Chapter 3

Representation and Processing

3.1 Introduction

In this chapter we will introduce some of the techniques that can be used to represent the kind of information that is needed for translation in such a way that it can be processed automatically. This will provide some necessary background for Chapter 4, where we describe how MT systems actually work.

Human Translators actually deploy at least five distinct kinds of knowledge:

- Knowledge of the source language.
- Knowledge of the target language. This allows them to produce texts that are acceptable in the target language.
- Knowledge of various correspondences between source language and target language (at the simplest level, this is knowledge of how individual words can be translated).
- Knowledge of the subject matter, including ordinary general knowledge and 'common sense'. This, along with knowledge of the source language, allows them to understand what the text to be translated means.
- Knowledge of the culture, social conventions, customs, and expectations, etc. of the speakers of the source and target languages.

This last kind of knowledge is what allows translators to act as genuine mediators, ensuring that the target text genuinely communicates the same sort of message, and has the same sort of impact on the reader, as the source text.[1] Since no one has the remotest idea how

[1] Hatim and Mason Hatim and Mason (1990) give a number of very good examples where translation requires this sort of cultural mediation.

to represent or manipulate this sort of knowledge, we will not pursue it here — except to note that it is the lack of this sort of knowledge that makes us think that the proper role of MT is the production of draft or 'literal' translations.

Knowledge of the target language is important because without it, what a human or automatic translator produces will be ungrammatical, or otherwise unacceptable. Knowledge of the source language is important because the first task of the human translator is to figure out what the words of the source text mean (without knowing what they mean it is not generally possible to find their equivalent in the target language).

It is usual to distinguish several kinds of linguistic knowledge:

- Phonological knowledge: knowledge about the sound system of a language, knowledge which, for example, allows one to work out the likely pronunciation of novel words. When dealing with written texts, such knowledge is not particularly useful. However, there is related knowledge about **orthography** which can be useful. Knowledge about spelling is an obvious example.
- Morphological knowledge: knowledge about how words can be constructed: that *printer* is made up of *print* + *er*.
- Syntactic knowledge: knowledge about how sentences, and other sorts of phrases can be made up out of words.
- Semantic knowledge: knowledge about what words and phrases mean, about how the meaning of a phrase is related to the meaning of its component words.

Some of this knowledge is knowledge about individual words, and is represented in dictionaries. For example, the fact that the word *print* is spelled the way it is, that it is not made up of other words, that it is a verb, that it has a meaning related to that of the verb *write*, and so on. This, along with issues relating to the nature and use of morphological knowledge, will be discussed in Chapter 5.

However, some of the knowledge is about whole classes or **categories** of word. In this chapter, we will focus on this sort of knowledge about syntax and semantics. Sections 3.2.1, and 3.2.2 discuss syntax, issues relating to semantics are considered in Section 3.2.3. We will look first on how syntactic knowledge of the source and target languages can be expressed so that a machine can use it. In the second part of the chapter, we will look at how this knowledge can be used in automatic processing of human language.

3.2 Representing Linguistic Knowledge

In general, syntax is concerned with two slightly different sorts of analysis of sentences. The first is **constituent** or **phrase structure** analysis — the division of sentences into their constituent parts and the categorization of these parts as nominal, verbal, and so on. The second is to do with **grammatical relations**; the assignment of grammatical relations such as SUBJECT, OBJECT, HEAD and so on to various parts of the sentence. We will discuss these in turn.

3.2.1 Grammars and Constituent Structure

Sentences are made up of words, traditionally categorised into **parts of speech** or **categories** including nouns, verbs, adjectives, adverbs and prepositions (normally abbreviated to N, V, A, ADV, and P). A **grammar** of a language is a set of rules which says how these parts of speech can be put together to make grammatical, or 'well-formed' sentences.

For English, these rules should indicate that (1a) is grammatical, but that (1b) is not (we indicate this by marking it with a '*').

(1) a. Put some paper in the printer.
b. *Printer some put the in paper.

Here are some simple rules for English grammar, with examples. A **sentence** consists of a **noun phrase**, such as *the user* followed by a **modal** or an **auxiliary verb**, such as *should*, followed by a **verb phrase**, such as *clean the printer*:

(2) The user should clean the printer.

A **noun phrase** can consist of a **determiner**, or **article**, such as *the*, or *a*, and a **noun**, such as *printer* (3a). In some circumstances, the determiner can be omitted (3b).

(3) a. the printer
b. printers

'Sentence', is often abbreviated to S, 'noun phrase' to NP, 'verb phrase' to VP, 'auxiliary' to AUX, and 'determiner' to DET. This information is easily visualized by means of a labelled bracketing of a string of words, as follows, or as a **tree diagram**, as in Figure 3.1.

(4) a. Users should clean the printer.
b. $[_S\ [_{NP}\ [_N$ users $]][_{AUX}$ should $][_{VP}\ [_V$ clean $][_{NP}\ [_{DET}$ the $][_N$ printer $]]]]$

The auxiliary verb is optional, as can be seen from (5), and the verb phrase can consist of just a verb (such as *stopped*):

(5) a. The printer should stop.
b. The printer stopped.

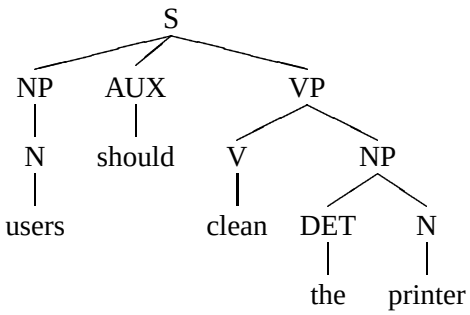

Figure 3.1 A Tree Structure for a Simple Sentence

NP and VP can contain **prepositional phrases** (PPs), made up of **prepositions** (*on, in, with*, etc.) and NPs:

(6) a. The printer stops on occasions.
b. Put the cover on the printer.
c. Clean the printer with a cloth.

The reader may recall that traditional grammar distinguishes between **phrases** and **clauses**. The phrases in the examples above are parts of the sentence which cannot be used by themselves to form independent sentences. Taking *The printer stopped*, neither its NP nor its VP can be used as independent sentences:

(7) a. *The printer
b. *Stopped

By contrast, many types of clause can stand as independent sentences. For example, (8a) is a sentence which consists of a single clause — *The printer stopped*. As the bracketing indicates, (8b) consists of two clauses co-ordinated by *and*. The sentence (8c) also consists of two clauses, one (*that the printer stops*) embedded in the other, as a sentential **complement** of the verb.

(8) a. [$_S$ The printer stopped]
b. [$_S$ [$_S$ The printer stopped]and [$_S$ the warning light went on]].
c. [$_S$ You will observe [$_S$ that the printer stops]].

There is a wide range of criteria that linguists use for deciding whether something is a phrase, and if it is, what sort of phrase it is, what category it belongs to. As regards the first issue, the leading idea is that phrases consist of classes of words which normally group together. If we consider example (2) again (*The user should clean the printer*), one can see that there are good reasons for grouping *the* and *user* together as a phrase, rather than grouping *user* and *should*. The point is *the* and *user* can be found together in many other contexts, while *user* and *should* cannot.

