2. Natural Language Processing (NLP)

This section provides a brief history of NLP, introduces some of the main problems involved in extracting meaning from human languages and examines the kind of activities performed by NLP systems.

2.1. Background

Natural language processing systems take strings of words (sentences) as their input and produce structured representations capturing the meaning of those strings as their output. The nature of this output depends heavily on the task at hand. A natural language understanding system serving as an interface to a database might accept questions in English which relate to the kind of data held by the database. In this case the *meaning* of the input (the output of the system) might be expressed in terms of structured SQL queries which can be directly submitted to the database.

The first use of computers to manipulate natural languages was in the 1950s with attempts to automate translation between Russian and English [Locke & Booth]. These systems were spectacularly unsuccessful requiring human Russian-English translators to pre-edit the Russian and post-edit the English. Based on World War II code breaking techniques, they took individual words in isolation and checked their definition in a dictionary. They were of little practical use. Popular tales about these systems cite many mis-translations including the phrase "*hydraulic ram*" translated as "*water goat*".

In the 1960s natural language processing systems started to examine sentence structure but often in an ad hoc manner. These systems were based on pattern matching and few derived representations of meaning. The most well known of these is Eliza [Weisenbaum] though this system was not the most impressive in terms of its ability to extract meaning from language.

Serious developments in natural language processing took place in the early & mid 1970s as systems started to use more general approaches and attempt to formally describe the rules of the language they worked with. LUNAR [Woods 1973] provided an English interface to a database holding details of moon rock samples. SHRDLU [Winograd] interfaced with a virtual robot in a world of blocks, accepting English commands to move the blocks around and answer questions about the state of the world. Since that time there has been parallel development of ideas and technologies that provide the basis for modern natural language processing systems. Research in computer linguistics has provided greater knowledge of grammar construction [Gazdar] and Artificial Intelligence researchers have produced more effective mechanisms for parsing natural languages and for representing meanings [Allen]. Natural language processing systems now build on a solid base of linguistic study and use highly developed semantic representations.

Recently (during the 1990s) natural language systems have either focused on specific, limited domains with some success or attempted to provide general purpose language understanding ability with less success. A major goal in contemporary language processing research is to produce systems which work with complete threads of discourse (with human like abilities) rather than only with isolated sentences [Russell & Norvig(a)]. Successes in this area are currently limited.

2.2. Problems

Two problems in particular make the processing of natural languages difficult and cause different techniques to be used than those associated with the construction of compilers etc for processing artificial languages. These problems are (i) the level of ambiguity that exists in natural languages and (ii) the complexity of semantic information contained in even simple sentences.

Typically language processors deal with large numbers of words, many of which have alternative uses, and large grammars which allow different phrase types to be formed from the same string of words. Language processors are made more complex because of the irregularity of language and the different kinds of ambiguity which can occur. The groups of sentences below are used as examples to illustrate different issues faced by language processors. Each group is briefly discussed in the following section (in keeping with convention, ill-formed sentences are marked with an asterix).

1. The old man the boats.

2. Cats play with string.
   * Cat play with string.

3. I saw the racing pigeons flying to Paris.
   I saw the Eiffel Tower flying to Paris.

4. The boy kicked the ball under the tree.
   The boy kicked the wall under the tree.

1. In the sentence "The old man the boats" problems, such as they are, exist because the word "old" can be legitimately used as a noun (meaning a collection of old people) as well as an adjective, and the word "man" can be used as a verb (meaning take charge of) as well as a noun. This causes ambiguity which must be resolved during syntax analysis. This is done by considering all possible syntactic arrangements for phrases and sub-phrases when necessary.

   The implication here is that any parsing mechanism must be able to explore various syntactic arrangements for phrases and be able to backtrack and rearrange them whenever necessary.

2. The problem with the second sentence in this group is that there is no numeric agreement between the subject and the verb (one is a singular form, the other plural). Grammars must be expressive enough to specify checks for such anomalies and also specify actions which should take place if they occur. Mechanisms to signal failure in processing such cases are useful. For example, when combining semantics for "colourless" and "green" in a phrase like "the colourless green car" a signal of failure marks a sub-phrase as ill-formed and prevents it being considered any further. In the case of problems with the numeric agreement between subject and verb it may be more appropriate to signal a warning. A warning marks a sub-phrase as potentially-flawed but does not reject it outright.

3. Assuming these sentences are taken in isolation so there is no previous dialog which introduces a racing pigeon named "the Eiffel Tower"...

   The sensible way to interpret the meaning of the second sentence is "While I was flying to Paris I saw the Eiffel Tower in its usual position - firmly rooted to the ground". What prevents the second sentence being restructured in the same way as the first is the inconsistency with objects like the Eiffel Tower and activities like flight. Sensible semantic rules must detect this inconsistency and perhaps halt progress on any sub-phrase which implies the Eiffel Tower is involved in a flying activity.

   A semantic rule in this case might reason as follows:
   (i) the set of objects capable of flight is {humans, birds, bats, aeroplanes}
   (ii) the Eiffel Tower is defined as a structural-object
   (iii) structural-objects are not in the set of objects capable of flight so signal "ill-formed semantics" and halt progress on this rule.

4. The implication in the first sentence is that the activity performed by the boy causes the ball to move to a position which is under the tree. The kicking activity has a meaning of "move, using the boy's foot as an instrument to cause that movement". In the second sentence the wall is assumed not to have changed position. The activity which took place was one of "strike, using the foot as an instrument". The apparent disambiguation in this case can take place if it is known that balls are mobile objects (and are often moved using a foot as an instrument) and walls are static objects.

