AI2 Module 2

Tutorial 3: ANSWERS

Bonnie Webber

School of Informatics

1. What is the least inclusive class of languages that includes all strings of the form $a^m b^n c^o$, where $m, n, o \geq 1$?
Answer: $a^m b^n c^o$ is Finite State.

(a) Write a grammar that generates this language.
Answer: One such grammar is

$$S \rightarrow \begin{cases} \text{a} & \text{A} \\ \text{a} & \text{B} \end{cases}$$

$$A \rightarrow \begin{cases} \text{a} & \text{A} \\ \text{a} & \text{B} \end{cases}$$

$$B \rightarrow \begin{cases} \text{b} & \text{B} \\ \text{b} & \text{C} \end{cases}$$

$$C \rightarrow \begin{cases} \text{c} & \text{C} \\ \text{c} & \end{cases}$$

(b) Draw an automaton that recognizes this language.
Answer: The minimal automaton is just like slide 11 from the notes (for the language $a^m b^n$) but with an extra transition, state, and loop.

(c) Now write another grammar that generates this same language.
Answer: Another such grammar is the mirror image of the grammar in (a), which generates right-to-left, rather than left-to-right.

$$S \rightarrow \begin{cases} \text{C} & \text{c} \\ \text{B} & \text{c} \end{cases}$$

$$C \rightarrow \begin{cases} \text{C} & \text{c} \\ \text{B} & \text{c} \end{cases}$$

$$B \rightarrow \begin{cases} \text{B} & \text{b} \\ \text{A} & \text{b} \end{cases}$$

$$A \rightarrow \begin{cases} \text{A} & \text{a} \\ \text{a} & \end{cases}$$

2. Consider two grammars that produce the same set of strings, such as those you constructed for the last problem. Will such grammars necessarily have the same set of rules? When are such grammars **weakly** equivalent? When are they **strongly** equivalent?

Answer: Two grammars that generate the same string set are always **weakly** equivalent: That's the criterion for weak equivalence. They are **strongly** equivalent only when the trees that one grammar assigns to a member of the string set are always the same as the trees the other grammar assigns to it. So two grammars that produce the same string set are **not** necessarily strongly equivalent.

3(a & b). Produce two analyses of the sentence

I saw a man in the park with a telescope.

according to the grammar given in the tutorial.

Answer: The students may produce any two of the following:

1. Two analyses that use the rule vp→ v np.

   In one analysis, the object np uses the nom→nom pp rule twice. In another analysis, "park with a telescope" is derived through one application of the nom→nom pp rule, and "man in the [park with a telescope] is derived through its own application of this rule.

2. Three analyses that use the vp→vp pp rule.

   In one analysis, this rule is applied once, using the PP "with a telescope". In the second analysis, this rule is applied twice, one using the PP "with a telescope" and once using the PP "in the park". In the third analysis, the rule is applied once, using the PP "in the park with a telescope".

(c). These two are still not the only possible ways to analyse this sentence. How many ways are there? (Think in terms of the different ways you might characterise the man, the park and the seeing.) What are all the possible analyses?

Answer: As indicated above, there are 5 possible analyses. They can be paraphrased as follows:

1. At the time of his sighting, the man can be in the park and he can be carrying/handling a telescope.
2. At the time of his sighting, the man can be in the park that contains a telescope.
3. The manner of sighting the man in the park can be with a telescope.
4. The location of sighting the man can be in the park and the manner of sighting can be with a telescope.
5. The location of sighting the man can be in the park that contains a telescope.

(d). To recognise only 2' (the correct version of sentence 2), we need to add a feature for **number**, as in `dcg8`. Number can take the values `sg` or `pl` or _ (meaning either value). The feature needs to be added to both the rules and the lexicon. Here are some revised rules. For the rest, tell the students to look at `dcg8`.

Grammar rules:
s → np(Num), vp(Num).
np(Num) → pro(Num).
np(Num) → det(Num), nom(Num). Also needed, but not required by sentence 2
nom(Num) → n(Num).
Lexicon
pro(Num) → [Word], pro(Word,Num).
pro(she,sg).
tv(Num) → [Word], tv(Word,Num).
tv(sees,sg).
tv(see,pl).
n(Num) → [Word], n(Word,Num). Also needed, but not required by sentence 2
n(man,sg).

To recognise only 3' and 4' (the correct versions of sentences 3 and 4), we also need to add a feature for **case** or **role**. This feature for pronouns can be marked **nominative** (for subjec position), or **possessive** (aka **genative**), or **oblique** (everything else). The rules that have to carry this feature, in order to check it at the vp-forming rule are the following (already marked for **number**):

s → np(Num,nom), vp(Num).

vp(Num)→tv(Num),np(Num,oblq).

np(Num,_) → det(Num), nom(Num). % only pronouns have case in English
np(Num,Case) → pro(Num,Case).

pp → prep, np(Num,oblq).

The lexicon would have to contain extended entries such as:

pro(Num,Case) → [Word], pro(Word,Num,Case).
pro(she, sg, nom).
pro(she, sg, oblq).