(9) a. A full set of instructions are supplied to the user.
b. The user must clean the printer with care.
c. It is the user who is responsible for day-to-day maintenance.
d. *User should clean the printer.

As regards what category a phrase like *the user* belongs to, one can observe that it contains a noun as its 'chief' element (one can omit the determiner more easily than the noun), and the positions it occurs in are also the positions where one gets proper nouns (e.g. names such as *Sam*). This is not to say that questions about constituency and category are all clear cut. For example, we have supposed that auxiliary verbs are part of the sentence, but not part of the VP. One could easily find arguments to show that this is wrong, and that *should clean the printer* should be a VP, just like *clean the printer*, giving a structure like the following, and Figure 3.2:

(10) $[_S\ [_{NP}\ [_N$ users $]][_{VP}\ [_{AUX}$ should $][_V$ clean $][_{NP}\ [_{DET}$ the $][_N$ printer $]]]]$

Moreover, from a practical point of view, making the right assumptions about constituency can be important, since making wrong ones can lead to having to write grammars that are much more complex than otherwise. For example, suppose that we decided that determiners and nouns did not, in fact, form constituents. Instead of being able to say that a sentence is an NP followed by an auxiliary, followed by a VP, we would have to say that it was a determiner followed by an noun, followed by a VP. This may not seem like much, but notice that we would have to complicate the rules we gave for VP and for PP in the same way. Not only this, but our rule for NP is rather simplified, since we have not allowed for adjectives before the noun, or PPs after the noun. So everywhere we could have written 'NP', we would have to write something very much longer. In practice, we would quickly see that our grammar was unnecessarily complex, and simplify it by introducing something like an NP constituent.

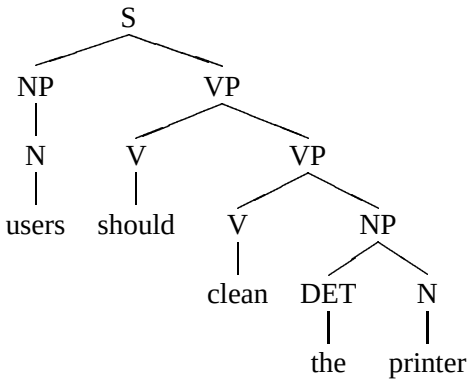

Figure 3.2 An Alternative Analysis

For convenience linguists often use a special notation to write out grammar rules. In this notation, a rule consists of a 'left-hand-side' (LHS) and a 'right-hand-side' (RHS) con-

nected by an arrow (→):

```
S → NP (AUX) VP

VP → V (NP) PP*

NP → (DET) (ADJ) N PP*

PP → P NP

N → user

N → users

N → printer

N → printers

V → clean

V → cleans

AUX → should

DET → the

DET → a

P → with
```

The first rule says that a **S**entence can be rewritten as (or decomposes into, or consists of) an NP followed by an optional AUX, followed by VP (optionality is indicated by brackets). Another rule says that a PP can consist of a P and an NP. Looked at the other way, the first rule can be interpreted as saying that an NP, and AUX and a VP make up a sentence. Items marked with a star ('*') can appear any number of times (including zero) — so the second rule allows there to be any number of PPs in a VP. The rules with 'real words' like *user* on their RHS serve as a sort of primitive dictionary. Thus the first one says that *user* is a noun, the fifth one that *clean* is a verb. Since the NP rule says that an N by itself can make up an NP, we can also infer that *printers* is an NP, and since (by the VP rule) a V and an NP make up a VP, *clean printers* is a VP. Thus, a grammar such as this gives information about what the constituents of a sentence are, and what categories they belong to, in the same way as our informal rules at the start of the section.

Returning to the tree representation in Figure 3.1, each node in the tree (and each bracketed part of the string representation) corresponds to the LHS of a particular rule, while the daughters of each node correspond to the RHS of that rule. If the RHS has two constituents, as in `NP → DET N`, there will be two branches and two daughters; if there are three constitituents, there will be three branches and three daughters, and so on.

It is worthwhile to have some terminology for talking about trees. Looking from the top,[2] the trees above start from (or 'are rooted in') a sentence node — the LHS of our sentence rule. Near the bottom of the trees, we have a series of nodes corresponding to the LHS's of dictionary rules and, immediately below them at the very bottom of the trees, actual words from the corresponding RHS's of the dictionary rules. These are called the 'leaves' or terminal nodes of the tree. It is normal to speak of 'mother' nodes and 'daughter' nodes (e.g. the S node is the mother of the NP, AUX, and VP nodes), and of mothers 'dominating' daughters. In practice most sentences are longer and more complicated than our example. If we add adjectives and prepositional phrases, and some more words, more complex trees can be produced, as shown in Figure 3.3, where the NP which is the left daughter of the S node contains an adjective and a noun but no determiner (the NP rule in our grammar above allows for noun phrases of this form), the NP in VP contains a determiner and a PP.

A large collection of such rules will constitute a formal grammar for a language — formal, because it attempts to give a mathematically precise account of what it is for a sentence to be grammatical. As well as being more concise than the informal descriptions at the beginning of the section, the precision of formal grammars is an advantage when it comes to providing computational treatments.

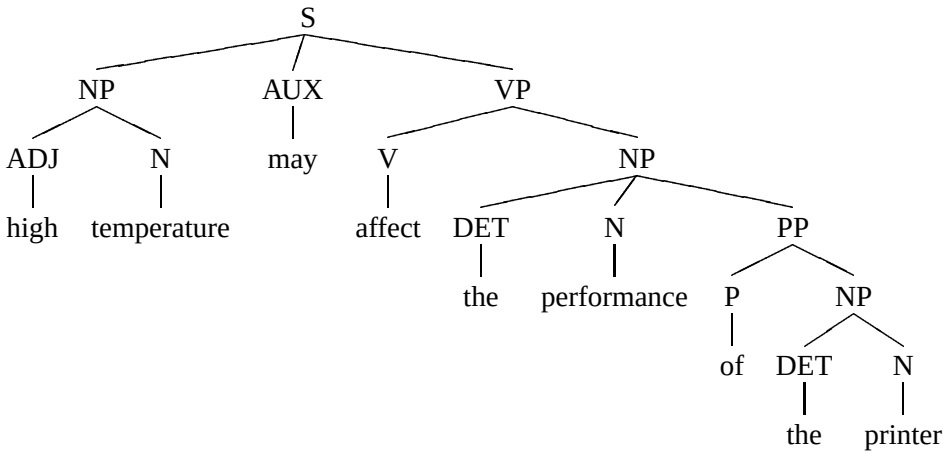

Figure 3.3 A More Complex Tree Structure

We should emphasise that the little grammar we have given is not the *only* possible grammar for the fragment of English it is supposed to describe. The question of which grammar is 'best' is a matter for investigation. One question is that of completeness – does the grammar describe *all* sentences of the language? In this respect, one can see that our example above is woefully inadequate. Another issue is whether a grammar is correct in the sense of allowing only sentences that are in fact grammatical: our example grammar falls down in this respect, since it allows the examples in (11), among many others.