The purpose of examining example sentences like the four above are to introduce the reader to some of the complexity that occurs within natural language statements. The following subsections describe the type of activities carried out by systems which process natural language and therefore have to deal with problems similar to those illustrated above.

2.3. Natural Language Processing - the tasks involved

A simplified view of Natural Language Processing emphasises four distinct stages [Fig 2.1]. In real systems these stages rarely all occur as separated, sequential processes. In the overview that follows it is assumed that syntactic analysis and semantic analysis will be dealt with by the same mechanism - the parser. The rest of this section examines processes shown in the diagram.

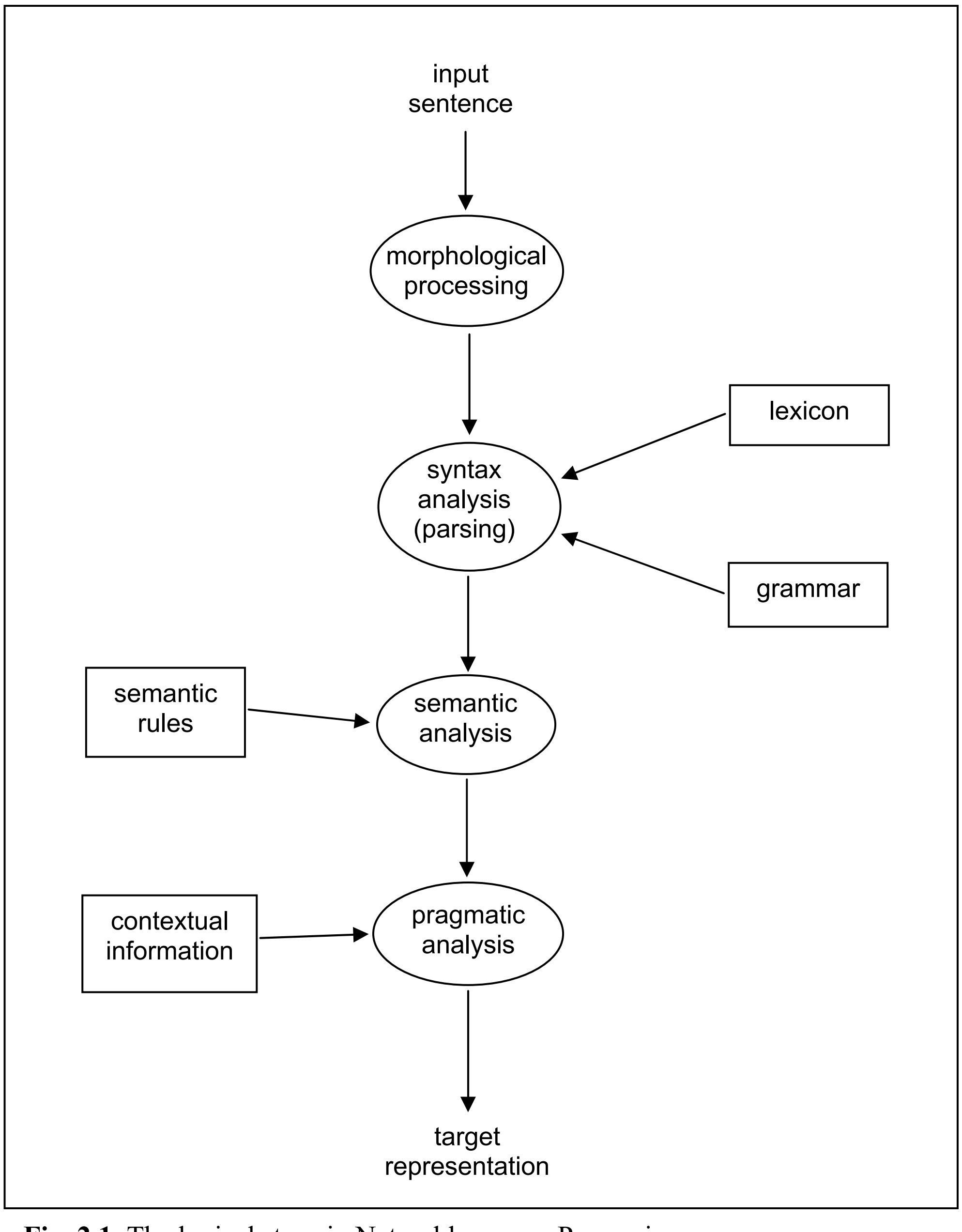


Fig. 2.1. The logical steps in Natural language Processing

2.3.1. Morphological Processing

The preliminary stage which takes place before syntax analysis is *morphological processing*.

The purpose of this stage of language processing is to break strings of language input into sets of tokens corresponding to discrete words, sub-words and punctuation forms. For example a word like "unhappily" can be broken into three sub-word tokens as:

un- happy -ly

Morphology is concerned primarily with recognising how base words have been modified to form other words with similar meanings but often with different syntactic categories. Modification typically occurs by the addition of prefixes and/or postfixes but other textual changes can also take place. In general there are three different cases of word form modification.

Inflection: textual representations of words change because of their syntactic roles. In English, for example, most plural nouns take -s as a suffix (and may require other modification), comparative and superlative forms of regular adjectives take *-er -est* suffixes.

Derivation: new words are derived from existing words. This manufacturing of words often occurs in a regular manner allowing following clear morphological rules. For example, in English some adjectives take -ness as a suffix when being used to create nouns (happy → happiness). The same principles apply in most human languages though the rules are different[1].

