CSE340 Spring 2021 - Homework 1 Solution

Due: Wednesday **February 3** 2021 by 11:59 PM on Gradescope

Note the due date which is different from the original planned due date for HW1

All submissions should be typed. Exception can only be made for drawing parse trees, which can be hand drawn and scanned in the submitted document.

When you submit your solution on Gradescope, you should indicate for each problem the page on which the solution is.

**Remember that no late submissions are accepted for homework assignments.**

**Problem 1.** Consider the list of tokens

T1 = { “abc”, abcd1e” }

T2 = { “abd” }

ID = Set of strings that consist of a letter or underscore that is followed by zero or more letters or digits.

NUM = Set of strings that consist of a non-zero digit that is followed by 1 or more digits or the string “0”.

Consider the input

abcd22\_abc\_ 123 0\_0abc abd 123abcd1e1

and the following sequence of calls:

t1 = lexer.GetToken();

t2 = lexer.GetToken();

t3 = lexer.peek(1);

t4 = lexer.peek(3);

t5 = lexer.peek(5);

t6 = lexer.peek(7);

t7 = lexer.GetToken();

t8 = lexer.GetToken();

Assume that space is a separator, but is otherwise ignored.

1. What are the values of t1, t2, t3, t4, t5, t6, t7 and t8?

**Answer**

The input will be broken down into tokens as follows

abcd22\_abc\_ 123 0\_0abc abd 123abcd1e1

ID ID ID NUM NUM ID ID NUM ID

t1 t2 t3 t4 t5 t6

t7 t8

The answer is

t1: {ID, “abcd22”}

t2: {ID, “\_abc”}

t3: {ID, “\_”}

t4: {NUM, 0}

t5: {T2, “abd”}

t6: {ID, “abcd1e1”}

t7: {ID, “\_”}

t8: {NUM, 123}

**Problem 2.** Consider the grammar

S → A B C

A → a A | ε

B → a B b| b

C → b C a | b

Assume that the conventions we discussed in class for writing grammars are followed.

1. What is the start symbol? Explain briefly!

The start symbol is S. By convention, and unless otherwise specified, the left hand side of the first rule is the start symbol

1. What are the non-terminals? Explain briefly!

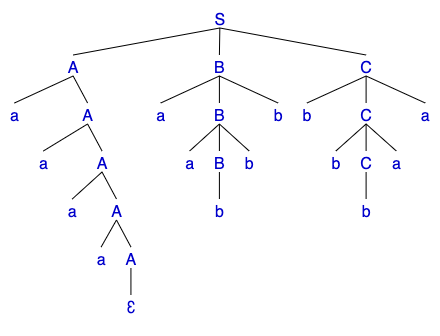
The non-terminals are S, A, B and C. By convention, and unless otherwise specified, the left hand side of the rules are the non-terminals.

1. What are the terminals? Explain briefly!

The terminals are a and b. By convention, and unless otherwise specified, the symbols, other than epsilon, that are not non-terminals are terminals. Epsilon is not a terminal. It does not correspond to a token

1. Give a parse tree for the input:

a a a a a a b b b b b b a a



1. In the parse tree of

a a a a a a b b b b b b a a

the root node is labeled S and its children are labeled A, B and C from left to right. Which parts of the input correspond to the children of S in the parse tree? You answer should have the following format:

The part of input that corresponds to A is aaaa

The part of input that corresponds to B is aabbb

The part of input that corresponds to C is bbbaa

The following is not needed, but I do it for emphasis. We break the input into three pieces corresponding to A, B and C as follows

a a a a a a b b b b b b a a

1. What is the language of this grammar (remember that the language of a grammar is the set of strings that can be derived from the start symbol or, equivalently, the set of strings that have parse trees according to the grammar). In your description of the language, you should try to be as precise but as brief as possible.