(11) a. *User cleans.

[2] For some reason, linguists' trees are always written upside down, with the 'root' at the top, and the leaves (the actual words) at the bottom.

b. *Users cleans printers.
c. *Users should cleans printers.

A grammar may also be incorrect in associating constituents with the wrong categories. For example, as we noted above, one would probably prefer a grammar which recognizes that determiners and nouns make up NPs, and that the NP that occur in S (i.e. subject NPs) and those that appear in VP (object NPs) are the same (as our grammar does) to a grammar which treats them as belonging to different categories — this would suggest (wrongly) that there are things that can appear as subjects, but not as objects, and vice versa. This is obviously not true, except for some pronouns that can appear as subjects but not as objects: *I*, *he*, *she*, etc. A worse defect of this kind is the treatment of words – the grammar gives far too little information about them, and completely misses the fact that *clean*, and *cleans* are actually different forms of the same verb. We will show how this problem can be overcome in Chapter 5.

In a practical context, a further issue is how easy it is to understand the grammar, and to modify it (by extending it, or fixing mistakes), and how easy it is to use it for automatic processing (an issue to which we will return). Of course, all these matters are often related.

3.2.2 Further Analysis: Grammatical Relations

So far we have talked about the kind of grammatical knowledge that can be expressed in terms of a constituent structure tree — information about the constituent units, and the parts of speech. But there are other kinds of information implicit in these representations which it is useful to make explicit. In particular, information about which phrases fulfil which grammatical relations or **grammatical functions** such as SUBJECT, OBJECT and SENTENTIAL COMPLEMENT. English SUBJECTs are normally the NPs which come before the verb, and OBJECTs normally occur immediately after the verb. In other languages these relations may be realised differently with respect to the verb. For example, in Japanese the normal word order is SUBJECT OBJECT VERB, and in Irish and Welsh it is VERB SUBJECT OBJECT. In many languages, such as Russian, the VERB, SUBJECT and OBJECT can appear in essentially any order. (In such languages the different grammatical relations can often be recognized by different forms of the noun – usually called **cases**. In English, this only occurs with pronouns — *he*, *she*, etc., are only possible as SUBJECTs). What this suggests, of course, is that while the constituent structures of languages differ greatly, they may appear more similar when described in terms of grammatical relations.

Phrases which serve as SUBJECT, OBJECT, etc., should also be distinguished from those which serve as MODIFIERs, or ADJUNCTs, of various sorts. For example, in the sentence (12) *You* is the SUBJECT of the verb *clean*, *the printer casing* is its OBJECT, whilst the prepositional phrases *with a non-abrasive compound* and *at any time* are ADJUNCTs.

(12) You can clean the printer casing with a non-abrasive compound at any time.

ADJUNCTs are prototypically optional — unlike SUBJECTs. For example, a sentence

which omits them is still perfectly well formed: there is nothing wrong with (13a), but omitting the SUBJECT, as illustrated in (13b) produces an ungrammatical result.[3]

(13) a. You can clean the printer casing.
b. ∗Can clean the printer casing.

There are various ways of representing sentences in terms of grammatical relations, but it is essentially not very different from that of constituent structure tree representation, which we have seen earlier in this chapter. The basic idea is to represent sentences in terms of their constituent parts (so a tree representation is convenient), but since one wants to represent the grammatical relation which the parts have to the whole, it is common to mark either the branches or the nodes with the appropriate relation. Figure 3.4 gives a representation of (14). This can be compared with a constituent structure representation for the same sentence in Figure 3.5.

(14) The temperature has affected the printer.

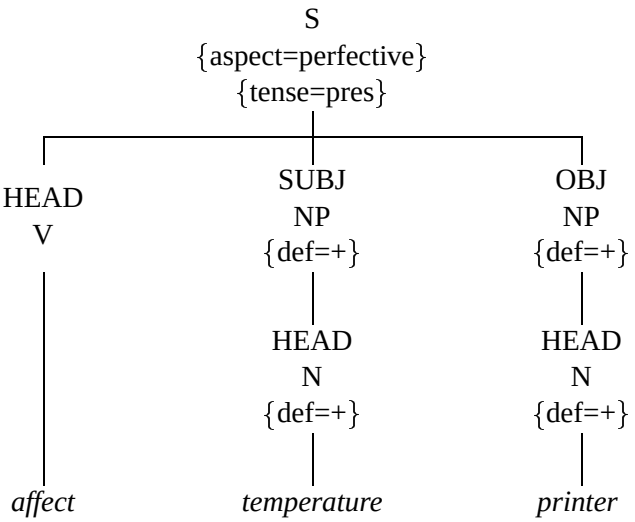

Figure 3.4 A Representation of Grammatical Relations

In Figure 3.4, the relations are marked on the nodes, and a new relation HEAD has been introduced. The HEAD element is, intuitively, the most important element from the point of view of the grammar of the whole phrase — the element which makes the phrase what it is. This is the noun in an NP, the verb in a VP or sentence, the preposition in a PP.

There are three important differences between this tree representing grammatical relations, and those representing constituent structure. First, instead of consisting of an NP, and a VP (containing a V and an NP), the representation of grammatical relations consists of a V and two NPs – the VP node has disappeared. Second, in this grammatical relations representation, the order of the branches is unimportant. This is possible, of course, because

[3]In English, SUBJECTs can only be omitted in imperative sentences, for example orders, such as *Clean the printer regularly*, and in some embedded sentences, e.g. the underlined part of *It is essential to clean the printer*

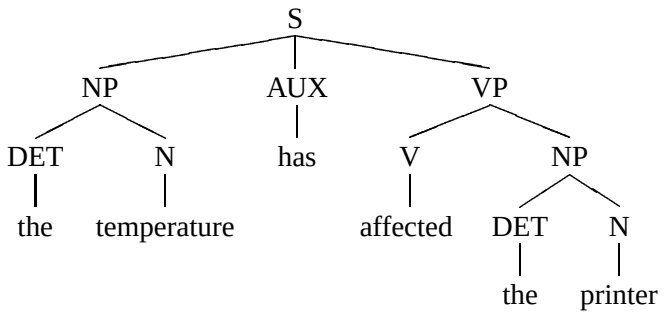

Figure 3.5 A Constituent Structure Representation

the grammatical relations have been indicated and this gives information about word order implicitly. Figure 3.4 could be redrawn with the branches in any order, and it would still be a representation of *The temperature affects the printer*, since this is the only sentence that has these items with these relations. By contrast, reordering the branches in a constituent structure tree might produce a representation of a quite different sentence, or no sentence at all.

The third difference is that some of the words have been missed out from Figure 3.4, and have been replaced by **features**, that is, pairs that consist of an **attribute**, such as `def`, `tense`, and `aspect`, and a **value**, such as `+`, `pres`, and `perfective`. The features `aspect=perfective` and `tense=pres` indicate that the sentence as a whole is in the present perfect tense. It is called perfect because it is used to describe events or actions that have been 'perfected' or completed, unlike, for example, a sentence such as *The temperature was affecting the printer*, where the 'affecting' is still going on at the time the writer is referring to. It is called *present* perfect because the auxiliary verb is in a present tense form (*has* not *had*). The feature `def=+` on the NPs means these NPs are definite. This definiteness indicates that the writer and reader have some particular object of the appropriate kind in mind. Compare, for example, *The printer has stopped* where one particular printer which is in some sense known to both writer and reader is being discussed, with *A printer has stopped*, where this is not the case.