Compounding: new words are formed by grouping existing words. This occurs infrequently in English (examples include "headache" and "toothpaste") but is widely used in other languages where it is morphologically possible to have infinitely long words.

The nature of morphological processing is heavily dependent on the language being analysed. In some languages single words (used as verbs) contain all the information about the tense, person and number of a sentence. In other languages this information may be spread across many words. For example the English sentence "I will have been walking..." where complex tense information is only available by examining the structure of auxiliary verbs. Some languages attach prefixes/suffixes to nouns to indicate their roles (see figure 2.2) others use word inflections to provide proximity information (see figures 2.3 & 2.4).

Noun + Suffix	Syntactic case	Meaning
Chennai-ukku	dative: destination	To Madras
Chennai-ukku-irundu	dative: source	From Madras
Chennai-le	containment	In Madras
Chennai-ai	object (formal)	Madras

Fig. 2.2. Suffix Attachment for Noun Cases (Formal Tamil)
NB: the spellings used are the author's phonetic spellings.

[1] In Tamil new verbs can be created by using equivalent English verbs, in their infinitive form without "to", with a complex but regular suffix containing tense and person information. In Setswana nouns like "student" and "teacher" are derived from the verbs "study" and "teach" by using mo- as a prefix (notice "student" and "teacher" are also derived from "study" and "teach" in English, "teacher" is a regular derivation, "student" is irregular).

Proximity	**Time**	**Things (inanimate)**
Near	i-ppa (this time: *now*)	i-ndtha (this thing: *this*)
Far	a-ppa (that time: *then*)	a-ndtha (that thing: *that*)
Question	e-ppa (what time: *when*)	e-ndtha (what thing: *which*)

Fig. 2.3. Proximity Information as Prefix Tags (Tamil)

Proximity	**Cow**	**Student**
Near Speaker	kgomo e (this cow)	mo-ithuti yo (this student)
Near Listener	kgomo e-o (that cow)	mo-ithuti yo-o (that student)
Far	kgomo e-le (that cow)	mo-ithuti yo-le (that student)

Fig. 2.4. Proximity Information by Demostrative Pronoun Inflection (Setswana)

As a language, English is more easy to tokenise and apply morphological analysis to than many. In some far-eastern languages words are not separated (by whitespace characters) in their written form (examples include Japanese and some Chinese languages). In many languages the morphology of words can be ambiguous in ways that can only be resolved by carrying out syntactic and/or semantic analysis on the input. Simple examples in English occur between plural nouns and singular verbs: "climbs" as in '*there are many climbs in the Alps*' or '*he climbs Everest in March*'. This example of ambiguity can be resolved by syntax analysis alone but other examples are more complex. "Undoable" could be analysed as ((un-do) -able) or as (un- (do-able)), ambiguity which cannot always be resolved at the syntax level alone.

The output from the morphological processing phase is a string of tokens which can then be used for lexicon lookup. These tokens may contain tense, number, gender and proximity information (depending on the language) and in some cases may also contain additional syntactic information for the parser. The next stage of processing is syntax analysis.

2.3.2. Syntax and Semantics

A language processor must carry out a number of different functions primarily based around syntax analysis and semantic analysis. The purpose of syntax analysis is two-fold: to check that a string of words (a sentence) is well-formed and to break it up into a structure that shows the syntactic relationships between the different words. A syntactic analyser (or parser) does this using a dictionary of word definitions (the lexicon) and a set of syntax rules (the grammar). A simple lexicon only contains the syntactic category of each word, a simple grammar describes rules which indicate only how syntactic categories can be combined to form phrases of different types.

Examples of a simple lexicon and grammar could be:

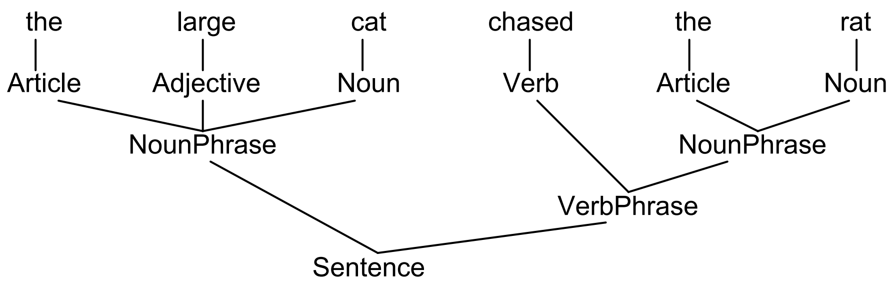
Lexicon

word	category
cat	Noun
chased	Verb
large	Adjective
rat	Noun
the	Article

Grammar

Sentence $\rightarrow$ NounPhrase, VerbPhrase[2]
VerbPhrase $\rightarrow$ Verb, NounPhrase
NounPhrase $\rightarrow$ Article, Noun
NounPhrase $\rightarrow$ Article, Adjective, Noun

This grammar-lexicon combination could destructure the sentence "*The large cat chased the rat*" as follows:



Often the task of a language processor is to analyse a sentence in a language like English and produce an expression in some formal notation which, as far as the computer system is concerned, concisely expresses the semantics of the sentence. An interface to a database might, for example, require a language processor to convert sentences in English or German into SQL queries. Semantic analysis is the term given to the production of this formalised semantic representation.

