Introduction to Language Theory and Compilation
Exercises
Session 5: Grammars revisited

Reminders

A **grammar** is described by four components $\langle V, T, P, S \rangle$ where:

- V is the set of variables
- T is the set of terminals
- P is the set of production rules

$$P \subseteq (V \cup T)^* V (V \cup T)^* \times (V \cup T)^*$$

- $S \in V$ is the start symbol

Removal of unproductive symbols

Grammar `RemoveUnproductive(`**Grammar** $G = \langle V, T, P, S \rangle$`)` **begin**
    $V_0 \leftarrow \emptyset$;
    $i \leftarrow 0$;
    repeat
        $i \leftarrow i + 1$;
        $V_i \leftarrow \{A \mid A \rightarrow \alpha \in P \wedge \alpha \in (V_{i-1} \cup T)^*\} \cup V_{i-1}$;
    until $V_i = V_{i-1}$;
    $V' \leftarrow V_i$;
    $P' \leftarrow$ set of rules of P that do not contain variables in $V \setminus V'$;
    `return(`$G' = \langle V', T, P', S \rangle$`)` ;

Removal of inaccessible symbols

Grammar `RemoveInaccessible(`**Grammar** $G = \langle V, T, P, S \rangle$`)` **begin**
    $V_0 \leftarrow \{S\}$; $i \leftarrow 0$;
    repeat
        $i \leftarrow i + 1$;
        $V_i \leftarrow \{X \mid \exists A \rightarrow \alpha X \beta$ in $P \wedge A \in V_{i-1}\} \cup V_{i-1}$;
    until $V_i = V_{i-1}$;
    $V' \leftarrow V_i \cap V$; $T' \leftarrow V_i \cap T$;
    $P' \leftarrow$ set of rules of P that only contain variables from V_i ;
    `return(`$G' = \langle V', T', P', S \rangle$`)` ;

Removal of useless symbols

```
Grammar RemoveUseless(Grammar G = ⟨V,T,P,S⟩) begin
    Grammar G1 ← RemoveUnproductive(G) ;
    Grammar G2 ← RemoveInaccessible(G1) ;
    return(G2) ;
```

Left factoring

```
LeftFactor(Grammar G = ⟨V,T,P,S⟩) begin
    while G has at least two rules with the same left-hand side and a common prefix do
        Let R = {A → αβ, ..., A → αζ} be such a set of rules ;
        Let 𝒱 be a new variable;
        V = V ∪ 𝒱 ;
        P = P \ R ;
        P = P ∪ {A → α𝒱, 𝒱 → β, ..., 𝒱 → ζ};
```

Removal of left recursion

```
RemoveLeftRecursion(Grammar G = ⟨V,T,P,S⟩) begin
    while G contains a left recursive variable A do
        Let R = {A → Aα, A → β, ..., A → ζ} be the set of rules that have A as left-hand side ;
        Let 𝒰 and 𝒱 be two new variables ;
        V = V ∪ {𝒰, 𝒱} ;
        P = P \ R ;
        P = P ∪ {A → 𝒰𝒱, 𝒰 → β, ..., 𝒰 → ζ, 𝒱 → α𝒱, 𝒱 → ε} ;
```

Exercises

Ex. 1. Remove the useless symbols in the following grammars:

$$(G_1)\begin{cases} S & \to & a \mid A \\ A & \to & AB \\ B & \to & b \end{cases}$$

$$(G_2)\begin{cases} S & \to & A \\ & & B \\ A & \to & aB \\ & & bS \\ & & b \\ B & \to & AB \\ & & Ba \\ C & \to & AS \\ & & b \end{cases}$$

Ex. 2. Consider the following grammar:

$$\begin{cases} E & \to & E\ op\ E \\ & & ID[E] \\ & & ID \\ op & \to & * \\ & & / \\ & & + \\ & & - \\ & & \Rightarrow \end{cases}$$

- Show that the above grammar is ambiguous.
- The priorities of the various operators are as follows: $\{[], \Rightarrow\} > \{*, /\} > \{+, -\}$.

  Modify the grammar in order for it to take operator precedence into account as well as left associativity.

Ex. 3. Left-factor the following production rules:

<stmt> $\to$ **if** <expr> **then** <stmt-list> **end if**
<stmt> $\to$ **if** <expr> **then** <stmt-list> **else** <stmt-list> **end if**

Ex. 4. Apply the left recursion removal algorithm to the following grammar:

$$\begin{cases} E & \to & E+T \\ & & T \\ T & \to & T*P \\ & & P \\ P & \to & ID \end{cases}$$

Ex. 5. (Exam-level question)

Definition. A CFG $\langle P,T,V,S \rangle$ is **LL(1)** iff for all pairs of derivations:

$$S \to^* wA\gamma \to w\alpha_1\gamma \to^* wx_1$$
$$S \to^* wA\gamma \to w\alpha_2\gamma \to^* wx_2$$

with $w, x_1, x_2 \in T^*$, $A \in V$ and $\gamma \in (V \cup T)^*$, *and* $First(x_1) = First(x_2)$, where $First(x)$ designates the first letter of the word x, we have: $\alpha_1 = \alpha_2$.

Start by removing unproductive symbols and then inaccessible symbols on the following grammar:

$$\begin{cases} S & \to & aE \mid bF \\ E & \to & bE \mid \varepsilon \\ F & \to & aF \mid aG \mid aHD \\ G & \to & Gc \mid d \\ H & \to & Ca \\ C & \to & Hb \\ D & \to & ab \end{cases}$$

Is the grammar obtained $LL(1)$? If necessary, apply left-recursion removal algorithm and left-factoring and check again.