These three differences are all intended to represent what is expressed by the sentence, abstracting away from the *way* it is expressed: we abstract away from the division into NP and VP, from the particular word order, and from the way in which the definiteness of the NPs and the tense and aspect of the sentence are realized (in English it is by the determiners, and the auxiliary verb respectively; in other languages it might be realized differently).

When it comes to describing the relationship between constituent structure, and what we might call relational structures, such as Figure 3.4, there are basically two approaches. One is simply to add information about grammatical relations to the grammar rules.

`S` $\rightarrow$ `NP{SUBJECT} AUX VP{HEAD}`

```
VP → V{HEAD} NP{OBJECT} PP{ADJUNCT}*

AUX → has{aspect=perfective, tense=pres}
```

The idea is that these annotations can be interpreted in such a way that a representation like Figure 3.4 can be constructed at the same time as the constituent structure tree. To do this requires a convention to 'flatten' the constituent structure tree 'merging' a structure (e.g. the structure of S) that is associated with the LHS of a rule with that of the HEAD daughter on the RHS, and a convention which simply merges in information that comes from items which do not have a grammatical relation, such as the AUX.

A second approach is to have special rules which relate the constituent structure representation to the representation of grammatical relations. One such rule might look like this:

```
[S NP:$1, AUX:$2, [VP V:$3, NP:$4 ]]
                  ↔
   [S HEAD:$3, SUBJ:$1, OBJ:$4 ]
```

In this rule, $1, $2, etc. are **variables**, or temporary names for pieces of structure. The idea is that such a rule matches a constituent structure such as that in Figure 3.3, and assigns (or 'binds') the variables to various pieces of structure. For example the NP containing *temperature* becomes bound to the variable $1. The rule can then be interpreted as an instruction to transform the constituent structure tree into a tree like Figure 3.4. This involves making this NP into the SUBJECT, making the V into the HEAD, and missing out the AUX entirely, among other things. The rule is rather simplified, of course, since it does not mention putting the information about perfective aspect into the grammatical relation representation, and ignores the problem of dealing with PPs, but it should give some idea.

The reader may also notice that the arrow used in this rule is bidirectional. This is intended to suggest that the rule simply states a correspondence between constituent structure, and grammatical relation representations, without suggesting that one is prior to the other. Thus, the idea is that one could equally well use the rule to transform Figure 3.4 into Figure 3.5 and vice versa. Similarly, the annotation approach is not supposed to be directional (though this may be somewhat harder to appreciate).

Many verbs have what are called **active** and **passive** forms, as in the following.

(15) a. Temperature affects printers. (*Active*)

b. Printers are affected by temperature. (*Passive*)

Notice that the object in the active sentence corresponds to the subject in the passive. This raises the question of what the grammatical relations SUBJECT and OBJECT should

mean. One possibility is to use the the terms in the sense of the 'surface' grammatical relations. The SUBJECTs of actives and the corresponding passives would be different, then. In particular, *temperature* would be the SUBJECT of (15a), and *printers* would be the SUBJECT of (15b). The alternative is to adopt a notion of a *deep* relation which picks out the same elements in both active and passive sentence. We would then say that (in English) the D-OBJECT ('deep' OBJECT) corresponds to the noun phrase after the verb in active sentences and to the noun phrase that precedes the verb in the corresponding passive. In active sentences, the surface and deep relations are the same, but they are different in passives, as can be seen from the following (in the passive sentence there is no surface OBJECT, and the D-SUBJECT has become a sort of ADJUNCT, in a PP with the preposition *by*).

(16) a. Temperature affects printers. (*Active*)
SUBJECT = *temperature*, OBJECT = *printers*
D-SUBJECT = *temperature*, D-OBJECT = *printers*

b. Printers are affected by temperature. (*Passive*)
SUBJECT = *printers*, OBJECT = 0,
D-SUBJECT = *temperature* D-OBJECT = *printers*

Interpreting SUBJECT as deep subject is clearly consistent with the general idea of abstracting away from surface characteristics in the grammatical relational representation. But it is not obviously the right move to make. For example, English verbs often vary their form depending on the nature of their subject (this is called **agreement** – as the following makes clear, there is also agreement of demonstratives like *this*/*these* with their head noun).

(17) a. These factors affect printers.
b. This factor affects printers.
c. *These factors affects printers.
d. *This factor affect printers.

However, the point to notice is that the agreement is with the surface subject, not the deep subject. Thus, if one wants to use a representation of grammatical relations to describe the phenomenon of agreement, the notion of SUBJECT had better be surface subject. This is not, in itself, a critical point here. The point we are making is simply that there is a range of options, and that the option chosen can make a difference for the overall description.

3.2.3 Meaning

Representing information about grammar in the form of grammar rules is useful in two ways in MT. First, as will become clear in the Chapter 4, it is possible to use the sort of linguistic representation that the rules provide to get simpler, and better descriptions of what is involved in translation, by abstracting away from some superficial differences between languages – as we have noted the abstract representations of sentences in different

languages are often more similar than the sentences themselves. But one can also use such representations as the basis for still more abstract representations of meaning. Working out the meaning of sentences is an important part of the translation process for human translators, and the ability to work out the meaning — to 'understand' (in some sense) the source text would allow an MT system to produce much better translations. This may sound an impossible task, and perhaps at some level it is. However, there is another, less ambitious, level where automatic 'understanding' is possible. In this section we will look at what this involves in a preliminary way (we will say more about it in Chapter 7).

It is useful to think of 'understanding' as involving three kinds of knowledge:

1 **Semantic** knowledge. This is knowledge of what expressions (individual words and sentences) mean, independent of the context they appear in.

2 **Pragmatic** knowledge. This is knowledge of what expressions mean in situations and particular occasions of use.

3 **Real world**, or common sense knowledge.

Consider the following example:

(18) The user may prefer to clean the printer every week with a non-corrosive fluid. Do not use abrasive or corrosive solvents, as this may harm its appearance.

One thing that is involved in understanding the meaning of this is working out the different **semantic relations** that the different NPs have to the predicates. For example, *a non-corrosive fluid* is understood as an *instrument* to be used in cleaning, *every week* indicates the time period in which the cleaning should be repeated, *the printer* denotes the thing to be cleaned, and *the user* denotes both the entity that has a preference, and which performs the cleaning. This is *semantic* information, because it is information that this sentence would convey on any occasion of use. However, recovering this information is not enough to 'understand' the example. One must also be able to work out that these sentences — or at least the second sentence — is to be understood as a *warning* not to do something. In this case, the form of the sentence is a fairly clear guide to this, but this is not always so. For example, sentences that are interrogative in form are often requests for information, but it is quite possible for such sentences to be interpreted as offers, requests for action, warnings, or as assertions (i.e. as giving information). This last case is what is called a rhetorical question; the following interrogatives might be interpreted in some of the other ways, depending on the context.

