



Compiler Construction WS07/08

Solution Sheet 5

1 LL(k)

a) Let $k \in \mathbb{N}$ and G an $LL(k)$ -grammar.

For any $u, x, y \in V_T^*$, and $\alpha, \beta, \gamma \in (V_T \cup V_N)^*$, such that:

$$S \Rightarrow_{lm}^* uY\alpha \Rightarrow_{lm} u\beta\alpha \Rightarrow_{lm}^* ux, \text{ and}$$

$$S \Rightarrow_{lm}^* uY\alpha \Rightarrow_{lm} u\gamma\alpha \Rightarrow_{lm}^* uy.$$

one of the following holds:

1. $(k+1) : x = (k+1) : y$

Obviously $k : x = k : y$ and because G is an $LL(k)$ -grammar, $\beta = \gamma$ holds.

2. $(k+1) : x \neq (k+1) : y$

That case is irrelevant, as the $k+1$ prefixes of x and y are distinguishable anyway.

Thus G is also an $LL(k+1)$ -grammar. ■

b) Let the grammar $G = (\{S\}, \{a, b, c\}, P, S)$ with the set of productions P :

$$S \rightarrow \underbrace{a \cdots a}_k b$$

$$S \rightarrow \underbrace{a \cdots a}_k c$$

The grammar G is obviously an $LL(k+1)$ grammar, as looking at the next $k+1$ input symbols suffices to determine the production rule to take. However, G is not an $LL(k)$ grammar, because:

$$S \Rightarrow_{lm}^* uS\alpha \Rightarrow_{lm} \underbrace{a \cdots a}_k b = x, \text{ and}$$

$$S \Rightarrow_{lm}^* uS\alpha \Rightarrow_{lm} \underbrace{a \cdots a}_k c = y$$

with $u = \alpha = \varepsilon$, $\beta = x$, and $\gamma = y$.

Then $k : x = k : y$, but $\beta \neq \gamma$. ■

c) Let G be any $LL(0)$ -grammar.

As $\forall x, y \in V_T^*, 0 : x = 0 : y \Leftrightarrow \varepsilon = \varepsilon$ and G is an $LL(0)$ -grammar, then $\forall u, x, y \in V_T^*$ and $\alpha, \beta, \gamma \in (V_T \cup V_N)^*$ with

$$S \Rightarrow_{lm}^* uY\alpha \Rightarrow_{lm} u\beta\alpha = ux, \text{ and}$$

$$S \Rightarrow_{lm}^* uY\alpha \Rightarrow_{lm} u\gamma\alpha = uy$$

implies $\beta = \gamma$.

\Rightarrow Each derivation step within G is unique.

\Rightarrow Because G is reduced, $|L(G)| = 1$.

\Rightarrow Each $LL(0)$ -language is a regular language.

As regular languages may contain more than one word, not every regular language is an $LL(0)$ -language.

$\Rightarrow LL(0)$ -languages \subsetneq Regular languages.

d) Let G be a left-recursive grammar with a production rule $A \rightarrow A\mu$. As all grammars are assumed to be reduced, A is a productive non-terminal and thus there exists a production rule $A \rightarrow \psi$. Additionally, there exist productions $\psi \Rightarrow_{lm}^* v \in V_T^*$ and $\mu \Rightarrow_{lm}^* w \in V_T^+$ (it is assumed that μ does not produce the empty word).

