CS5012 P2 Grammar Engineering Report

1 Introduction

The content of this practical is to build up grammar for a tiny set of English which contains several sentences to be parsed as well as sentences that should not be parsed, says negative examples. The tasks are done creating context-free grammar(CFG) at first and then it is refined to the featured grammar which implements number agreements and subcategorisation for verbs.
This report would explain the idea, implementation, critical reflection of the grammars mentioned above respectively.

2 CFG

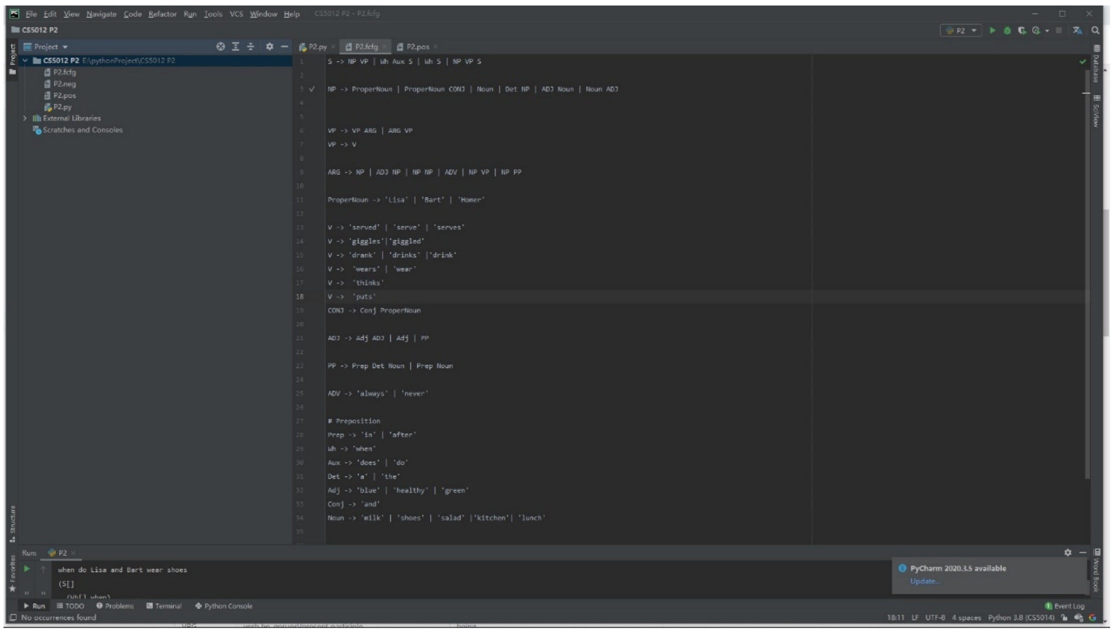
2.1 Inspiration

The first step of defining rules is to write down each part of speech that appears in any of the sentences and analyse their structure. In detail, I tagged each word and then thought about whether they might belong to a larger category, then looked at their possible position in the sentence. For example, nouns with postpositional preposition and nouns with adjectives should belong to the Nominal, i.e. the nouns with their modifiers, while the Nominal is a part of noun phrases(NP). Moreover, several sentences given are structured with NP followed by VP, but the arguments of the VP(ARG) vary, such as a single noun, a prepositional adverb, a prepositional noun, and a sub-sentence etc.

Note that even a context-free grammar built for the given sentence should have the possibility of parsing some sentences with similar structure.

2.2 implementation

Soon I set up a draft edition of the grammar:



```
S -> NP VP | Wh Aux S | Wh S | NP VP S

NP -> ProperNoun | ProperNoun CONJ | Noun | Det NP | ADJ Noun | Noun ADJ


VP -> VP ARG | ARG VP
VP -> V

ARG -> NP | ADJ NP | NP NP | ADV | NP VP | NP PP

ProperNoun -> 'Lisa' | 'Bart' | 'Homer'

V -> 'served' | 'serve' | 'serves'
V -> 'giggles'|'giggled'
V -> 'drank' | 'drinks' |'drink'
V ->  'wears' | 'wear'
V ->  'thinks'
V ->  'puts'
CONJ -> Conj ProperNoun

ADJ -> Adj ADJ | Adj | PP

PP -> Prep Det Noun | Prep Noun

ADV -> 'always' | 'never'

# Preposition
Prep -> 'in' | 'after'
Wh -> 'when'
Aux -> 'does' | 'do'
Det -> 'a' | 'the'
Adj -> 'blue' | 'healthy' | 'green'
Conj -> 'and'
Noun -> 'milk' | 'shoes' | 'salad' |'kitchen'| 'lunch'
```

```
when do Lisa and Bart wear shoes
(S[]
```

Figure 1. naïve version CFG

However, the ARG should not be able to derive so many categories if we want to implement the grammar in an elegant way, because the ARG often has more than one word, which means that the ARG needs to derive itself or the VP for derivation, which can lead to ambiguity. Alternatively, I allow ARG to derive the first component after VP, in this case, only NP. But if phrases such as 'a healthy green salad', 'the milk' can be derived by NP, the rule may look ugly, so I design the Nominal to handle the words above, which may be unnecessary.

Besides, the rule ‘VP -> ARG VP | VP ARG’ also allow redundant parses for sentences, such as 'when does Lisa drink milk in the kitchen', since two derivations are both applicable for them.

Therefore, I optimised the grammar based on these insights despite a few challenges and stored it in P2C.fcfg.

The one sentence that I found would be difficult if I delete 'VP -> ARG VP' is 'Bart always drinks milk' as there will be no rule to handle the premodifier of verbs. As a concession, I had to take the adverb out of the ARG and instead just allow the 'VP -> ADV V' directly.

