Introduction to Language Theory and Compilation Exercises

Session 5: Pushdown automata and parsing

Reminder

A *pushdown automaton* (PDA) P is described by 7 components: $\langle Q, \Sigma, \Gamma, \delta, q_0, Z_0, F \rangle$

- Q is a finite set of states, $q_0 \in Q$ is the starting state and $F \subseteq Q$ is the set of accepting states,
- Σ is a finite *input alphabet*,
- Γ is a finite *stack alphabet*, $Z_0 \in \Gamma$ is the start symbol on the stack,
- $\delta : Q \times (\Sigma \cup \{\varepsilon\}) \times \Gamma \to 2^{(Q \times \Gamma^*)}$ is the transition function.

A PDA *configuration* is a triple $\langle q, w, \gamma \rangle \in Q \times \Sigma^* \times \Gamma^*$:

- $q \in Q$ is the current state
- $w \in \Sigma^*$ is the remaining input
- $\gamma \in \Gamma^*$ is the current stack content

The *initial configuration* of P when reading a word w is thus $\langle q_0, w, Z_0 \rangle$.

Configuration change: Given two configurations $\langle q, aw, X\beta \rangle$ and $\langle q', w, \alpha\beta \rangle$ of P, where $a \in \Sigma \cup \{\varepsilon\}$ and $X \in \Gamma$, we say that P can move from configuration $\langle q, w, \gamma \rangle$ to configuration $\langle q, w, \gamma \rangle$ iff $(q', \alpha) \in \delta(q, a, X)$. In this case, we write $\langle q, aw, X\beta \rangle \vdash_P \langle q', w, \alpha\beta \rangle$.

Accepted Languages

A PDA P defines two languages, $L(P)$ and $N(P)$ depending on which acceptance notion is used:

- $L(P)$, or *final state accepted language*: A word w is accepted by P if there is an execution of P on w that ends in a final state of P.

  More formally: $L(P) = \{w \mid \text{there are } q \in F \text{ and } \gamma \in \Gamma^* \text{ such that } \langle q_0, w, Z_0\beta \rangle \vdash_P^* \langle q, \varepsilon, \gamma \rangle\}$
- $N(P)$, or *empty stack accepted language*: A word w is accepted by P if there is an execution of P on w that ends with the stack of P being empty.

  More formally: $N(P) = \{w \mid \text{there is } q \in Q \text{ such that } \langle q_0, w, Z_0\beta \rangle \vdash_P^* \langle q, \varepsilon, \varepsilon \rangle\}$

Exercises

Ex. 1. Design a pushdown automaton that accepts the language made of all words of the form ww^R where w is any given word on the alphabet $\Sigma = \{a, b\}$ and w^R is the mirror image of w. Test your automaton on the input word *abaaaaba*.

Ex. 2. Give the parse tree for the following input according to the grammar presented in Table 1:

```
begin      ID := ID - INTLIT + ID ;      end
```

Ex. 3. A ***top-down parser*** builds a parse tree using a top-down approach in which a given grammar $G = \langle V, T, P, S \rangle$ will be assimilated to the following PDA M ($\mid Q^M \mid = 1, \$ \in \Sigma$):

$$M = \langle \{q\}, T \cup \{\$\}, V \cup T \cup \{\$\}, \delta, q, S \rangle$$

For simplicity, we suppose that the rules of the grammar G are indexed and ordered by numbers, that is, $P = \{r_1, \ldots, r_n\}$. The stack is initialized with the grammar's start symbol ($S \dashv$). We now define the transitions of M. There are actually three kinds of transitions in the transition function δ:

Match $\langle q, ax, a\gamma \rangle \rightarrow \langle q, x, \gamma \rangle$: we match the top of the stack with the next input symbol and remove both

Produce $\langle q, x, A\gamma \rangle \rightarrow \langle q, x, \alpha\gamma \rangle$ if there is a production rule r_i that has the form $A \rightarrow \alpha$: we replace a variable A on top of the stack with its production α

Accept $\langle q, \$, \$ \dashv \rangle \rightarrow \langle q, \varepsilon, \dashv \rangle$: we match the "end of input" symbols and signal that we accept the given input

Simulate a top-down parser on the following input according to the grammar presented in Table 1:

```
begin      A :=  BB - 314 + A ;      end
```

Remark In practice, it is also very useful to keep track of the rules used in the Produce transitions of accepting executions !

Ex. 4. A ***bottom-up parser*** builds a parse tree using a bottom-up approach in which a given grammar $G = \langle V, T, P, S \rangle$ will be assimilated to the following PDA:

$$M = \langle \{q\}, T \cup \{\$\}, V \cup T \cup \{\$\}, \delta, q, \varepsilon \rangle$$

We start with an empty stack. The three kinds of transitions in the transition function δ are:

Shift $\langle q, \alpha x, \gamma \rangle \rightarrow \langle q, x, \gamma\alpha \rangle$: push the next input symbol on the stack

Reduce $\langle q, x, \gamma\alpha \rangle \rightarrow \langle q, x, \gamma A \rangle$ if there is a rule r_i of the form $A \rightarrow \alpha$: replace the corresponding input α by the corresponding symbol A on the stack, without touching the input

Accept $\langle q, \varepsilon, \vdash S \rangle \rightarrow \langle q, \varepsilon, \varepsilon \rangle$: we accept the input if we manage to get to the end of the input with the start symbol on the stack

Simulate a bottom-up parser on the same input according to the grammar presented in Table 1.

(1)	<S>	$\rightarrow$	<program> $
(2)	<program>	$\rightarrow$	begin <statement list> end
(3)	<statement list>	$\rightarrow$	<statement> <statement tail>
(4)	<statement tail>	$\rightarrow$	<statement> <statement tail>
(5)	<statement tail>	$\rightarrow$	ε
(6)	<statement>	$\rightarrow$	ID := <expression> ;
(7)	<statement>	$\rightarrow$	read (<id list>) ;
(8)	<statement>	$\rightarrow$	write (<expr list>) ;
(9)	<id list>	$\rightarrow$	ID <id tail>
(10)	<id tail>	$\rightarrow$	, ID <id tail>
(11)	<id tail>	$\rightarrow$	ε

(12)	<expr list>	$\rightarrow$	<expression> <expr tail>
(13)	<expr tail>	$\rightarrow$	, <expression> <expr tail>
(14)	<expr tail>	$\rightarrow$	ε
(15)	<expression>	$\rightarrow$	<primary> <primary tail>
(16)	<primary tail>	$\rightarrow$	<add op> <primary> <primary tail>
(17)	<primary tail>	$\rightarrow$	ε
(18)	<primary>	$\rightarrow$	(<expression>)
(19)	<primary>	$\rightarrow$	ID
(20)	<primary>	$\rightarrow$	INTLIT
(21)	<add op>	$\rightarrow$	$+$
(22)	<add op>	$\rightarrow$	$-$

Table 1: CF grammar where <S> is the start symbol (see last rule) and $ denotes the end of the input