Let $k \in \mathbb{N}$ by arbitrary. Then G is not an $LL(k)$ -grammar, because A is reachable (G is reduced) and the following holds:

$$S \Rightarrow_{lm}^* uA\mu^n\alpha \Rightarrow_{lm} u \underbrace{\psi\mu^n}_{\beta} \alpha \Rightarrow_{lm}^* u \underbrace{vw^n}_x z$$

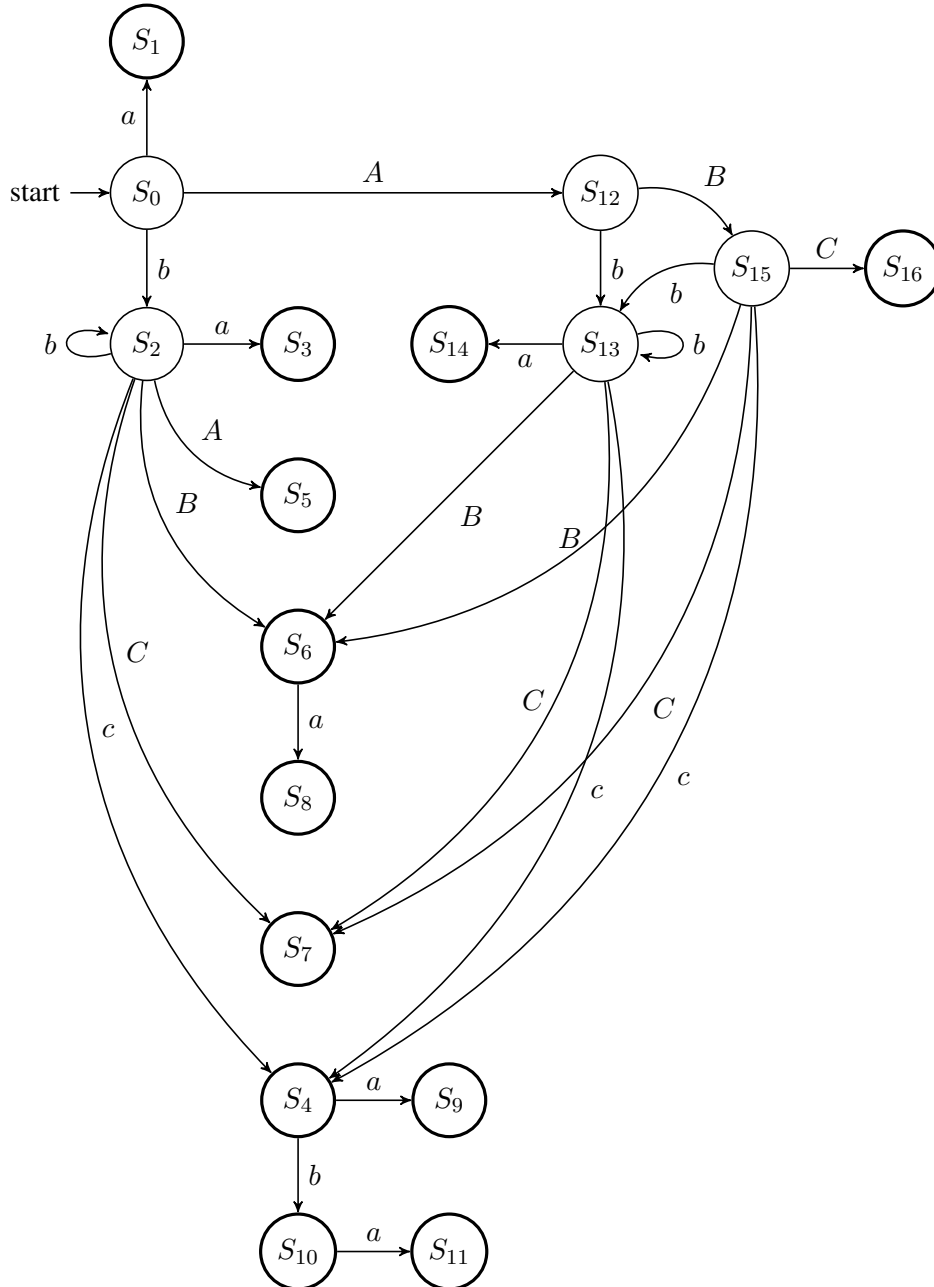
$$S \Rightarrow_{lm}^* uA\mu^n\alpha \Rightarrow_{lm} u \underbrace{A\mu^{n+1}}_{\gamma} \alpha \Rightarrow_{lm}^* u \underbrace{vw^{n+1}}_y z$$

with $\alpha \Rightarrow_{lm}^* z \in V_T^*$.