Furthermore, another difficulty is sub-sentence structure, that is, 'Lisa thinks Homer thinks Bart drinks milk'. The pain point is the ARG of 'thinks' is a sub-sentence and it is repeated twice. I solved it by allowing S (start of the sentence) to loop itself for fitting infinite sub-sentence. But it may lead to other issues which will be discussed later.

And there is a particularly tricky one which is 'Lisa puts the milk in the kitchen. Grammatically, 'in the kitchen' is a PP(preposition) for 'puts', but if I design the rule 'ARG ->NP PP', the sentence could have several parses because this rule is repeated with 'Nominal -> Nominal PP'. And the PP would be parsed as the modifier of the Nominal.

2.3 Critical reflection

Once I had optimised the rules, there was much less possible parsing of the sentences. Actually, all the sentences only have a single parse except for the last one. The reason is the iterative derivation of Nominal but I believe that both parses are technically permissible as there is no ambiguity. Furthermore, the loop structure in the rules, e.g. 'Nominal -> Adj Nominal', allows the grammar to parse nouns with more adjective.

However, the exception is the structure of the parse tree. As I mentioned before, the fragment 'in the kitchen' does not modify 'puts' which is semantically and syntactically irrational. So the interpretation of this sentence is wrong from this perspective. Besides, I am not sure it is reasonable to allow the derivation of S. After all, S is supposed to represent the beginning of the sentence and should not appear in the middle of the parse tree. Besides, it also accepts some ungrammatical sentences or part of sentences, which shows that the CFG is not strict. it accepts sentence with

infinite preposition or determiner, e.g. ‘milk in the kitchen on Monday at home by bus’ or ‘a the an milk’, due to the loop in the Nominal rule.
Briefly, this CFG can parse all positive examples as well as some extra statements, although it might be semantically ambiguous for keeping the code simple and elegant.

3 Featured Grammar

This CFG can accept some negative samples that are grammatically incorrect while also parsing all positive samples because it does not constrain the sentence well. For instance, 'Lisa drink milk' can be parsed as its structure is legal and the words are implemented. That is why the featured grammar is introduced.

3.1 Inspiration

Intuitively, the featured grammar is implemented by constraining one of the parameters of the RHS and LHS of a rule to be equal. And for English, the number agreement should specify that the modifier corresponds to the singular and plural of the noun as well as the tense of verbs corresponding to its subject or modifier. Thus a basic implementation requires only 2 parameters since the number can also represent both forms of verbs in the present tense.

Subcategorisation of verbs can specify the structure of sentences to eliminate ambiguity, while the number agreement can eliminate grammatical errors. As the example shown, it consists of HEAD and TAIL, which specify the next ARG and the next ARG after that respectively.

In conclusion, the featured grammar can be engineered based on the CFG and the three parameters mentioned above for number agreement as well as subcategorisation.

3.2 Implementation

I generated rules by considering the verb as the head of the sentence, that is, all derivations are based on it, e.g. for rule 'V[TENSE = present, NUM = pl, SUBCAT = [HEAD= np, TAIL = [HEAD = pp, TAIL = nil]]]-> 'drink' ', it means that should be followed by a NP and a PP after that, and the number agreement shows that it is a general present tense verb. Therefore the verb can significantly determine the structure of the parse tree. Moreover, the rule 'VP[NUM=?n, TENSE=?t, SUBCAT=?rest] -> VP[NUM=?n, TENSE=?t, SUBCAT=[HEAD=?arg, TAIL=?rest]] ARG[CAT=?arg]' assigns head and tail to the ARG and VP respectively, allowing the ARG to appear multiple times with a clear hierarchy.

In addition to the above basic framework, I implemented the number agreement for NPs so that the featured grammar is built. However, I set SUBCAT as 'subsentence' for some verbs and set the rule 'SUBS -> NP[NUM=?n] VP[NUM=?n, TENSE=?t, SUBCAT=subsentence] S' to avoid fragments such as 'when Bart giggles' being parsed, which is different from my CFG.

3.3 Critical reflection

Compared to CFG, the featured grammar can parse all valid sentences without parsing any invalid sentences. In particular, the sentences have only one parse tree, meaning there is no ambiguity. Besides, 'in the kitchen' can now modify 'puts', i.e. the structure of the tree is correct in the last sentence.

Similarly, number agreements have been successfully implemented, for example, 'Bart wears a blue shoes' and 'Bart and Lisa drinks milk' which are ungrammatical can not be accepted as expected.

Nevertheless, I was concerned that the grammar may be too specific, as the sentence structure relies on the tagging of verbs. Even for the same verb, I have to write another rule if the number and type of ARGs are different. For instance, I need to design the very complex SUBCAT for 'drink' in the fragment 'drink milk after lunch on Monday'.

In conclusion, the featured grammar effectively avoids ambiguity and eliminates grammatical errors although it might be complex and difficult to expand.

4 Extension Works

Following sentences as extension part of this practical have been implemented by the grammar:

Bart likes drinking milk
Lisa may have giggled
Lisa may not have served Bart
what does Homer drink
what salad does Bart serve
whom does Homer serve salad
whom do Homer and Lisa serve
where does Lisa put the milk
what does Lisa put in the kitchen

They contain new structure such as nouns modified by auxiliary verbs, Wh+PP/NP, and verbs modified by gerunds. In addition to adding the verbs corresponding to the structure, I devised the rule that the tense specified by the specified auxiliary verb (VTENSE) must be equal to the tense of the linked verb and auxiliary verb. Once again, these rules are more difficult to parse other sentences, although they do not derive the wrong sentences, such as ‘may had’ .