

CS2104 Tutorial 1: Syntax

1. Parsing and Ambiguous Grammars (Sebesta 3.12)

Consider the following BNF with $\langle S \rangle$ as the start symbol:

```
 $\langle S \rangle ::= a \langle S \rangle c \langle B \rangle \mid \langle A \rangle \mid b$   
 $\langle A \rangle ::= c \langle A \rangle \mid c$   
 $\langle B \rangle ::= d \mid \langle A \rangle$ 
```

- (a) For each of the following strings, state whether they are accepted or rejected by the grammar:
- abcd
 - acc cbd
 - accbcc
 - acd
 - acc
 - aaacdccccd
- (b) The grammar is ambiguous. What is the string that would result in more than one parse tree?

2. Writing Grammars

In JAVA, the while-statement has two forms:

```
while (condition)
    single-statement;

while (condition) {
    single-statement;
    single-statement;
    ...
}
```

You may assume that the BNF rule for the condition part is already defined as $\langle \text{condition} \rangle$, and the rule for single-statement as $\langle \text{statement} \rangle$.

- Write a BNF rule $\langle \text{while-stmt} \rangle$ that can match all JAVA while-statement.
- Write the same grammar using EBNF and compare the two forms.

3. Writing Grammars

For each of the following languages, where possible, use BNF (with $\langle S \rangle$ as the start symbol) to describe a grammar that would accept those, and only those sentences in the language. (You are allowed to use the empty string in your production rules).

- $L(G) = \{w \mid w \text{ is an even binary number without leading zeros}\}$
eg. $01 \notin L(G)$, $1 \in L(G)$, $10 \in L(G)$, $010 \notin L(G)$, $0 \in L(G)$, $100110 \in L(G)$ etc.
- $L(G) = \{ww \mid w \text{ is any binary string}\}$
eg. $10 \notin L(G)$, $11 \in L(G)$, $1010 \in L(G)$, $101101 \in L(G)$, $101111 \notin L(G)$.
- $L(G) = \{w \mid w \text{ is any binary string with even number of 1's and odd number of 0's}\}$
eg. $10 \notin L(G)$, $110 \in L(G)$, $011 \in L(G)$, $101 \in L(G)$, $10101 \notin L(G)$, $01010 \in L(G)$.

4. Equivalent Grammars

Two grammars are *equivalent* iff any string accepted by the first grammar, is also accepted by the second, and vice versa.

Check whether the following pairs of grammars are equivalent (where $\langle S \rangle$ is the starting non-terminal). (Try to (i) look for counter example for those unequal grammars; or/and also (ii) understand what the grammar tries to do)

- | | | |
|-----|--|---|
| (a) | Grammar I
$\langle S \rangle ::= 1 \langle A \rangle$
$\langle A \rangle ::= 11 \mid 11\langle A \rangle$ | Grammar II
$\langle S \rangle ::= 1\langle S \rangle \mid e$
(‘e’ is the empty string) |
| (b) | Grammar I
$\langle S \rangle ::= \langle \text{int} \rangle + \langle S \rangle$ | Grammar II
$\langle S \rangle ::= \langle S \rangle + \langle \text{int} \rangle$ |
| (c) | Grammar I
$\langle S \rangle ::= a \mid b \mid a \langle S \rangle \mid b \langle S \rangle$ | Grammar II
$\langle S \rangle ::= \langle S \rangle \langle S \rangle \mid a \mid b$ |

5. Modifying Grammars

There are a few ways to resolve the dangling-else ambiguity given in the grammar:

$$\begin{aligned} \langle S \rangle ::= & \text{if } \langle E \rangle \text{ then } \langle S \rangle \\ & \mid \text{if } \langle E \rangle \text{ then } \langle S \rangle \text{ else } \langle S \rangle \end{aligned}$$

- (a) You are to remove the ambiguity by *modifying* the grammar to...
- ...match the ‘else’ to the nearest (or innermost) nested ‘if’.
 - ...introduce a word ‘endif’ at the end of an if-statement.
- (‘modify’ meaning that you are allowed to add/remove terminals, non-terminals and rules).
- (b) Are your two modified grammars *equivalent* to the original ambiguous grammar?
- (c) How does (i) C, (ii) Java avoid/resolve the dangling-else ambiguity?

6. Modifying Grammars

A grammar defined using empty strings (most of the time) can be modified to an equivalent one without the use of empty strings.

- (a) Given the following grammar with ‘e’ as the empty string:

$$\begin{aligned} \langle A \rangle ::= & a\langle B \rangle c\langle B \rangle a \\ \langle B \rangle ::= & b\langle B \rangle \mid e \end{aligned}$$

how would you modify it to an equivalent one which does not use the empty string?

- (b) Note that sometimes, this cannot be done. Give an example of such a grammar.