the assumption of the conduit model that requires information to flow in one direction only between modules. In an *interleaved* architecture (McDonald 1980; Hovy 1985, 1988b), two modules can take input from each other, so construction within one module can be co-ordinated with the construction in the other[2]. For instance, an initial module might process until it requires the second module to make a decision. Process control is passed to that module, which may keep control until it in turn needs the first module to make some decision, whence control is returned to the first module. Control can be passed back and forth in this manner until both processes reach termination. This is called an interleaved architecture because of the back-and-forth movement of the decision-making. Figure 7.4d shows such a system for multi-sentential generation, whereby content selection is interleaved with the realisation of the content.

The popularity of the interleaved approach is growing in the generation area, since decisions about *what to say* (content selection module) depend on the choice of *how to say* (realisation module). For example, the when expressing a process, the decision to include the Actor and Actee will depend on whether a clause or nominal-group is used to express the process -- nominal expression including both Actor and Actee is awkward, and thus less likely, e.g., *the buying of the book by Mary*.

Hovy (1988b) talks of the interleaved architecture as a *limited-commitment* strategy: processing in the first module is delayed until necessitated by the second process: "the planner need assemble only a partial set of generator instructions -- enough for the realisation component to start working on -- and can then continue planning when the realisation component requires further guidance." (p4).

2.2.3 Integrated Processing

A third architecture for handling information flow is called an integrated architecture, first implemented by Appelt (1985) (for discussion, see Hovy 1985, 1988b; Kantrowitz & Bates 1992). An integrated architecture does away with the notion of separate processing modules -- a single process is used which ignores boundaries between representational levels. For instance, Kantrowitz & Bates (1992) describe their integrated generation system as follows:

> "...we have tried to identify commonalities in the operations and representations used at each linguistic level and to generalize them into a single framework that can be used for all aspects of generation. The resulting integrated generator has a single engine, with any modularisation of the generator appearing solely in the organization of its rules." (p14).

---

[2] An interleaved architecture might involve more than two modules.

Hovy (1988b) offers another description of an integrated generator (although he himself opts for an interleaved approach):

> "With the integrated approach, planning and generation is one continuous process: the planner-realizer handles syntactic constraints the same way it treats all other constraints (such as focus or lack of requisite hearer knowledge), the only difference being that syntactic constraints tend to appear late in the planning-realisation process."

Integrated processing is growing in popularity, since it is becoming more common to build natural language systems on top of dedicated formalism processors, for instance, unification systems (e.g., PATR (Shieber 1984); FUF (Elhadad 1991); TFS (Emele & Zajac 1990); ALE (Carpenter 1992)), or knowledge-representation systems (e.g., Loom (MacGregor & Bates 1987)).

2.3 Deterministic vs. Non-Deterministic Processing

In both analysis and generation, a central issue concerns whether or not processing should be *deterministic* -- should the processor definitely resolve each decision when it is reached, or should the processor allow the decision to remain unresolved, until further processing supports one choice over the other.[3]

In generation, we may reach a point where two alternative means of expressing the semantics both seem open. One path, however, may lead to a dead-end in the generation process -- the chosen alternative does not lead to any appropriate realisation. The existence of combinations of choices which do not lead to realisation has been called a *generation gap* (Meteer 1990).

In analysis, there may be two alternative lexemes for a particular spelling, for instance ‘orange’ as a noun and ‘orange’ as an adjective. Choosing one may lead to a correct sentence analysis, the other not leading to any analysis at all.

Deterministic processors are set up so that whenever a decision needs to be made, sufficient information can be obtained to make an appropriate choice. A deterministic parser might look at the next few words of the sentence, to see if one alternative can be eliminated. A deterministic generator might have its resources configured such that whenever a decision must be made, the necessary information is already at hand.

Non-deterministic processors do not make a definite choice, but either follow one alternative, until proven false, and then try another alternative (backtracking architecture), or else pursues all alternatives simultaneously (parallel architecture).

Deterministic and non-deterministic processing approaches will be considered further in chapter 9 (in relation to parsing), and chapter 11 (in relation to generation).

3. Processing Demands on Resources

This section looks at the demands that sentence processing places on the resource model: how the nature of the resources is shaped by the types of processing performed.

3.1 Bi-directionality

Bi-directionality has been an issue in the past decade, for several reasons. As researchers solve issues in one processing-direction, they have attempted to re-use their

---

[3] Sometimes, at the end of processing, the decision has still not been resolved. This is true in the case of ambiguous sentences (in analysis), or alternative realisations (in generation).

resources for the reverse direction. Kasper (1988a), for instance, discusses the re-use of the Nigel generation grammar for analysis.

Another factor is the growing focus on systems which involve both analysis and generation, e.g., dialogue systems and text-summarisation systems. Rather than have a distinct grammar for each process, it is easier to provide a single resource, which can be used in both processes. In machine-translation, where analysis and generation use language-differentiated resources, the use of bi-directional resources in each language allows translation in both directions.