Now let $n \geq k$. Then obviously $k : x = k : y$, but $\beta \neq \gamma$. ■

2 Viable Prefixes

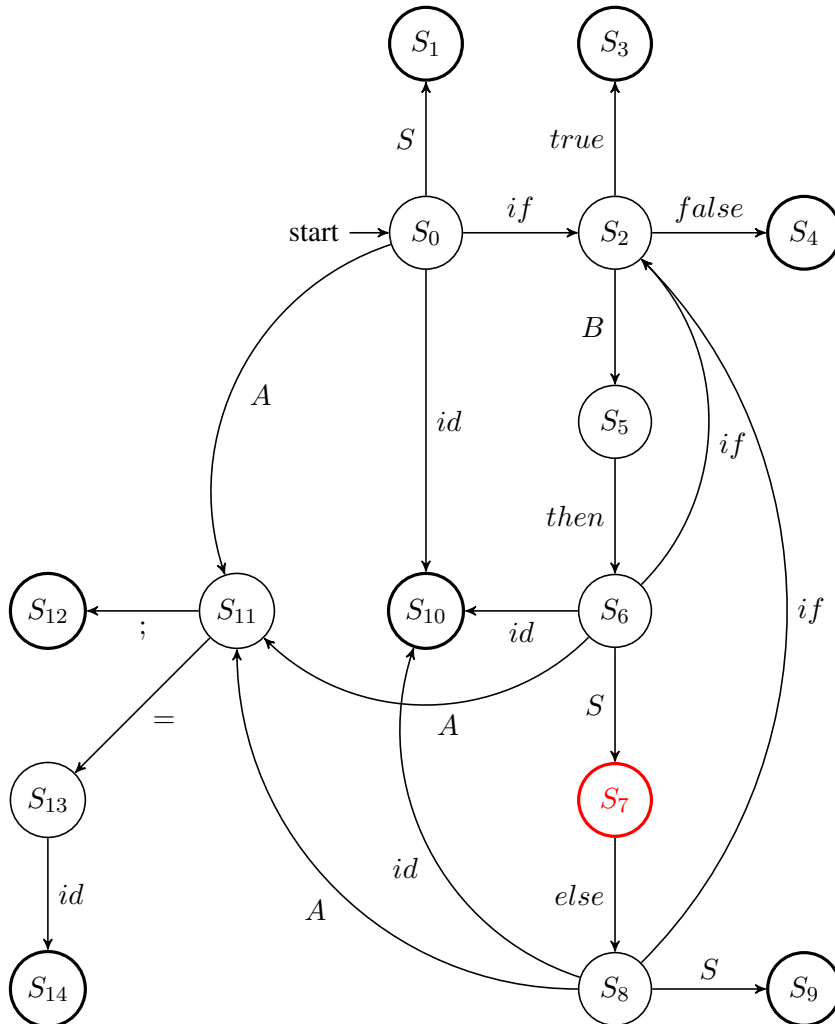
Whether the given words are viable prefixes can be easily read off the LR-DFA(G):



So, AbC and $ABbbbb$ are viable prefixes of G , because they are prefixes of words accepted by the LR-DFA(G). The other words are no viable prefixes.

3 LR(0)

a), b) LR-DFA(G) with one inadequate state S_7 :



where the states S_0 to S_{14} are defined as follows:

