# *Homework 3 -Solution*

1. ***Examine the revised MiniJava grammar posted for Project 2. Find out the non-terminals which have production rules that will make it impossible to generate an LL (k) parser.***

Solution:

- Nonterminal Type fails test 1.1 as it has a left recursive production rule.
- Nonterminal Expression fails test 1.1 as it has left recursive production rules.

For a rule which is left recursive the FIRST sets of it productions will always overlap no matter how large the lookahead is.

2. ***Exercise 3.6***

Solution:

(a)

| | nullable | FIRST | FOLLOW |
|---|---|---|---|
| S | No | u | |
| B | No | w | v y x z |
| D | Yes | y x | z |
| E | Yes | y | x z |
| F | Yes | x | z |

**(b) Parse Table**

| | U | v | w | x | y | Z |
|---|---|---|---|---|---|---|
| S | S → uBDz | | | | | |
| B | | | B →W<br>B →Bv | | | |
| D | | | | D →EF | D→EF | D→EF |
| E | | | | E → | E→y | E→ |
| F | | | | F → x | | F → |

**(c)** The grammar is left recursive (B→Bv), hence cannot be LL(1) (in general LL(k)).

**(d)** Replace the left recursive rule (B→ Bv) and rule B→w with the following rules:

B → wB'

B' → vB' | ϵ

3. *Exercise 3.7 (a) and (b) in the textbook.*

Solution:

Consider the following grammar:

$$\begin{aligned} S &\rightarrow G\$ \\ G &\rightarrow P \\ G &\rightarrow PG \\ P &\rightarrow i : R \\ R &\rightarrow \epsilon \\ R &\rightarrow iR \end{aligned}$$

(a) Left-factor this grammar.

**Answer:**

$$\begin{aligned} S &\rightarrow G\$ \\ G &\rightarrow PG' \\ G' &\rightarrow G \mid \epsilon \\ P &\rightarrow i : R \\ R &\rightarrow iR \mid \epsilon \end{aligned}$$

(b) Show that the resulting grammar is LL(2). You can do this by constructing FIRST sets *etc*, containing 2-symbol strings; but it is simpler to construct an LL(1) parsing table and then argue convincingly that any conflicts can be resolved by looking ahead one more symbol.

**Answer:**

Here is the LL(1) parse table:

| | FIRST | FOLLOW | $i$ | $\$$ |
|---|---|---|---|---|
| $S$ | $\{i\}$ | $\{\$\}$ | $S \rightarrow G\$$ | |
| $G$ | $\{i\}$ | $\{\$\}$ | $G \rightarrow PG'$ | |
| $G'$ | $\{i, \epsilon\}$ | $\{\$\}$ | $G' \rightarrow G$ | $G' \rightarrow \epsilon$ |
| $P$ | $\{i\}$ | $\{i, \$\}$ | $P \rightarrow i : R$ | |
| $R$ | $\{i, \epsilon\}$ | $\{i, \$\}$ | $R \rightarrow \epsilon$<br>$R \rightarrow iR$ | $R \rightarrow \epsilon$ |

Note that there are two productions predicted for $R$ on lookahead $i$: $R \rightarrow \epsilon$ and $R \rightarrow iR$, so the grammar is not LL(1). The conflict arises because $R \rightarrow \epsilon$ is predicted by

$$\text{FOLLOW}(R) \subseteq \text{FOLLOW}(P) \subseteq \text{FIRST}(G) \subseteq \text{FIRST}(P) = \{i\}$$

while $R \rightarrow iR$ is predicted by $\text{FIRST}(i) = \{i\}$. We cannot tell if the lookahead $i$ is part of a continuing recursion on $R$ or part of what can follow $R$, in this case $P$. Consider what happens if we lookahead one more token and see the semicolon (:) instead of $i$ — this disambiguates the predicted production, allowing us to decide if the lookahead $i$ is part of a continuing recursion on $R$ or the beginning of a derivation from $P$, which can legally follow derivations from $R$.

4.

a).

| Stack | Input | Action |
|---|---|---|
| $ | 000111$ | shift |
| $0 | 00111$ | shift |
| $00 | 0111$ | shift |
| $000 | 111$ | shift |
| $0001 | 11$ | Reduce S->01 |
| $00S | 11$ | shift |
| $00S1 | 1$ | Reduce S->0S1 |
| $0S | 1$ | shift |
| $0S1 | $ | Reduce S->0S1 |
| $S | $ | accept |

b)

| Stack | Input | Action |
|---|---|---|
| $ | aaa*a++$ | shift |
| $a | aa*a++$ | Reduce S->a |
| $S | aa*a++$ | shift |
| $Sa | a*a++$ | Reduce S->a |
| $SS | a*a++$ | shift |
| $SSa | *a++$ | Reduce S->a |
| $SSS | *a++$ | Shift |
| $SSS* | a++$ | Reduce S->SS* |
| $SS | a++$ | shift |
| $SSa | ++$ | Reduce S->a |
| $SSS | ++$ | Shift |
| $SSS+ | +$ | Reduce S->SS+ |
| $SS | +$ | shift |
| $SS+ | $ | Reduce S->SS |
| $S | $ | accept |

***5. Exercise 3.5 in textbook.***

**Solution:**

FIRST and FOLLOW sets.

| | **nullable** | **FIRST** | **FOLLOW** |
|---|---|---|---|
| S' | No | $ { WORD begin end \ | |
| S | Yes | { WORD begin end \ | $ \ } |
| B | No | \ | { WORD begin end \ |
| E | No | \ | $ } { WORD begin end \ |
| X | No | { WORD begin end \ | $ } { WORD begin end \ |

LL(1) Parse table.

| | **$** | **** | **{** | **}** | **Begin** | **WORD** | **end** |
|---|---|---|---|---|---|---|---|
| **S'** | S'→ S$ | S'→ S$ | S'→ S$ | | S'→ S$ | S'→ S$ | S'→ S$ |
| **S** | S → | S →<br>S → XS | S → XS | S → | S → XS | S → XS | S → XS |
| **B** | | B→ \ begin {WORD} | | | | | |
| **E** | | E → \end {WORD} | | | | | |
| **X** | | X → BSE<br>X → \ WORD | X → {S} | | X→ begin | X→WORD | X →end |