Pontificia Universidad Católica de Chile
Departamento de Ciencia de la Computación
IIC2223 – Teoría de Autómatas y Lenguajes Formales
Segundo semestre de 2024
Profesor: Cristian Riveros
Ayudante: Amaranta Salas

Ayudantia 13

Apiladores

Problema 1

1. Demuestre un autómata apilador para el siguiente lenguaje:

$$L_1 = \{w \in \{0,1\}^* \mid 2 \cdot |w|_0 = |w|_1\}$$

dónde $|w|_a$ representa el número de a-letras en w. Explique su correctitud.

2. Para una palabra $w \in \{0,1\}^+$ sea $bin(w) \in \mathbb{N}$ el número natural que representa la palabra binaria w, donde los bits más significativos son los primeros. Demuestre un autómata apilador para el siguiente lenguaje:

$$L_2 = \{u\#v \mid u, v \in \{0,1\}^+ \text{ y } bin(v^r) = bin(u) + 1\}$$

donde v^r es el reverso de v (esto es, si $v = a_1 \dots a_n$, entonces $v_r = a_n a_{n-1} \dots a_1$). Explique su correctitud.

Problema 2

La notación polaca de una sentencia φ en lógica proposicional se define recursivamente como:

$$\begin{aligned}
\texttt{np}(0) &= 0 \\
\texttt{np}(1) &= 1 \\
\texttt{np}(\neg\varphi) &= \neg \cdot \texttt{np}(\varphi) \\
\texttt{np}(\varphi_1 \wedge \varphi_2) &= \wedge \cdot \texttt{np}(\varphi_1) \cdot \texttt{np}(\varphi_2) \\
\texttt{np}(\varphi_1 \vee \varphi_2) &= \vee \cdot \texttt{np}(\varphi_1) \cdot \texttt{np}(\varphi_2) \\
\texttt{np}([\varphi]) &= [\cdot \texttt{np}(\varphi) \cdot]
\end{aligned}$$

Donde $\cdot$ representa la concatenación y el alfabeto es $\Sigma = \{0, 1, \neg, \wedge, \vee, [,]\}$. Considere el lenguaje de todas las sentencias en lógica proposicional que se evalúan verdadero:

$$L = \{\texttt{np}(\varphi) | \varphi \text{ es una sentencia en lógica proposicional y } \varphi \equiv 1\}$$

Construya un autómata apilador alternativo para L y explique la correctitud de su construcción.

Problema 3

Sea $\mathcal{D} = (Q, \Sigma, \Delta, q_0, F)$ un autómata apilador alternativo. Sea R una expresión regular sobre Q, esto es, $\mathcal{L}(R) \subseteq Q^*$. Decimos que D *acepta una palabra* $w \in \Sigma^*$ *por* R*-stack* si existe una ejecución de $\mathcal{D}$ sobre w que empieza en la configuración inicial (q_0, w) y termina en una configuración (γ, ϵ) con $\gamma \in \mathcal{L}(R)$. Se define el lenguaje aceptado por R-stack como:

$$\mathcal{L}_R(\mathcal{D}) = \{w \in \Sigma^* \mid \mathcal{D} \text{ acepta } w \text{ por } R\text{-stack}\}.$$

En otras palabras, $\mathcal{D}$ acepta w por R-stack si existe una ejecución tal que el contenido del stack de la última configuración satisface la expresión regular R. Por ejemplo, si $F = \{p_1, \ldots, p_k\}$ y $R = p_1 + \cdots + p_k$ entonces es fácil ver que $\mathcal{L}(\mathcal{D}) = \mathcal{L}_R(\mathcal{D})$.
Demuestre que para todo autómata apilador alternativo $\mathcal{D}$ y para toda expresión regular R, existe un autómata apilador alternativo $\mathcal{D}'$ tal que $\mathcal{L}_R(\mathcal{D}) = \mathcal{L}(\mathcal{D}')$.