$$\begin{aligned}
 S_0 &= \left\{ \begin{array}{l} [S' \rightarrow .S], \\ [S \rightarrow .if\ B\ then\ S], \\ [S \rightarrow .if\ B\ then\ S\ else\ S], \\ [S \rightarrow .A;], \\ [A \rightarrow .A = id], \\ [A \rightarrow .id] \end{array} \right\} & S_8 &= \left\{ \begin{array}{l} [S \rightarrow if\ B\ then\ S\ else\ .S], \\ [S \rightarrow if\ B\ then\ S], \\ [S \rightarrow if\ B\ then\ S\ else\ S], \\ [S \rightarrow .A;], \\ [A \rightarrow .A = id], \\ [A \rightarrow .id] \end{array} \right\} \\
 S_1 &= \{ [S' \rightarrow S.] \} & S_9 &= \{ [S \rightarrow if\ B\ then\ S\ else\ S.] \} \\
 S_2 &= \left\{ \begin{array}{l} [S \rightarrow if\ .B\ then\ S], \\ [S \rightarrow if\ .B\ then\ S\ else\ S], \\ [B \rightarrow .true], \\ [B \rightarrow .false] \end{array} \right\} & S_{10} &= \{ [A \rightarrow id.] \} \\
 S_3 &= \{ [B \rightarrow true.] \} & S_{12} &= \{ [S \rightarrow A;.] \} \\
 S_4 &= \{ [B \rightarrow false.] \} & S_{11} &= \left\{ \begin{array}{l} [S \rightarrow A;.], \\ [A \rightarrow A. = id], \end{array} \right\} \\
 S_5 &= \left\{ \begin{array}{l} [S \rightarrow if\ B\ .then\ S], \\ [S \rightarrow if\ B\ .then\ S\ else\ S] \end{array} \right\} & S_{13} &= \{ [A \rightarrow A = .id], \} \\
 S_6 &= \left\{ \begin{array}{l} [S \rightarrow if\ B\ then\ .S], \\ [S \rightarrow if\ B\ then\ .S\ else\ S], \\ [S \rightarrow .if\ B\ then\ S], \\ [S \rightarrow .if\ B\ then\ S\ else\ S], \\ [S \rightarrow .A;], \\ [A \rightarrow .A = id], \\ [A \rightarrow .id] \end{array} \right\} & S_{14} &= \{ [A \rightarrow A = id.], \} \\
 S_7 &= \left\{ \begin{array}{l} [S \rightarrow if\ B\ then\ S.], \\ [S \rightarrow if\ B\ then\ S\ .else\ S] \end{array} \right\}
 \end{aligned}$$

c) Successful run of LR-DFA(G) for the input word $w = if\ true\ then\ id = id; else\ id;$:

Stack	Remaining Input	Action
S_0	$if\ true\ then\ id = id; else\ id;$	shift
S_0S_2	$true\ then\ id = id; else\ id;$	shift
$S_0S_2S_3$	$then\ id = id; else\ id;$	reduce $B \rightarrow true$
$S_0S_2S_5$	$then\ id = id; else\ id;$	shift
$S_0S_2S_5S_6$	$id = id; else\ id;$	shift
$S_0S_2S_5S_6S_{10}$	$= id; else\ id;$	reduce $A \rightarrow id$
$S_0S_2S_5S_6S_{11}$	$= id; else\ id;$	shift
$S_0S_2S_5S_6S_{11}S_{13}$	$id; else\ id;$	shift
$S_0S_2S_5S_6S_{11}S_{13}S_{14}$	$; else\ id;$	reduce $A \rightarrow A = id$
$S_0S_2S_5S_6S_{11}$	$; else\ id;$	shift
$S_0S_2S_5S_6S_{11}S_{12}$	$else\ id;$	reduce $S \rightarrow A;$
$S_0S_2S_5S_6S_7$	$else\ id;$	shift
$S_0S_2S_5S_6S_7S_8$	$id;$	shift
$S_0S_2S_5S_6S_7S_8S_{10}$	$;$	reduce $A \rightarrow id$
$S_0S_2S_5S_6S_7S_8S_{11}$	$;$	shift
$S_0S_2S_5S_6S_7S_8S_{11}S_{12}$	ϵ	reduce $S \rightarrow A;$
$S_0S_2S_5S_6S_7S_8S_9$	ϵ	reduce $S \rightarrow if\ B\ then\ S\ else\ S$
S_0S_1	ϵ	accept