# CS5012 - Language and Computation

# Practical 2 - Grammar Engineering

Student number: 130018883 Words: 1070

---

**Introduction**

The purpose of this practical was to engineer a grammar that for a small subset of English, defined by eight sentences to be parsed and complemented by three invalid sentences that should return no parses. This was done firstly using a context free grammar that defines rules to parse all of the sentences, and then refined to a unification grammar that includes more strict rules that prevent invalid sentences from being parsed.

**Context Free Grammar**

The first goal was to define a context free grammar that successfully parsed all eight of the example language. I did this by first writing a lexicon that tagged each part of speech that appeared in any of the sentences. I then looked at the sentences systematically, and added rules to the grammar that were necessary to parse them. For example, the first sentence has the basic parse shown below in Figure 1:

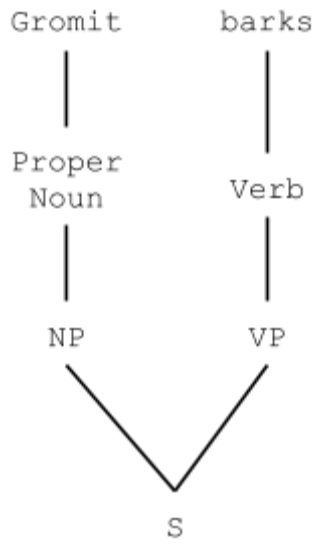


Figure 1

In order to encode this, I added the rules displayed in Figure 2, which were sufficient to successfully parse the sentence when combined with the entries in the lexicon that converted parts of speech to individual words:

```
S → NP VP

NP → ProperNoun

VP → Verb
```

Figure 2

After I looked at each sentence, I ran my model to confirm that the sentence was successfully parsed, and that all previous sentences were still being parsed, as well as looking for ambiguity in the parsing.

The one sentence that I found particularly tricky was when `'does Wallace eat cheese'`. I'd originally parsed it by taking `'when does'` as a question phrase, and adding the a rule `'S → QP S'`, which did succeed in parsing the sentence correctly. However, when I started the next section and began seeking agreement between words I realised that this was not the right way to parse the sentence. I also noticed that the sentence fragment `'does Wallace eat cheese'` should be a valid sentence too, so looked for an alternative.

After some research I found a method whereby the fragment `'does Wallace eat cheese'` is classified as a sentence question, and a rule `'S → Wh SQ'` is added to parse sentences of the form I had, so I chose to implement this. I don't include a `'S → SQ'` rule, as there are no sentences of this form in our language, however if this is added then sentences such as `'does Wallace eat cheese'` can be parsed successfully.

Once I had gone through all sentences, I had defined all of the rules necessary to successfully parse all of the sentences. Furthermore, almost all sentences only had a single parse, which I was happy with, as it prevents any ambiguity. The only exception was with the sentence `'Gromit often eats tasty cheese in the kitchen on Friday'` which had two separate parses. This was due to the rule `'NP → NP PP'` and that this sentence has two consecutive prepositional phrases at the end, which can be taken in either order. I believe that both parses are technically permissible, although I would prefer if there was a unique parse that corresponded to the most natural reading of the sentence. For the purposes of this practical I was happy to leave the ambiguity in, but it would be something I would look at if I were to go into more detail.

I also added the three invalid sentences to this model, and these were parsed successfully, as expected. In order to prevent these from being accepted, more granular rules would need to be added, and the grammar refined to a unification grammar.

**Unification Grammar**

In order to prevent grammatically incorrect sentences being accepted by this grammar, some features must be added to represent agreement, and categorisation of verbs.

The first thing I looked at was to accept sentences like `'Gromit barks'` but reject sentences like `'Gromit eat cheese'` i.e. forming agreement between nouns and verbs. I added the `'AGR'` feature to phrases in the language, and within this agreement feature had an entry for number and person. The majority of rules that were adapted simply used a variable such as `'ARG=?a'` as the specific details did not matter, just that they agreed.

This lead to some interesting points that needed to be covered; e.g. the rule `'NP → NP Conj NP'` needs to be adapted so that the left hand side is always plural. The right hand side is unspecified, as they can be singular or plural, but the left is always plural as it describes at least two nouns.

The next thing I looked at was the particular sentence `'when does Wallace eat cheese'` as I saw it had the potential to be hard to deal with. As mentioned in the previous section, I had parsed this incorrectly at first, so had to redefine it with the sentence fragment `'does Wallace eat cheese'` parsed as a sentence question. I saw that the agreement must be applied between `'does'` and `'Wallace'` and that `'eat'` is the infinitive form of the verb, as opposed to a present tense. In order to make this acceptable in our grammar I added the `'FORM'` feature, and specified between infinitive form and when it was present tense.

I also took this opportunity to redefine how I parsed the gerund form of 'eating' in the sentence. I used a unique part of speech in my context free grammar, as it served a specific purpose, but since I had started to differentiate between verb forms I decided to redefine this as `'Verb[FORM=Gerund]'` and update the rules accordingly.

Finally, I had to look at how to prevent sentences of the form `'Gromit barks Wallace'` being parsed. The problem here is that the verb 'barks' is being parsed with 'Wallace' as an argument, which does not make sense in the real world. I added the `'SUBCAT'` feature to the verbs, and defined them as transitive and intransitive. For our example language I decided to make 'barks' intransitive. This is not a perfect representation, as one may bark something to another, but it's sufficient to define our language. Going forward, I'd look at not accepting proper nouns as arguments, or something similar, to account for this.

The total set features I added to the grammar are summarised in Figure 3 below. I found that after these were added, and the rules updated accordingly, the eight valid sentences were parsed successfully and the three sample invalid sentences were not parsed, as desired. There was one case of ambiguity again, that coincided with that seen in the previous grammar.

```
AGR    =  NUM = sg / pl
          PER = 1 / 3
FORM   =  Pres / Inf / Gerund
SUBCAT =  Trans / Intrans
```

Figure 3