ECS 140A Homework 1

Hardy Jones
999397426
Professor Olsson
Winter 2014

1. Indicate if the string belongs to the class of object at the head of the column by placing "yes" or "no" in the approriate box.

string	<chairs>	<set>	<table>	<room>
oak	no	no	yes	yes
armchair bench oak	no	no	no	yes
dinette plant	no	no	no	yes
bench stool	yes	no	no	no
oak rug	no	no	no	yes
oak teak dinette	no	no	no	no
teak teak	no	no	no	no
teak bench oak oak stool	no	no	no	no
dinette plant	no	no	no	yes
rug plant	no	no	no	no
couch	no	no	no	no
armchair bench	yes	no	no	no
teak lamp	no	no	no	no
bench stool teak bench stool	no	no	no	yes
bench stool teak bench teak stool	no	no	no	no
teak bench plant	no	no	no	yes

2. Rewrite the original grammar in EBNF notation (as given in the textbook).

⟨*room*⟩ ::= [⟨*chairs*⟩] ⟨*table*⟩
| ⟨*table*⟩ (couch | ⟨*stuff*⟩)

⟨*table*⟩ ::= oak
| teak
| dinette
| ⟨*table*⟩ ⟨*chairs*⟩

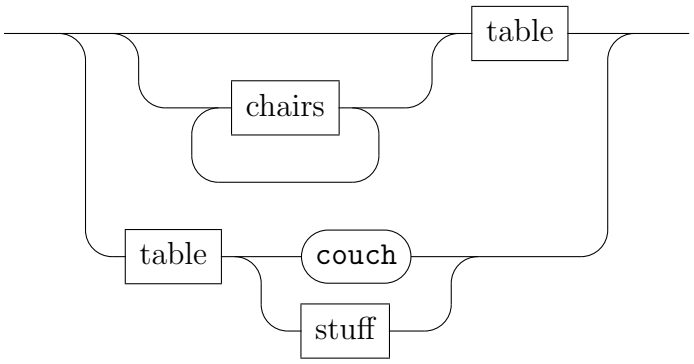
⟨*chairs*⟩ ::= ⟨*chair*⟩ { ⟨*chair*⟩ }

⟨*chair*⟩ ::= armchair
| bench
| stool

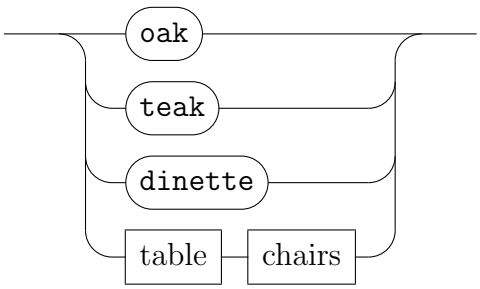
⟨*stuff*⟩ ::= rug
| plant

3. Rewrite the above grammar using syntax graphs (aka syntax diagrams).

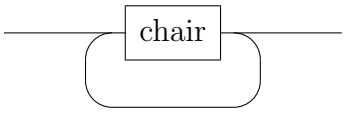
room



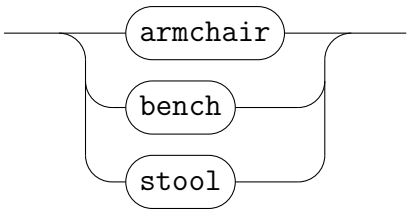
table



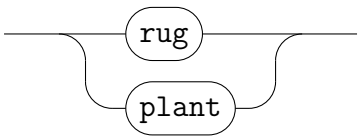
chairs



chair



stuff



4. Consider all strings that can be derived from <table>. Can each of those also be derived from <room>? Explain.

   Yes, every string that can be derived from <table> can be derived from <room>. Since a <room> can be just a <table>, we can make a straight derivation to <table>.

5. Suppose the production <table> <chairs> were added as an alternative in the rule for <room>. Would this change allow any additional strings to be considered a <room>? If yes, give one such new string. If no, explain.

   No, this would not allow additional strings in the language. Since <room> can be a <table>, and a <table> can be a <set>, and a set can be a <table> <chairs>, we would not have gained anything by adding this alternative.

6. Modify the original grammar so that in strings produced by <chairs>, any stools precede any armchairs or benches. Show only the changed rule(s) and any new rule(s), not the entire grammar.

   $\langle chairs \rangle$ ::= ($\langle stoolie \rangle$ | $\langle armbench \rangle$) { $\langle armbench \rangle$ }

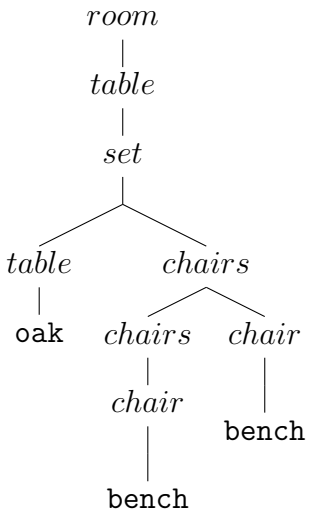
   $\langle stoolie \rangle$ ::= stool { $\langle stoolie \rangle$ }

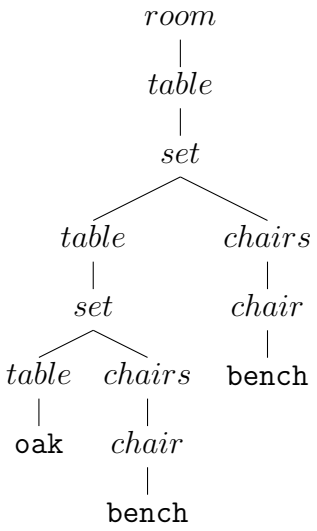
   $\langle armbench \rangle$ ::= armchair
   | bench

7. The original grammar does not give a unique parse tree for every input string. Give a string where the grammar is ambiguous and give two different parse trees for that string. What problems would ambiguity cause in a real programming language?

   A simple example is: `oak bench bench`.

   It can be parsed in the following ways.





   Ambiguity in a real programming language can lead to incorrect results. The simple case is arithmetic expressions. If the grammar is ambiguous with respect to arithmetic expressions, $1 + 2 \cdot 3$ could create the following incorrect parse tree resulting in 9.