(19) a. Would you like some cake?
b. Don't you think it is cold in here?
c. Can't you see what you are doing to that printer?

Of course, the key words here are 'depending on the context'. Working out, for example, that (19b) is interpreted as a request for the speaker to close a window depends on many

things in the context where it is uttered (it might also, for example, be a comment on the social atmosphere). The sort of knowledge of social and linguistic conventions involved here is part of what is normally thought of as pragmatic knowledge.

But even this is not enough to understand the example completely. For example, there are the pronouns *this*, and *it* in the second sentence. It is obvious (to the human reader) that *this* should be interpreted as cleaning with an abrasive or corrosive solvent, and that *it* should be interpreted as referring to the printer (i.e. the sense is: 'cleaning ... may harm the printer's appearance'). But this is not the only semantically and pragmatically possible interpretation. One could imagine the same sentence being uttered in a context where it is the appearance of the fluid that will be affected (imagine one is dealing with a precious fluid of some kind):

(20) Do not place the fluid in sunlight, as this may harm its appearance.

What is involved here is real world, or common sense knowledge, perhaps the knowledge that if a corrosive fluid comes into contact with a printer (or something similar), it is the printer's appearance that is damaged. This is not knowledge about the meanings of the words, or about how language is used in different social contexts.

What You Say and What They Hear:
A Normal Conversation in the Linguistics Common Room

Similarly, consider the meaning of a word like *printers*. Semantic knowledge should supply the information that one interpretation of this refers to a collection of machines which perform the activity of printing, or perhaps to such things in general (as in *printers are expensive and unreliable*). Real world knowledge will indicate that the members of this collection are typically of a certain size (bigger than pencils, but smaller than houses, say),

and have certain parts, and characteristic flaws. When someone utters the word *Printers!*, in an exasperated tone, with a piece of chewed up paper in their hand, you may realize that what they intend to convey is some quite complicated attitude, including annoyance. It is pragmatic knowledge that allows you to work out that this is their intention, and that they do not, for example, want you to go and buy them a number of printers.

Of course, the distinctions between these different kinds of knowledge are not always clear, and they interact in complex ways in determining how an utterance is actually understood. Nevertheless, the basic idea of the distinction should be clear.

How can this sort of information about sentences be represented? The representation of pragmatic and common sense or real world knowledge raises many difficult problems, and is not really necessary for understanding the discussion in the following chapters, so we will postpone discussion until Chapter 6. However, we will say something about semantic representations here.

One kind of semantic representation would provide different relation names, and indicate which NP had which relation. In the following example, which is a simplified part of (18) , one might have relations like INSTRUMENT, AGENT (for the user), and THEME or PATIENT (for the printer), giving a representation like Figure 3.6 These relations are sometimes called **semantic roles**, **(deep) cases**, or **thematic roles**.

(21) The user cleans the printer with a non-abrasive solvent.

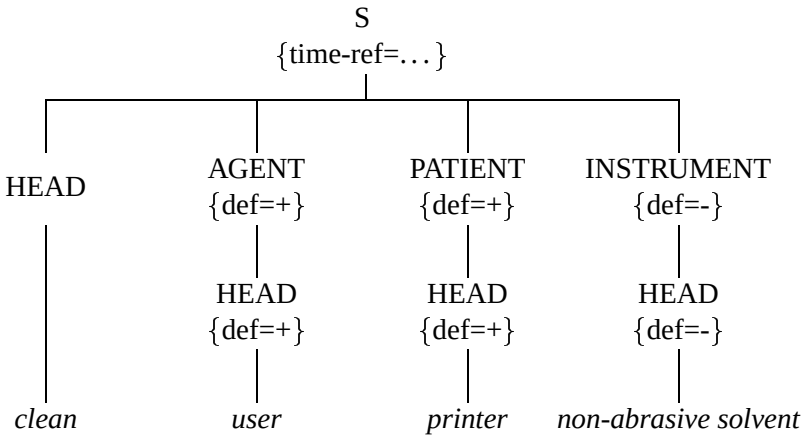

Figure 3.6 A Representation of Semantic Relations

Such a representation looks very much like Figure 3.4, except that the labels SUBJECT, OBJECT, etc. have been replaced by these thematic roles, and syntactic information about tense and aspect has been replaced by information about time reference.[4] The rules that relate these roles to grammatical relations would say things like "The AGENT will normally correspond to the SUBJECT of an active sentence, and an NP in a *by*-phrase in a passive sentence"; "The INSTRUMENT normally appears in a PP with the preposition *with*", "The PATIENT is very often the OBJECT of active sentences" However, there are

[4]We have not specified the time-reference information: see Chapter 7.

some verbs which violate these general patterns. For example, they are very different with *like* and *please* – the PATIENT (*bright colours* in the following example) is normally the OBJECT of *like*, but the SUBJECT of *please*.

(22) a. Children like bright colours.
b. Bright colours please children.

The usefulness of a semantic representation is further explored in Chapter 7.

3.3 Processing

In the previous sections, we have tried to give an outline of some of the different kinds of knowledge that are needed in text understanding (and hence, translation), and how they can be represented. We will now give an idea of how this knowledge can be manipulated automatically. We will do this in two stages. First, we will look at what is called **analysis**, or **parsing**. This is the process of taking an input string of expressions, and producing representations of the kind we have seen in the previous section. Second, we will look at **synthesis**, or **generation**, which is the reverse process – taking a representation, and producing the corresponding sentence.

It may be helpful to point out at the beginning that though the representations we have given are generally graphic objects — trees or networks drawn with lines — these are not themselves the representations that the computer deals with. For example, the standard internal representation of a tree is as a **list**, containing sublists, with any labels on a node being represented as the first element of the list. If we write lists between '(' and ')', and separate elements with commas, then the tree representation given in Figure 3.1 would look as follows (in fact, we have already shown this sort of representation for linguistic trees).

(S, (NP, (N, users)), (AUX, should), (VP, (V, clean), (NP, (DET, the), (N, printer))))

Lists are one of the datastructures that can be represented and manipulated very easily within a computer.

3.3.1 Parsing

The task of an automatic parser is to take a formal grammar and a sentence and apply the grammar to the sentence in order to (a) check that it is indeed grammatical and (b) given that it is grammatical, show how the words are combined into phrases and how the phrases are put together to form larger phrases (including sentences). So, for example, a parser would use the rules we gave above to check that the sentence *The temperature has affected the printer* consists of a noun phrase, consisting of the noun *Temperature* followed by an auxiliary verb, followed by a verb phrase, and that the verb phrase *affected the printer* consists of the verb *affect* and a noun phrase consisting of the noun *printers*. In effect, this gives the same information as the sorts of tree structure we have given

above, for example in Figure 3.5. Thus, one can think of a parser as taking sentences, and producing such representations (assuming the sentences are in fact well-formed according to the grammar).