In order to carry out semantic analysis the lexicon must be expanded to include semantic definitions for each word it contains and the grammar must be extended to specify how the semantics of any phrase are formed from the semantics of its component parts. For example the grammar rule above *VerbPhrase $\rightarrow$ Verb, NounPhrase* states how the syntactic group called *VerbPhrase* is formed from other syntactic groups but says nothing about the semantics of any resulting *VerbPhrase*. Using a simplified form of logic the grammar and lexicon can be expanded to capture some semantic information. This is illustrated in the following example.

[2] This notation is used to describe the fact that a *Sentence* composed of a *NounPhrase* followed by a *VerbPhrase*.

Lexicon

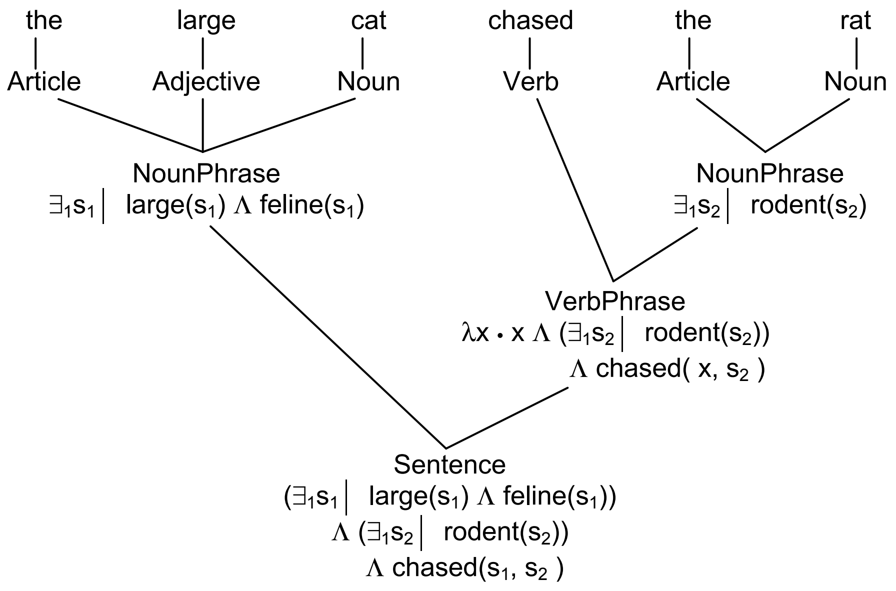
word	category	semantics
cat	Noun	$\lambda x \cdot feline(x)$
chased	Verb	$\lambda xy \cdot x \Lambda y \Lambda chased(x, y)$
large	Adjective	$\lambda x \cdot largesize(x)$
rat	Noun	$\lambda x \cdot rodent(x)$
the	Article	$\exists_1 \langle gensym \rangle$

NB: terms like "feline" and "chased" are used in this example as primitive semantic relations.

Grammar

Syntactic rule	Semantic rule
Sentence → NounPhrase, VerbPhrase	apply VerbPhrase (NP)
VerbPhrase → Verb, NounPhrase	apply Verb (NounPhrase)
NounPhrase → Article, Noun	apply Noun (Article)
NounPhrase → Article, Adjective, NounPhrase	apply Adjective (Article) Λ apply Noun (Article)

NB: the above is simplified for readability, apply is used to provide an argument to a λ form so: apply $\lambda x \cdot rodent(x)(Ralf) \rightarrow rodent(Ralf)$


the
large
cat
chased
the
rat
Article
Adjective
Noun
Verb
Article
Noun
NounPhrase
$\exists_1 s_1 | large(s_1) \Lambda feline(s_1)$
NounPhrase
$\exists_1 s_2 | rodent(s_2)$
VerbPhrase
$\lambda x \cdot x \Lambda (\exists_1 s_2 | rodent(s_2)) \Lambda chased(x, s_2)$
Sentence
$(\exists_1 s_1 | large(s_1) \Lambda feline(s_1)) \Lambda (\exists_1 s_2 | rodent(s_2)) \Lambda chased(s_1, s_2)$

Syntactic and semantic structures produced on analysing a simple sentence.

The example above demonstrates how grammar rules are used to specify the method for producing semantic forms. The rule describing the legal syntactic form for a *VerbPhrase*, for example, also describes how to create semantics for verb phrases. In this way semantics are formed and grouped into larger sub-phrases until, after applying all of the relevant rules, a semantic expression describing the whole sentence is produced.

2.3.3. Semantics and Pragmatics

After semantic analysis the next stage of processing deals with pragmatics. Unfortunately there is no universally agreed distinction between semantics and pragmatics. This document, in common with several other authors [Russel & Norvig(c)] makes the distinction as follows: semantic analysis associates meaning with isolated utterances/sentences; pragmatic analysis interprets the results of semantic analysis from the perspective of a specific context (the context of the dialogue or state of the world etc). This means that with a sentence like "*The large cat chased the rat*" semantic analysis can produce an expression which means *the large cat* but cannot carry out the further step of inference required to identify the large cat as Felix. This would be left up to pragmatic analysis. In some cases, like the example just described, pragmatic analysis simply fits actual objects/events which exist in a given context with object references obtained during semantic analysis. In other cases pragmatic analysis can disambiguate sentences which cannot be fully disambiguated during the syntax and semantic analysis phases. As an example consider the sentence "*Put the apple in the basket on the shelf*". There are two semantic interpretations for this sentence. Using a form of logic for the semantics:

1. put the apple which is currently in the basket onto the shelf

$$(\exists_1 a : apple \mid \exists b : basket \land inside(a, b)) \land \exists_1 s : shelf \Rightarrow puton(a, s)$$

2. put the apple into the basket which is currently on the shelf

$$\exists_1 a : apple \land (\exists_1 b : basket \mid \exists_1 s : shelf \land on(b, s)) \Rightarrow putin(a, b)$$

Pragmatic analysis, in consulting the current context, could choose between the two possibilities above based on the states and positions of objects in the world of discourse.

