HW2 Solutions

February 13, 2013

1 T or F

1.a False.

1. All context-free languages are also context-sensitive languages (from notes).

2. A language is context-sensitive iff it can be described by a context-sensitive grammar (from notes).

3. A context-free language can be described by a context-sensitive grammar. $\because$ 1 and 2.

4. $\therefore$ *Not* every grammar describing a context-free language is a context-free grammar. $\because$ 3.

1.b True.

5. A language is context-free iff it can be described by a context-free grammar (from notes).

6. All context-sensitive languages are recursively enumerable languages (from notes).

7. $\therefore$ All context-free grammars describe recursively enumerable languages. $\because$ 5, 1, and 6.

1.c True.

8. A language is recursively enumerable iff it can be described by an unrestricted grammar (from notes).

9. $\therefore$ Any context free language can be defined with an unrestricted grammar. $\because$ 5, 7, and 8.

1.d True.

10. All regular grammars are context-free grammars (from notes).

1.e False.

11. Left-linear grammars allow for null productions on RHS, and context-sensitive grammars do not.

12. $\therefore$ *Not* all left-linear grammars are context-sensitive grammars. $\because$ 11.

1.f True.

13. $\therefore$ Any language described by a left-linear grammar can be described by a context-sensitive grammar. $\because$ 10, 5, 1, and 2.

1.g True.

14. A language is regular iff it can be described by a regular grammar (from notes).

15. All regular languages are also context-free languages (from notes).

16. $\therefore$ Any language described by a regular expression can be described using a context-free grammar. $\because$ 14 and 15

1.h True.

17. LL(k) $\subseteq$ LR(1), $\forall$k $\geq$ 1 (from Fall 2012 Week 5 notes)

1.i True.

18. LR = SLR(k1) = LALR(k2) = LR(k3), $\forall$k1 , k2 , k3 $\geq$ 1 (from notes).

19. $\therefore$ LL(3) $\subseteq$ LALR(k), $\forall$k $\geq$ 1. $\because$ 17 and 18.

1.j True.

21. LL(1) $\subset$ LR(k), $\forall$k $\geq$ 1 (from notes).

2 Grammar Classification

2.a Yes.

2.b No.

<B> y → y violates condition that γ must be non-empty.

2.c No.

2.d No.

2.e Type 2, but not Type 3 grammar:

<S> → <A>

<A> → <B> y <D>

<B> → x <B> | ε

<D> → <D> z | z

2.f Type 3 grammar:

<A> → x <A> | y <B>

<B> → z <B> | z

In Regular Expression form: x*yz+

3 Skipped

4 Derivations

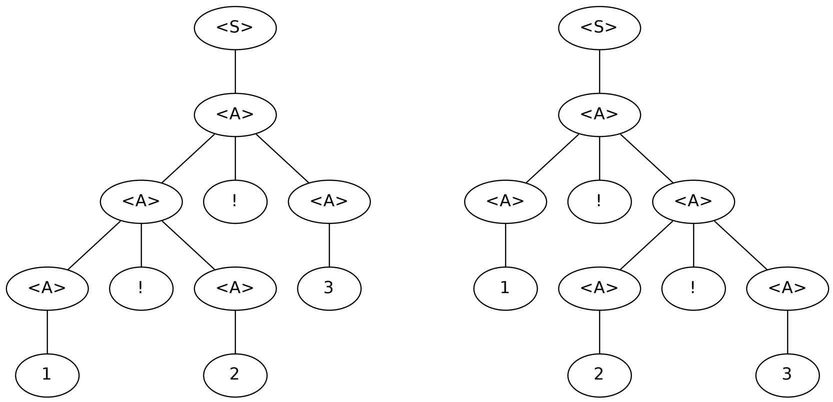
4.a Yes

4.b 2 right-most derivations:

<S> → <A> ! <A> → <A> ! 3 → <A> @ <A> ! 3 → <A> @ 2 ! 3 → 1 @ 2 ! 3

<S> → <A> @ <A> → <A> @ <A> ! <A> → <A> @ <A> ! 3 → <A> @ 2 ! 3 → 1 @ 2 ! 3

4.c 2 parse trees:



5 First and Follow

5.a Is ambiguous?

None of the grammars are ambiguous.

5.b First Sets

Grammar A:

First(<A>) = {tick,ε}

First(tick tock <A>) = {tick}

First(ε) = {ε}

Grammar B:

First(<A>) = {tick,ε}

First(<A> tick tock) = {tick}

First(ε) = {ε}

Grammar C:

First(<A>) = {tick}

First(tick tock <A>) = {tick}

First(tick tock) = {tick}

5.c Follow Sets

Grammar A:

Follow(<S>) = {eof}

Follow(<A>) = {eof}

Grammar B:

Follow(<S>) = {eof}

Follow(<A>) = {tick, eof}

Grammar C:

Follow(<S>) = {eof}

Follow(<A>) = {eof}

5.d First+ Sets

Grammar A:

First+(<A> $\rightarrow$ tick tock <A>) = {tick}

First+(<A> $\rightarrow$ ε) = {ε,eof}

Grammar B:

First+(<A> $\rightarrow$ <A> tick tock) = {tick}

First+(<A> $\rightarrow$ ε) = {ε,tick,eof}

Grammar C:

First+(<A> $\rightarrow$ tick tock <A>) = {tick}

First+(<A> $\rightarrow$ tick tock) = {tick}

5.e Is LL(1)?

Grammar A:

Yes. First+(<A> → tick tock <A>) ∩ First+(<A> → ε) = ∅

Grammar B:

Yes. Describes same language as Grammar A.

Grammar C:

Yes. Can be described with LL(1) grammar by left-factoring.

<S> → <A>

<A> → tick tock <B>

<B> → tick tock | ε

6 Unambiguous Context Free Grammars

Both are correct. A syntax tree generated by any derivation can be generated by a left-most derivation.

7 Rewrite to LL(1)

Remove left-recursion from R

<R> → aba<S> | caba<S>

<S> → bc<S> | ε

Left-factor Q to remove conflicts

<Q> → b<T>

<T> → bc | c

Compute First+ sets for all alternative productions, and show they are disjoint to prove the grammar is LL(1).

8 First sets needed to satisfy LL(k) condition.

LL(k) parsers have a lookahead of k. The parser can tolerate alternative productions that share the first k-1 symbols. So, to make sure an LL(k) parser can parse a grammar, one would have to compute the First$_k$+ sets for alternative productions, where the First$_k$+ function returns sets of strings containing the first k symbols of productions. If the First$_k$+ sets for all alternative productions are disjoint, then the grammar is an LL(k) grammar.

9 Elevator Grammar

Similar to the parenthesis matching problem:

$E \rightarrow E\uparrow E \downarrow \mid \varepsilon$

10 Simple Arithmetic Expression Grammar

10.a First+ sets.

No ε in grammar, so First+ $\equiv$ First sets.

First(<expr> → <term> + <term>) = {1,2,3,4,5,(}

First(<expr> → <term> - <term>) = {1,2,3,4,5,(}

First(<expr> → <term>) = {1,2,3,4,5,(}

Sets are obviously not disjoint, so grammar is not LL(1).

10.b Left-factor <exp> to derive LL(1) grammar.

<exp> → <term> <exp'>

<exp'> → + <term> | - <term> | ε