How can this be done? There are many ways to apply the rules to the input to produce an output tree – many different **procedures**, or **parsing algorithms** by which an input string can be assigned a structure. Here is one method:

1 For each word in the sentence, find a rule whose right hand side matches it. This means that every word would then be labelled with its part of speech (shown on the left hand side of the rule that matched it). This step is exactly equivalent to looking up the words in an English dictionary. Given rules of the type `N` $\rightarrow$ `user`, `N` $\rightarrow$ `printer`, and `V` $\rightarrow$ `clean`, this will produce a partial structure as we can see at the top left corner (Stage 0) of Figure 3.7.

2 Starting from the left hand end of the sentence, find every rule whose right-hand side will match one or more of the parts of speech (Stage 1 of Figure 3.7).

3 Keep on doing step 2, matching larger and larger bits of phrase structure until no more rules can be applied. (In our example, this will be when the sentence rule finally matches up with a noun phrase and a verb phrase which have already been identified). The sentence is now parsed (Stage 2-4 of Figure 3.7).

It is generally possible to find more than one algorithm to produce a given result. As already mentioned, this is certainly true of parsing: the algorithm given here is just one of many possible variants which differ in their ability to cope efficiently with different types of grammar. The one we gave started out with the words of the sentence, and built the tree 'bottom up'. However, we could also have used an algorithm that built the tree 'top-down', starting with the S node. Essentially, what this algorithm would do is guess that it is looking at a sentence, and then guess that the sentence starts with a noun phrase, and then guess that the noun phrase consists of a noun, and then check to see whether there really is a noun at the start of the sentence. Each time there is a choice of possibilities (maybe the noun phrase starts with a determiner) it makes the first choice and, if that proves incorrect, backs up and tries the next alternative. During the course of parsing a sentence with a complicated grammar it would eventually get the right answer – perhaps only after many wrong guesses. (The algorithms that MT and other NLP systems use are more sophisticated and efficient than this, of course). The first few stages in a top-down parse are illustrated in Figure 3.8.

This description applies only to building the surface, constituent structure tree, of course. As regards other levels of representation (representations of grammatical relations, and semantic representations), there are two basic approaches, as we noted above. If information about other levels of representation is represented as annotations on the constituent structure rules, then it should be possible to construct these other representations at the same time as the constituent structure representation. This is slightly harder if the relationships between levels is stated in a separate collection of rules. In this case, the natural thing to