2.4. Section Summary

This section has provided examples of some of the problems associated with analysing human languages and has described the most important stages in Natural Language Processing. The next section examines the different approaches that can be used to specify grammars and lexicons. The specification of a grammar and lexicon for even a small subset of a natural language is a non-trivial activity. The correctness/integrity of the grammar and lexicon are of great importance since their use underpins all syntactic and semantic analysis.

3. Specifying Grammars for Natural Languages

The single most important consideration for the design of Lkit concerned the types of grammar the toolkit would accept. This section gives a brief overview of the various types of grammar most commonly used to specify natural languages for computer-based analysis.

All grammars perform similar tasks, they specify a set of rules which work together to define legal sentences in their language. Context free grammars (like BNF) are the easiest for people to write and are easy to deal with computationally. However natural human languages are not context free. Most grammar formalisms draw a compromise by writing rules in a context free style then augmenting these rules to deal with the kind of complications discussed earlier (numeric agreement, handling semantics etc).

There are a variety of different approaches used for grammar construction and specification but it is useful to examine the use of any grammar in two ways:
1. as a conceptual design tool - a strategy for capturing and describing the complexity of a natural language;
2. as a formal notation for describing the syntactic (and possibly some semantic) characteristics of a language in a way that can be used by a parser.

The first of these categories is concerned primarily with the theory and structure of languages, the second is concerned with the description and implementation of grammars. The relationship between the conceptual design of a grammar and the notation used to specify/code it is similar to the relationship between the conceptual design of a program and the programming language used to write it. In this analogy the parser is similar to an interpreter or compiler that recognises the programming language

Unfortunately it is impossible to completely separate the theoretical/conceptual design of a grammar from issues of implementation. Different conceptual designs of grammar place different requirements on the notations that may be used to code them. Additionally different notations have different representational capabilities and may therefore restrict conceptual designs they can describe. Case Grammars, for example, are mainly concerned with the theoretical structure of a language and can be implemented by different parsers using different notations for grammar rules. However Case Grammars influence the choice of notation used to describe them since some grammatical notations cannot adequately represent them. Conversely notations based on Context Free Grammars (like Recursive Transition Networks) impose severe constraints on the conceptual design of grammars which they describe.

The rest of this section describes various types of grammar indicating, where appropriate, their requirements and/or restrictions.

3.1. Case Grammars

Case Grammars [Fillmore] attempt to describe any given sentence in terms of a fixed frame of slots (called cases) which explicitly capture information about any activities described in the sentence, the instigators of those activities, positions, times, etc. Though there is no universally agreed set of cases or their names a common subset is outlined below.

Case	Meaning
Action	the action which takes place (usually related to the main verb of a sentence)
Actor	the instigator of the action (often an animate entity which/who *does* the action)
Object	the entity which is acted upon or is changed by the action
Source	the starting position for an entity (ie: its position before the action takes place)
Destination	the resulting position for an entity (after the action has completed)
Location	the location for the action
Instrument	an object/entity used in order to cause the event (eg: *key* in "*He unlocked the door with a key*"
Time	the time/date of the action

For example case analysis of the sentence "*Gary repaired the car in the garage on Sunday*" could generate the following case frame:

Action	repairs
Actor	Gary
Object	car
Location	garage
Time	Sunday

With Case Grammars rules describe syntactic constraints but also describe manipulations geared towards producing case frames (sets of case slots). These may be flat (as in the example above) or nested hierarchical structures which can form the basis of semantic representations.

To implement a Case Grammar the notation used to express rules must allow them to produce case frames (ie: structures without a pre-determined form). Grammar notations whose only output reflects the syntactic structure of their input are not suitable since they cannot be used to construct case frames.

3.2. Semantic Grammars

Semantic Grammars (also called Domain Specific Grammars) [Rich & Knight] place no restrictions on the notations used to describe them but use rules built around domain specific phrase types rather than typical abstract categories. For example, in a medical domain the sentence "*there is a bleeding ulcer in the stomach towards the lesser curve*" the two phrases "*in the stomach*" and "*towards the lesser curve*" might be classified by a semantic grammar as a locator-phrase and a location-qualifier respectively. In a more general purpose grammar for English these two phrases would be of the same type: prepositional-phrase.

The motivation for doing this is that, in limited domains, grammars can be developed faster by people who are familiar with the domain but are not necessarily expert linguists. The problems of grammar construction are reduced by allowing it to focus on the domain and ignore features of language which are not used. Additionally, domain experts are more able to help describe (and debug) suitable grammar rules when the terminology is more natural to them.

Unfortunately it is generally recognised that semantic grammars are more difficult to extend in order to capture syntactic generalisations or to deal with increasingly complex/varied language as the domain expands. In short domain-specific/semantic grammars do not scale up well.

Semantic grammars can be constructed using various different grammar notations but typically the result of applying a semantic grammars reflects the semantic structure of its input rather than its syntactic structure. To achieve this semantic grammars associate semantic actions with each of their grammar rules, any grammar notation used with semantic grammars must support this.

3.3. Definitive Clause Grammars (DCGs)

Definitive Clause Grammars [Pereira & Warren] use rules which are based on logical implication expressions. These expressions are equivalent to Prolog clauses with the result that a Prolog interpreter can be used as a parser for DCGs. An unadapted Prolog interpreter would perform a top-down, depth first parse which is not particularly efficient.

