Pontificia Universidad Católica de Chile
Escuela de Ingeniería
Departamento de Ciencia de la Computación

IIC2223 — Teoría de Autómatas y Lenguajes Formales — 2' 2021

PAUTA TAREA 5

Pregunta 1

Un posible autómata apilador alternativo para este lenguaje es el siguiente. Sea $P = (Q, \Sigma, \Delta, q_0, F)$ donde:

$$\begin{aligned} \Sigma &= \{0, 1, \vee, \wedge, \neg, [,]\} \\ Q &= \Sigma \cup \{q_0, q_f\} \\ F &= \{q_f\} \end{aligned}$$

Y la relación de transición Δ se define de la siguiente manera:

$$\begin{aligned} \Delta \quad = \quad & \underbrace{\{(q, a, aq)) \mid q \in Q \wedge a \in \Sigma\}}_{\text{shift}} \cup \\ & \underbrace{\{(a_1 a_2 \wedge, \epsilon, a) \mid a_1, a_2 \in \{0,1\} \wedge a = [a_1 \wedge a_2]\}}_{\text{reduce}} \cup \\ & \underbrace{\{(a_1 a_2 \vee, \epsilon, a) \mid a_1, a_2 \in \{0,1\} \vee a = [a_1 \vee a_2]\}}_{\text{reduce}} \cup \\ & \underbrace{\{(a_1 \neg, \epsilon, a) \mid a_1 \in \{0,1\} \vee a = [\neg a_1]\}}_{\text{reduce}} \cup \\ & \underbrace{\{(]a_1[, \epsilon, a_1) \mid a_1 \in \{0,1\}\}}_{\text{reduce}} \cup \\ & \{((1q_0, \epsilon, q_f)\} \end{aligned}$$

La idea del autómata es aprovechar la estructura de la notación polaca y los conceptos de *shift* y *reduce* de los *Bottom-up PDA* para procesar los símbolos de cada sentencia de manera análoga a encontrar una derivación por la derecha de una palabra de una gramática. En este caso podemos ver a los 0 y 1 como las variables, donde cada una tiene una producción por cada combinación de valores y operadores que evalúan a 0 o 1 respectivamente.

Dado lo anterior, la distribución de puntaje es la siguiente:

- **(1 punto)** Por definir de manera correcta los estados del autómata.
- **(1 punto)** Por definir correctamente las transiciones de *shifting*.
- **(1 punto)** Por definir correctamente las transiciones de *reducing*.
- **(1 punto)** Por definir correctamente la transición final.

Pregunta 2

Sea G una gramática en forma normal de Chomsky. Definimos $\text{follow}_k(\gamma)$, como $\gamma \in (V \cup \Sigma)^*$, como

$$\text{follow}_k(\gamma) = \{v | S \Rightarrow^* \alpha X \beta \Rightarrow^* \alpha\gamma\beta \wedge v \in \text{first}_k(\beta\#)\}$$

Suponemos que G no tiene variables inútiles. Luego, podemos definir el $\text{follow}_k(\gamma)$ equivalentemente como:

$$\text{follow}_k(\gamma) = \bigcup_{X : X \Rightarrow^* \gamma} \text{follow}_k(X)$$

Luego de esto, podemos definir el algoritmo:

```
Function Follow_k(G, γ)
    V := CKY*_k(G, γ)
    O := ∅
    foreach X ∈ V do
        O := O ∪ follow_k(X)
    return O
```

Y definimos la función para encontrar los X:

```
Function CKY*_k(G, γ)
    for i := 1 to n do
        X_ii := ∅
        if γ_i ∈ Σ then
            foreach X → γ_i ∈ P do
                X_ii := X_ii ∪ {γ_i}
        else
            X_ii := {γ_i}
    for k := 1 to n do
        for i := 1 to n − k do
            X_{i,i+k} := ∅
            for j := i to i + k + 1 do
                foreach X → Y Z ∈ P do
                    if Y ∈ X_ij ∧ Z ∈ X_{j+1,i+k} then
                        X_{i,i+k} = X_{i,i+k} ∪ {X}
    return X_1n
```

Donde $\text{follow}_k(X)$ corresponde al algoritmo de follow visto en clases. Vemos que el algoritmo utiliza una versión modificada del algoritmo CKY para encontrar las letras que cumplen con la propiedad antes descrita, para luego retornar el conjunto de X que llevan en uno o más pasos a γ. Luego es trivial calcular el follow_k de esas variables con el algoritmo visto en clases.

Dado lo anterior, la distribución de puntaje es la siguiente:

- **(1 punto)** por encontrar la propiedad que une variables X con los γ
- **(1 punto)** por encontrar la variación del algoritmo CKY

- **(1 punto)** por utilizar el algoritmo CKY en la resolución del problema
- **(1 punto)** por el algoritmo general