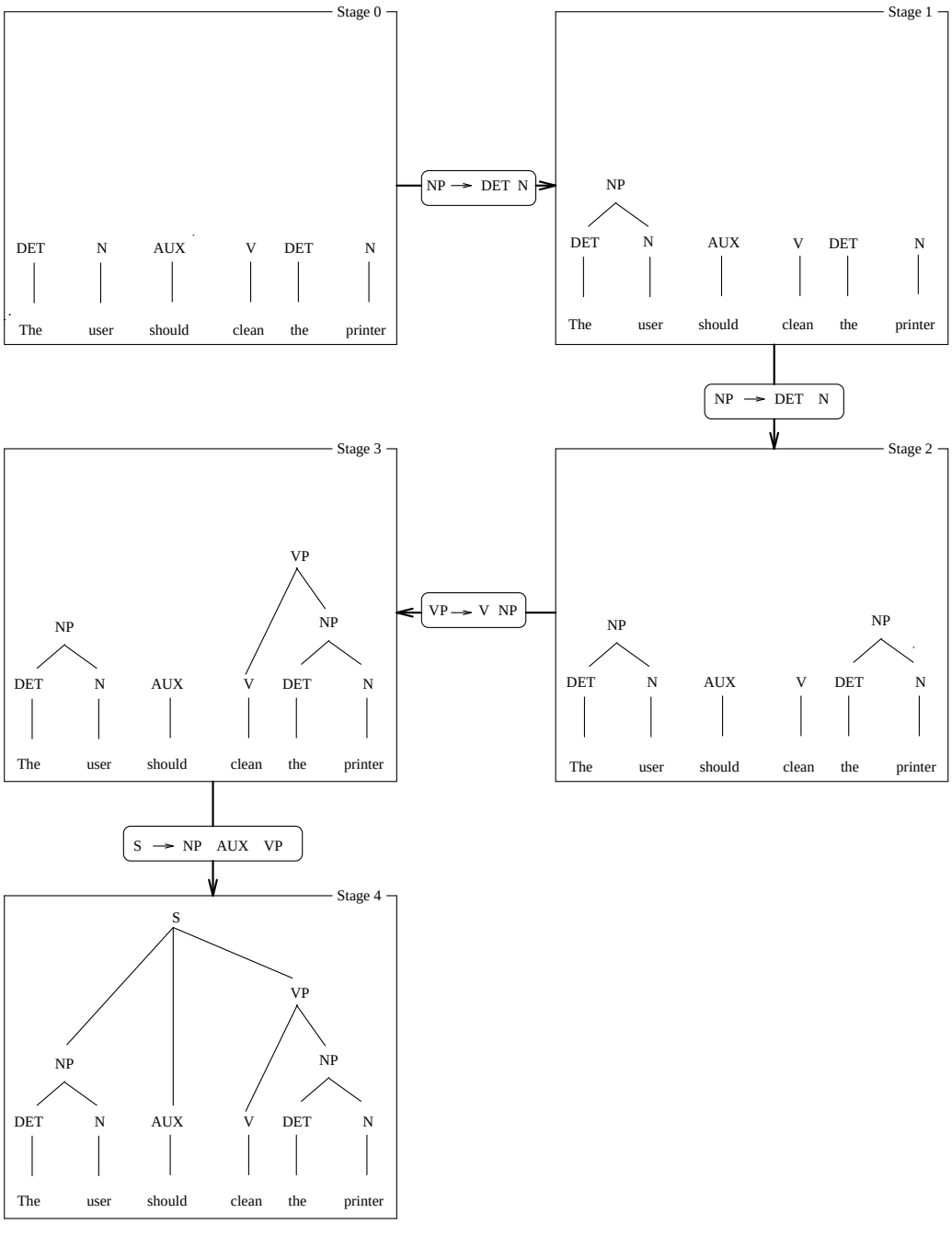

Figure 3.7 Parsing Using a Bottom-Up Algorithm

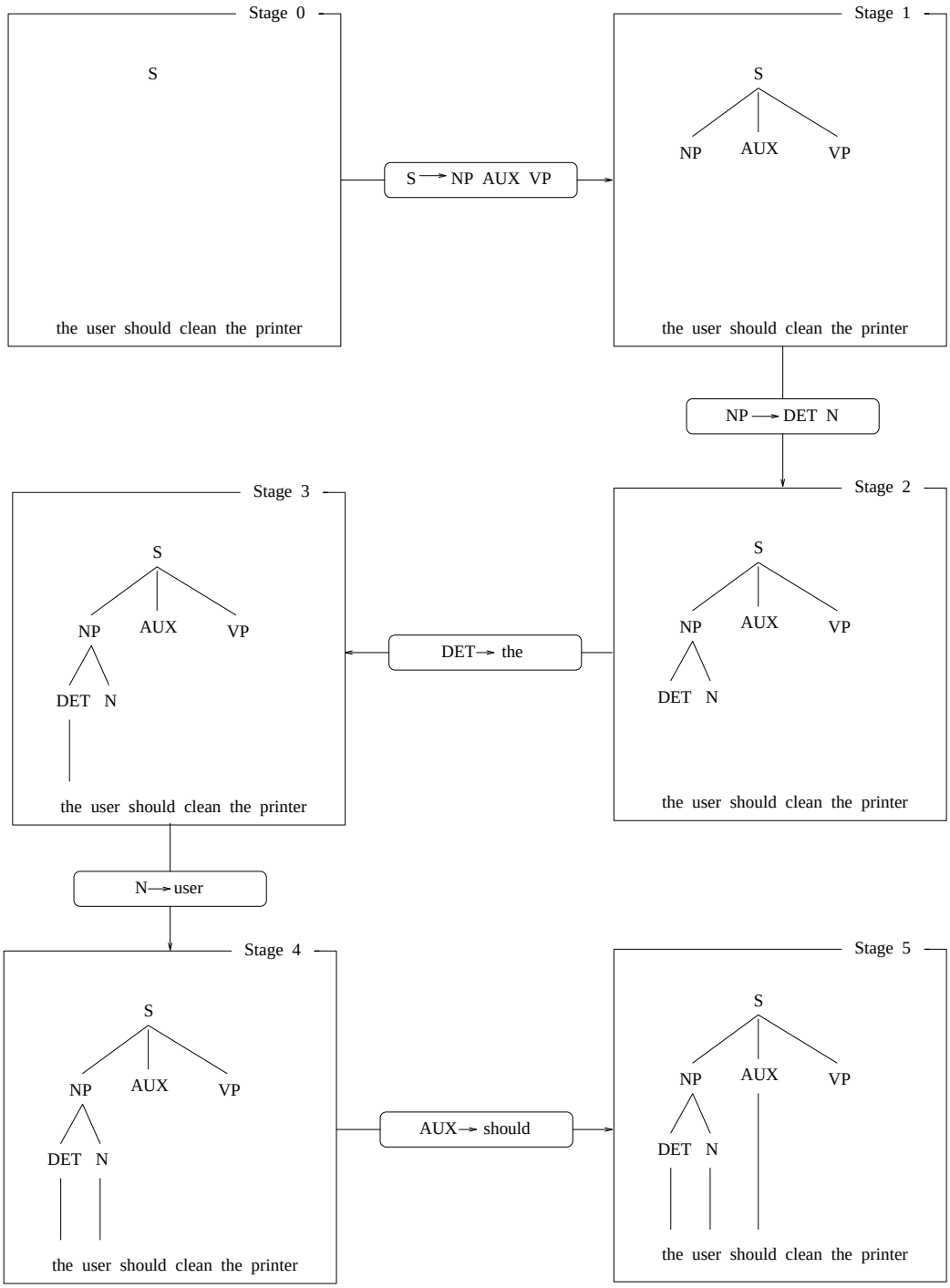

Figure 3.8 Parsing Using a Top-Down Algorithm

do is to first build the constituent structure representation, and apply these rules to that representation.

The simplest procedure for this operates 'recursively' down the surface (constituent) structure tree, dealing with each node in turn. Beginning at the root node, the algorithm looks for a rule whose lhs matches this node, and its daughters. In the case of the following rule (which we gave above, but repeat here for convenience), this means the root node must be labelled with an S, and there must be three daughters, labelled NP, AUX, and VP, and the VP must in turn contain a daughter labelled V, and a daughter labelled NP.

$$[_{S}\ \texttt{NP:\$1, AUX:\$2,}\ [_{VP}\ \texttt{V:\$3, NP:\$4}\ ]]$$
$$\leftrightarrow$$
$$[_{S}\ \texttt{HEAD:\$3, SUBJECT:\$1, OBJECT:\$4}\ ]$$

One interpretation of such a rule leaves the constituent structure tree untouched, and creates a new structure representing the grammatical relations. This requires the algorithm to create a temporary structure corresponding to the rhs of the rule. This will be labelled S, and will contain three daughters, one the HEAD, one the SUBJECT, and one the OBJECT. Of course, this structure cannot be complete yet, because it is not yet known what these daughters should contain. However, the algorithm now deals with the daughter nodes of the surface structure tree in exactly the same way as it dealt with the root node (hence the process is called recursive). That is, it tries to find rules to match each of NP, AUX, V, and NP, and produce the corresponding structures. When it has done this, it will be able to fill in the parts of the temporary structure it created originally, and a representation of the grammatical relations will have been produced. This can be seen in Figure 3.9.

A similar procedure can be used to interpret the rules that relate grammatical relation structures to semantic structures. There are a number of details and refinements which should really be described, such as how we ensure that all possible grammatical relation structures are produced, what we do about nodes that are mentioned on the LHS but not on the RHS, and so on. But these are refinements, and do not matter here, so long as this basic picture is clear.

3.3.2 Generation

So far, we have described how to take an input string, and produce a representation. But, obviously, for most applications, the reverse process is also necessary. Equally obviously, how hard this is depends on where you start from. Generating a string from a constituent structure representation like those above is almost trivial. At worst one needs to do something to the words to get the correct form (e.g. to get *clean*, not *cleans* in *The user should clean the printer regularly*). For the rest, it is simply a matter of 'forgetting' what structure there is (and perhaps the not-so-trivial matter of arranging punctuation).

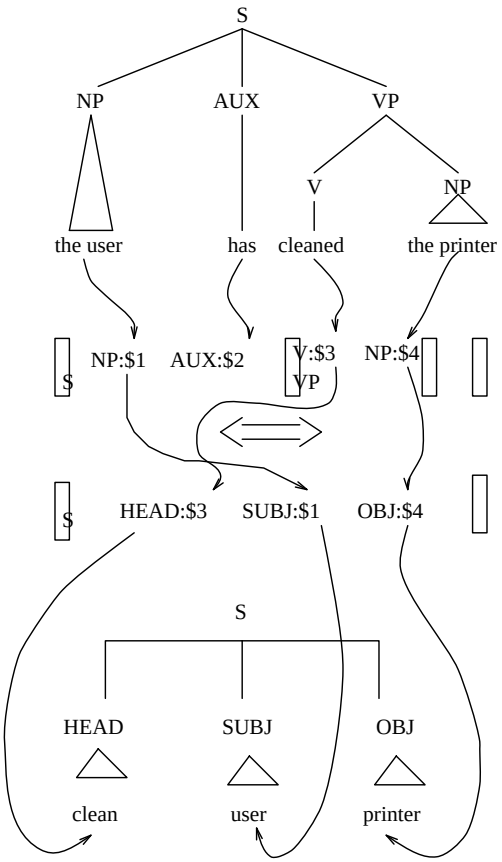

Figure 3.9 Building a Representation of Grammatical Relations

Starting from a representation of grammatical relations, or a semantic representation is harder.