Rules in DCGs are Horn clauses which may have multiple antecedents but only a single consequent. Non-terminal symbols in the grammar act as predicates, for example:

$$\text{NounPhrase(NP)} \wedge \text{VerbPhrase(VP)} \Rightarrow \text{Sentence(append(NP, VP))}$$

In practice DCG rules are written like rules in other grammars (rather than as logical expressions) so the rule above could be written:

$$\text{Sentence} \rightarrow \text{NounPhrase VerbPhrase.}$$

Rules are augmented to allow context sensitive checks to be specified, for example to ensure numeric agreement between subject and verb thereby accepting sentences like "Cats_{plur} play_{plur} with string" while rejecting others like "Cat_{sing} play_{plur} with string". Typically such augmentations might be written:

$$\text{Sentence} \rightarrow \text{NounPhrase(num (n)) VerbPhrase(num (n))}$$

Additional augmentation allows semantics to be assembled for the target category of a rule

```
Sentence(semantics append(s1,s2)) →
        NounPhrase(semantics  (s1)) VerbPhrase(semantics ( s2))
```

The appearance of such rules becomes more complex when different augmentations are combined, ie:

```
Sentence (semantics append(s1,s2)) →
        NounPhrase(num (n), semantics  (s1))
        VerbPhrase(num (n), semantics  (s2))
```

Rules are complicated further when transformations & checks are specified for semantic structures. DCGs are popular because of their declarative style but are often associated with inefficient parsing strategies.

3.4. Lexical Functional Grammars (LFGs)

Lexical grammars present a different approach to grammar/lexicon construction by removing rules from the grammar and embedding them instead in the lexicon. The justification for this is that the valence of words (the number and type of other syntactic groups they associate with) is a feature associated with words rather than their meanings or syntactic classification. The valence of a word defines the legal structure of sentences it may occur in. For example, the verbs "dined", "ate" and "devoured" are all part-tense verb forms and all have similar meanings but (because of their transitivity) impose different restrictions on sentences they can be used in. The following sentences illustrate the point:

a. The guests devoured the meal.
b. * The guests devoured.
c. * The guests dined the meal.
d. The guests dined.
e. The guests ate the meal.
f. The guests ate.

A lexical rule for "ate" would contain the following kind of information:

ate	specifiers	$NP_{type\ =\ animate,\ participation\ =\ mandatory}$
	complements	$NP_{type\ =\ food\text{-}stuff,\ participation\ =\ optional}$

This rule states that the word "ate" is mandatorily specified (prefixed) by a noun phrase describing something animate and optionally complemented (followed) by a noun phrase describing something which is a food-stuff.

Because lexical grammars have rule like constructions occurring inside the lexicon their use imposes different sets of requirements on any syntax analyser/parser that implements them. This is also true because the structure of lexical rules tends to be different to the structure of rules used with other grammars, DCGs for example.

Proponents of LFGs cite their elegance in handling some complexities of language that other grammars struggle to represent and the way that specifier/complement notation can be uniformly applied to all parts of speech. Others claim that LFGs are less suitable for fast prototyping small grammars aimed at restricted language domains and that lexicons tend to contain even more duplication than usual.

As with many areas of linguistics the choice between an LFG approach and others is largely down to personal preference. Consider the use of prepositions with nouns [Murphy]. Different nouns take different pronouns, eg:

...demand for...
...cause of...
...increase in...
...solution to...

LFGs would handle this by embedding notions of legal pronoun use inside rules attached to the lexical entries for the nouns concerned. Other formalisms would attach type information to the lexical definitions of pronouns and sets of acceptable pronoun types to definitions of nouns. General purpose grammar rules would then be responsible for checking for legal noun-pronoun type correspondence.

3.5. Augmented Transition Networks (ATNs)

ATNs [Woods 1970] are usually described as a highly developed and extended form of State Transition Network (STN). A basic STN for natural language would have arcs labelled as some terminal syntactic category. Recursive Transition Networks (RTNs) are a development of STNs. They consist of collections of small STNs with arcs which can be labelled with names of other networks (non-terminals). ATNs are a further development of RTNs which allow arcs to be labelled with tests and actions which may bind or reference ATN variables. Fig 2.4 shows a simple ATN.