L(G) = {w ∈ T\*: S ⇒\* w }

where T = {“a”, “b”} and T\* is the set of sequences of symbols from T, including th empty sequence. In other words,

The language of G = set of strings of terminals that can be derived from S

This question is asking us to specify this set. Using the notation a*i* to denote *i* consecutive a’s (if *i* = 0, a*i* is epsilon), we can write

1. Set of strings that can be generated from A = { a*i* : *i* ≥ 0 }
2. Set of strings that can be generated from B = { a*i* b*i+1* : *i* ≥ 0 }
3. Set of strings that can be generated from C = { b*i+1* a*i* : *i* ≥ 0 }

L(G) = { a*i* a*j* b*j+1* b*k+1* a*k* : *i*,*j*,*k* ≥ 0 }

Note that we use three different indices because the parts that correspond to A, B and C are not related and need not have equal numbers of a’s.

In your solution, you do not need to write the answer in a format like the one above. You could describe A, B and C and say that L(G) consists of strings of the following form: zero or more initial sequence of a’s followed by a sequence of matching a and b’s with a b in the middle followed by a sequence of matching b’s and a’s with a b in the middle.

In the definition, a *sequence of matching x’s and y’s with a z in the middle* is a sequence of x’s followed by a z followed by a sequence of y’s equal in number to the x’s (matching).

1. Is this grammar ambiguous? Explain! (hint: think of a rightmost derivation)

The grammar is not ambiguous, because any input string can be represented by exactly one parse tree. To argue that this is indeed the case, consider a string in L(G). The string must have a number of a’s (from A and B) followed by a number of b’s (from B and C) followed by a number of a’s (from C).

Let us examine the string from right to left. Say, for example, that there are 3 a’s at the end. Before these 3 a’s we should have 4 b’s such that C = bbbbaaa. What we are saying is that the end of the string can only come from a C if C is bbbbaaa. Before the first b of C, there are a number of b’s coming from B. Let us say, for example that there are 3 b’s, this means that b must be aabbb given the general form of B. before the first a of B there must be a number of a’s, all of which must be from A.

What this example shows is that there is a unique way to break a string in L(G) into parts that correspond to A, B and C. Given a string w in L(G), we explained why there are unique wA, wB and wC such w = and A =>\* wA , B =>\* wB and C =>\* wC. It should be clear that derivations of strings from A, B and C are unique and given that there is a unique way to break the input into A, B and C, there is only one parse tree for the input.

**Note**. This explanation is more than I expect you to write in your solution. You can still get full credit if you have relevant elements of this explanation in your answer.

**Problem 3.** Consider the grammar

A → Z Y X | X Z Y

X → a B | Y

B → b B | X | ε

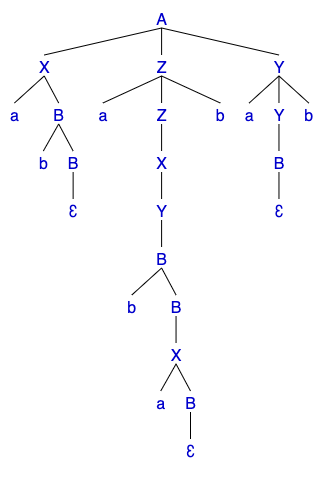
Y → a Y b | B

Z → a Z b | X

where A, B, X, Y and Z are non-terminal, A is the start symbol and a and b are tokens. Remember that ε represent the empty sequence. Y → ε means that Y does not have to match any tokens or, equivalently, it matches an empty sequence of tokens.

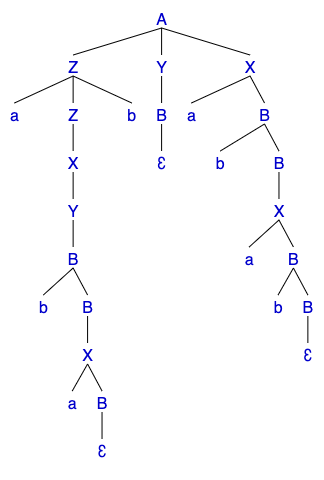
1. Give a parse tree for the sequence of tokens:

a b a b a b a b



1. Give a another (different) parse tree for the sequence of tokens:

a b a b a b a b



1. Is this grammar ambiguous? Why?

This grammar is ambiguous because the input a b a b a b a b has two different parse trees.