The WAG system has been designed as a bi-directional system -- using the same linguistic resources for both analysis and generation. Most work using Systemic grammar bi-directionally has involved some hand-translation between the generation grammar and the analysis grammar. The parsing grammar used by Weerasinghe & Fawcett (1993), for instance, requires hand-translation from the Genesys generation grammar. The bulk of Kasper's (1988a) parsing grammar is automatically produced from the Nigel generation grammar. However, for parsing purposes, the user needs to specify additional information, in the form of a context-free backbone for the grammar, and the mapping between this sub-grammar and the Nigel grammar.

Bateman *et al.* (1992) use the term *non-directional* resources rather than *bi-directional*: this term brings out more fully the fact that the resources are neutral between either process.

The movement from a direction-dedicated resource to a bi-directional one is usually not straight-forward. My experience has shown that three types of modifications may be required to produce a bi-directional resource:

1. **Addition of Information**: Each process makes particular demands on the resources, and a grammar defined for one process may lack information necessary for another. Often additional information needs to be added to support the new process. For instance, in the Nigel generation grammar, the bulk of the ordering is derived from a set of *default ordering statements*, which provide the orderings of constituents, unless explicitly ordered by the realisations of features. For analysis, default orderings do not make much sense, we rather require explicit statement of possible orderings. To use the Nigel grammar for parsing in the EDA project (University of Sydney -- see Matthiessen, O'Donnell & Zeng 1991), explicit ordering information was added to the realisation component of the lexico-grammar.
2. **Re-Representation of Existing Information**: Often information is present in the resources, but not in a form suitable for bi-directional use. For instance, the inquiries of Penman's chooser-inquiry interface are procedurally-implemented, and are thus dedicated to generation. To make these resources available for parsing (an ISI project), Robert Kasper and myself declarativised the inquiries, into a form suitable for both generation and parsing.
3. **Extension of Coverage:** A generation grammar does not need to handle all possible ways of expressing a given chunk of ideation -- as long as it has one then it can produce output. A parser, on the other hand, must handle what appears in the text, and thus must know about a wider range of grammatical forms. To use an existing generation grammar for parsing of real text thus requires extension of the grammatical resources.

For discussion on bi-directional processing, see also Appelt (1987), Jacobs (1988), and Block (1987).

3.2 Declarativisation

A bi-directional system requires declarative representation of the resources, since procedurally-implemented resources are hard-wired for a particular process. Declaratively

represented resources can be applied to different processing needs. The WAG parser, reported in chapter 10, relies heavily on internal re-representation of the resources, internally transforming the lexico-grammar into a form more suitable for analysis.

Declarative representation also allows the processes to be modified without modifying the entire resource (e.g., to switch from a top-down parser to a bottom up parser with procedurally implemented resources means re-writing the entire resource).

3.3 Resource Orientation

Even when declaratively represented, resources tend to be expressed in a form which favours one processing direction over another. The standard network representation of Systemic grammar, for instance, favours generation over parsing. The choice of representation may also favour a particular control strategy. For instance, associating semantico-grammar mapping constraints with grammatical features, rather than with semantic features, favours a target-driven generation strategy, since the mapping resources are most easily accessed in terms of the grammar.

In designing the representation scheme for resources in bi-directional system, one needs to take this factor into account, compromising between the needs of each process, and in regards to the types of control strategies which are desired.

3.4 Single Formalism for all levels

If we use the same formalism for all strata, then we can re-use the same processes (e.g., unification, forward-chaining, etc.) on each of the strata. It is possible, for example, to use the same process to construct a semantic structure from a lexico-grammatical structure as is used to construct a lexico-grammatical structure from the semantic structure (although the orientation of the resources, discussed above, may work against this).

Resource representation in WAG is based on a common formalism, extending the system network and realisation notation out of the grammar, such that ideational, interactional, textual and graphological structures can all be represented in this formalism.[4] In work so far unpublished, I have represented context, exchange-structure, and generic structure within the WAG formalism, showing that the usefulness of the formalism is quite wide.

There are some types of representation, however, which are not easily modelled in terms of system-networks and systemic structures. Some textual meanings, for instance, have been modelled in terms of textual fields instead (see chapter 5). The WAG constraint language has, however, been extended to allow testing or assertion of membership in the various textual fields, e.g., to test whether a particular entity is in the *relevance* textual field.

4. Summary

This section has provided an overview of processing, showing how sentence processing can be broken up into a series of modules, each translating a source representation into a target representation. Various means of performing each translation were discussed (control strategies), and also various means of relating the modules (information flow architectures).

I then discussed several ways in which processing shapes the resources -- requiring the resources to be declarative and bi-directional, and to use a standard formalism throughout all linguistic levels.

---

[4] At present, graphological structures are not represented in the WAG formalism, but they could be.