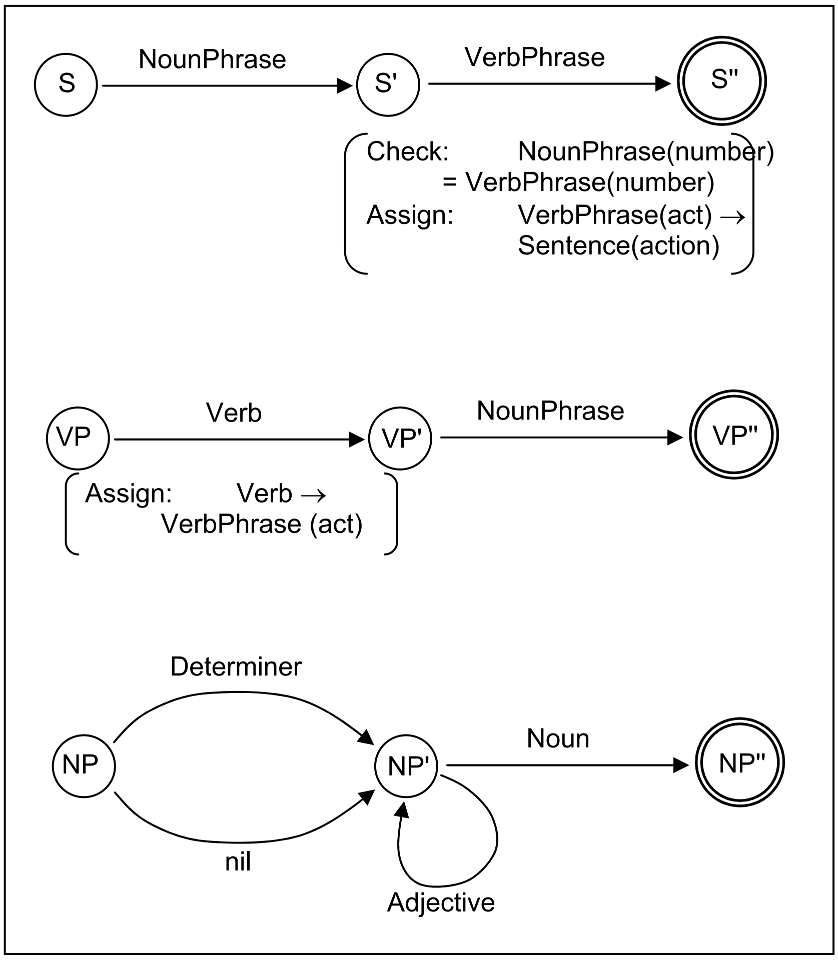

S
NounPhrase
S'
VerbPhrase
S"
Check: NounPhrase(number) = VerbPhrase(number)
Assign: VerbPhrase(act) → Sentence(action)
VP
Verb
VP'
NounPhrase
VP"
Assign: Verb → VerbPhrase (act)
Determiner
NP
NP'
Noun
NP"
nil
Adjective

Fig 2.4. An example of a simple ATN.

ATNs are used by parsers which are flexible, typically top-down and can build structures of arbitrary complexity. They were probably the most widely used type of grammar in Natural Language Processing after the success of LUNAR [Woods 1973] until the mid 1980s. They lend themselves to the construction of powerful grammars which do not have to be described in a network-like formalism but none-the-less tend to be more procedural in nature than some other grammars. Since the 1980s ATN use has declined, this has been in part due to some researchers switching to DCGs which appear declarative in style and use augmentations based on unification rather than assignment (as in ATNs). Others researchers have moved away from ATN grammars in order to switch to parsers which can employ a greater variety of search strategy, in these cases there is often some similarity between the grammars they use and ATN grammars.

3.6. Summary

This section has presented an overview of the most regularly cited types of grammar used in Natural Language Processing (a branch of Computational Linguistics). This list is by no means exhaustive and different approaches are preferred in Pure Linguistics to those commonly used in Computational Linguistics. Research effort in Pure Linguistics is focused in a range of areas (some of which overlap with work in Computational Linguistics), including:

- finding grammatical generalisations which are applicable in all human languages and grammar formalisms that are language independent [Haegeman];
- formally specifying grammars for languages which have not previously been analysed[3];
- examining how old languages have evolved into modern languages.

The next section examines the design and use of semantic representations and investigates their use in building grammars and lexicons. A later section describes the type of notations used by Lkit to define grammars and lexicons which can describe both syntactic and semantic rules of a language.

[3] A current research project aims to describe a grammar for the language of the San people of the Kalahari. These are hunter-gatherer people whose culture had until recently remained largely unaffected by the main-stream cultures active in their geographical region. They are one of the few remaining groups of known peoples whose language does not exist in any written form. Specifying a written form for their language is the goal of a larger piece of research.

4. Semantics

The goal of most NLP[4] systems is to extract meaning from their language input. This *meaning* might ultimately be expressed as SQL for instance if the interface is for a database but in order to generate target representations NLP systems must first create intermediate representations to capture and refine meanings from their input. This intermediate representation that captures *meaning* is the semantic representation of the system.

In general, semantic representations need to capture details of objects and their relationships, events and the chains of causality that tie them together. A detailed discussion of semantic representations is beyond the scope of this document but the following section intends to highlight the issues of particular importance to semantic representations used specifically for NLP.

4.1. Forms of Semantic Representation

Any semantic representation can be viewed on three levels:

1. The conceptual level: what is represented and why. This level describes the representational capabilities. A simple representation may only capture details of objects and object-object relationships. A more sophisticated representation may capture details of events, their timings, possible outcomes and inter-event dependencies.

2. The abstract form: most representations can be described in abstract forms, using diagrams for example. These diagrams show what the representation makes explicit and what must be inferred. The abstract form shows the structure of the representation and the primitives used to build it. Abstract forms are not deterministically derived from the conceptual form. Even very simple concepts can be expressed by a variety of different abstract forms. Fig 4.1 & 4.2 show examples of this.