If the relations between syntactic, grammatical relation structures, and semantic structures are described by means of explicit rules, then one approach is to use those rules in the same way as we described for parsing, but 'in reverse' — that is with the part of the rule written after the $\longleftrightarrow$ interpreted as the lhs. Things are not quite so straightforward when information about grammatical relations, and/or semantics is packed into the constituent structure rules.

One possibility is to have a completely separate set of procedures for producing sentences from semantic or grammatical relation structures, without going through the constituent structure stage (for example, one would need a rule that puts HEAD, SUBJECT, and OBJECT into the normal word order for English, depending on whether the sentence was active or passive, interrogative or declarative). This has attractions, in particular, it may be that one does not want to be able to generate exactly the sentences one can parse (one may want one's parser to accept stylistically rather bad sentences, which one would not want to produce, for example). However, the disadvantage is that one will end up describing again most, if not all, of the knowledge that is contained in the grammar which is used for

parsing.

A naive (and utterly impractical) approach would be to simply apply constituent structure rules at random, until a structure was produced that matched the grammatical relation structure that is input to generation. A useful variation of this is to start with the whole input structure, and take all the rules for the category S (assuming one expects the structure to represent a sentence), and to compare the grammatical relation structure each of these rules produces with the input structure. If the structure produced by a particular rule matches the input structure, then build a partial tree with this rule, and mark each of these parts as belonging to that tree. For example, given the rule for S above, one could take the grammatical relation structure of a sentence like *The user has cleaned the printer* and begin to make a phrase structure tree, as is illustrated in Figure 3.10.

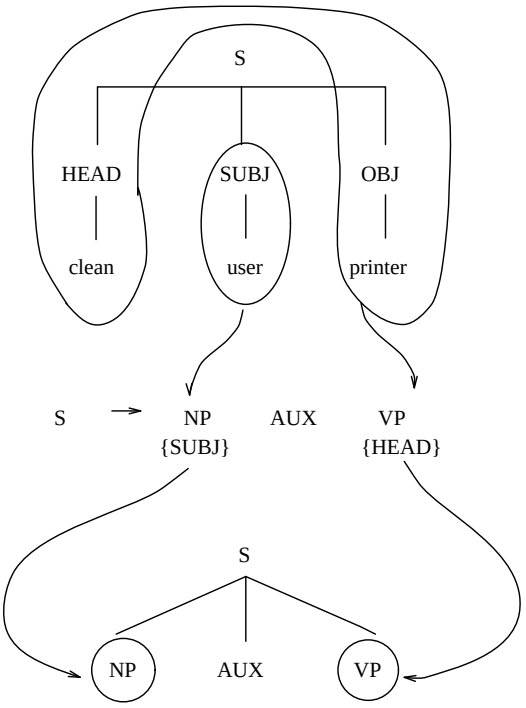

Figure 3.10 Generation from a Grammatical Relation Structure 1

One can see that a partial constituent structure tree has been created, whose nodes are linked to parts of the grammatical relation structure (a convention is assumed here whereby everything not explicitly mentioned in the rule is associated with the HEAD element). Now all that is necessary is to do the same thing to all the parts of the Grammatical relation structure, attaching the partial trees that have been constructed in the appropriate places. This is illustrated in Figure 3.11. Again, there are many refinements and details missed out here, but again, all that matters is the basic picture.

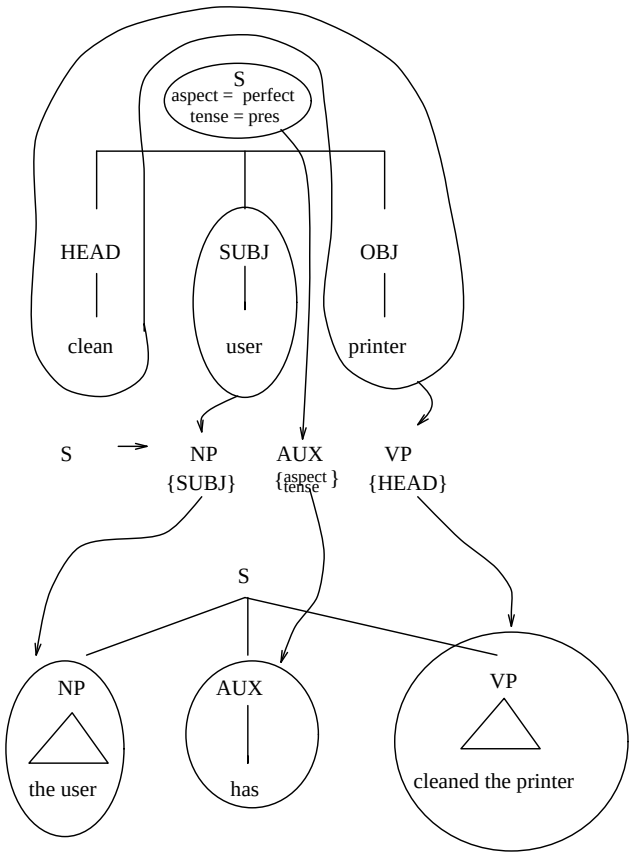

Figure 3.11 Generation from a Grammatical Relation Structure 2

3.4 Summary

This chapter has introduced the different kinds of knowledge needed to do translation, namely grammatical or syntactic knowledge, semantic, pragmatic and real world knowledge. Focussing on syntactic and semantic knowledge, we then looked at how this knowledge can be represented and described. Finally, again concentrating on syntax and semantics, we looked briefly at how this knowledge can be used for processing by means of parsing and generation algorithms.

3.5 Further Reading

A somewhat more detailed discussion of many of the issues touched on in this Chapter can be found in Hutchins and Somers (1992), especially Chapters 1, 3, 5, and 7.

The issue of how linguistic knowledge should be represented and described is one of the key concerns of Linguistic theory, and will be covered by most introductory books on Linguistics. On syntax, Brown and Miller (1991) is an accessible introduction. An elementary introduction to linguistic semantics can be found in Hurford and Heasley (1983),

a somewhat more advanced introduction can be found in Kempson (1977).

It is by no means the case that linguists agree on the sorts of representation that are required, though the use of some kind of constituent structure is almost universal. In particular, there is disagreement about how one should think about more abstract levels of representation. Here Borsley (1991) provides a useful comparative discussion at an introductory level. Discussion of the special requirements that MT makes of linguistic representation and description can be found in Van Eynde (1993b).

The issue of how linguistic representations and descriptions can be used for processing is the topic of the fields of Computational Linguistics and Natural Language Processing (NLP). Here Allen (1987); Grishman (1986); Gazdar and Mellish (1989) and Winograd (1983) provide excellent introductions, though all go well beyond what is required for a basic understanding. Parts of Charniak and Wilks (1976) are more elementary, though now somewhat out of date.

Much work in NLP focusses on analysis rather than synthesis or generation. For an introduction to issues in generation, see McDonald (1987).

NLP is also a key area of interest in the field of Artificial Intelligence (AI), and many introductions to AI contain some useful introductory material on NLP, examples are Rich (1983); Charniak and McDermott (1985); Tennant (1981); Barr and Fiegenbaum (1981). Many of the entries in Shapiro (1987) will also be useful.