3. The physical realisation of the representation: the implementation form, data structures and inference mechanisms.

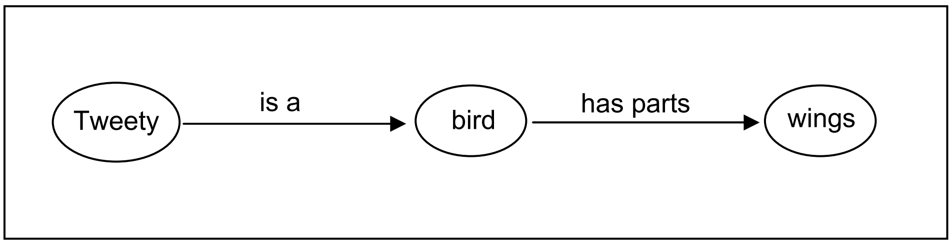


Fig. 4.1.. An abstract form capturing the relationship between *Tweetie* and *wings*

[4] Natural Language Processing

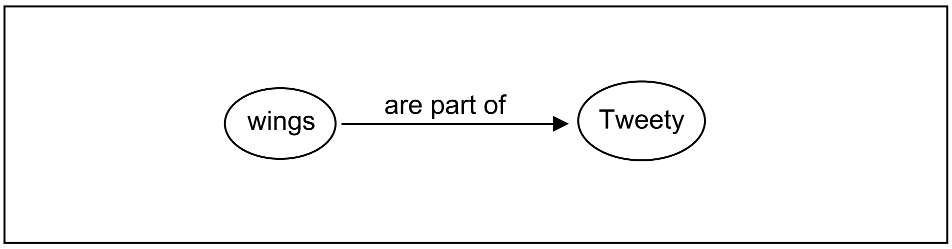


Fig. 4.2. An alternative abstract form capturing the relationship between *Tweetie* and *wings*

The choice of form for level 3 has impact on the capabilities of the abstract representation. A poor choice can limit its expressive power or make it clumsy to use. Some representations use simple logic for their implementation form. This has some advantages: it is well understood, it has precise, unambiguous semantics and it supports inference. However once this choice has been made it imposes restrictions at level 1 & 2 (it is difficult to deal with degrees of truth and the influence of time etc).

A major design effort in any Natural Language Processing system is the semantic representation. Pioneering work into suitable representations took place in the 1970s with Wilks's investigation into semantic primitives for his translation system [Wilks] and Schank's Conceptual Dependency Theory [Schank]. Wilks classified objects according to their semantic categories for example *PhysicalObjects*, *AnimateObjects* and *ManiplulableObjects* (those that could move or be moved). These classifications were used to guide his system in forming unambiguous descriptions of the language input. For example given the sentences:

1. The boy kicked the ball under the tree.
2. The boy kicked the wall under the tree.

The classification of "ball" as a *ManipulableObject* would cause the system to recognise that the activity described by sentence (1) was one of movement, while the classification of "wall" as a *StaticObject* in sentence (2) implied a *striking* activity.

Schank's Conceptual Dependency Theory reduced all verb forms to specialisations of a small number of primitive actions (eleven in the original work) which described activities like moving a physical object, applying a force to an object, ingesting foodstuff, etc. Each primitive action had a clearly defined set of legal uses, known preconditions and known results. For example an *Actor* could only move an *Object* if they could manipulate that object and they were in the same location. The result of moving the *Object* would be that both *Actor* and *Object* were at a new location.

In addition the representation described objects by a set of object states which took numeric values. For example the state *Health* had a range between +10 (perfect health) and -10 (dead). The physical condition of an object had a range from -10 (broken beyond repair), through -5 (damaged) to 0. The most interesting of these states relate to the mental & emotional states of humans and other intelligent entities, the state of *Joy* had a range from -10 to +10 with -5 indicating depression and +5 indicating happiness.

State values were associated with adjectives and adjectival phrases and were also generated by rules of inference. A rule of inference might state for example, the conditions under which a human could suffer a deterioration in their *Joy* value.

As well as specifying sets of primitive acts and states, Conceptual Dependency defined how these primitives could be combined to describe more complex concepts. In addition to other relationships, Conceptual Dependency explicitly represented causal relationships between one concept and another. A sentence like "*The boy ate a bad apple, it made him sick*" would map onto a form with two separate concepts: *eating-a-bad-apple* and *being-sick* with the second of these being (unintentionally but unavoidably) caused by the first.

The work of both Schank and Wilks, while dated in Artificial Intelligence research terms, introduced many of the ideas still considered significant when developing semantic representations. Their work identified the importance of developing useable semantic primitives and mechanisms for grouping these primitives into higher level forms. Designers of lexicons and grammars must consider both of these issues: the primitives and how they are grouped.

4.2. Building Semantic Representations

Semantic representations are built from semantic fragments attached to words in the lexicon. Semantic rules contained within grammar rules describe how these fragments are combined to form the semantics for larger phrases. For example if the target representation for a sentence like "*The boy eats an apple*" is some thing like INGESTS(young-male-human, fruit) and the relevant lexical entries are somewhat trivially:

eats	INGEST
boy	young-male-human
apple	fruit

The (unlikely) grammar rule below would produce the required semantics:

```
Sentence → NounPhrase1 Verb NounPhrase2
Semantics:
      Verb_semantics( NounPhrase1_semantics, NounPhrase2_semantics )
```

This approach to deriving semantics by combining the action of a number of rules is known as "combinatorial semantics". Language processors can implement it two ways:

1. apply the relevant semantic processing each time a new phrase is formed;
2. apply semantic processing only when a complete and satisfactory parse has been found for an input sentence.

Each method has its own advantages. Strategy 1 has an obvious processing overhead but can generate useful information during the parse which helps to prevent exploration of semantically ill-formed phrases.

4.3. Summary

In order for a language processor to do more than check that its input is syntactically valid or restructure its input solely on the basis of its syntactic composition it must be capable of performing some analysis of the *meaning* of its input. This is the purpose of semantic analysis. Semantic descriptions of words are included in the lexicon and rules describing the ways these semantics can be combined and reshaped are included in the grammar.

Proper use of semantics not only results in a representation of meaning but may also be used to arrive at the correct interpretation for certain sentences. The example provided in the preceding section used semantic categories to interpret the sentence: "*the boy kicked the wall under the tree*".