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

IIC2223 — Teoría de Autómatas y Lenguajes Formales — 2' 2025
IIC2224 — Autómatas y Compiladores — 2' 2025

PAUTA TAREA 5

Pregunta 1

Vamos a demostrar que $L^{2\text{pal}}$ cumple la negación del lema de bombeo libre de contexto. P.D:

Para todo $N > 0$.
Existe la palabra $z = a^N b^N b^N a^N a^N b^N \in L^{2\text{pal}}$ con $|z| \geq N$ tal que
Para toda división $z = uvwxy$ con $v \cdot x \neq \epsilon$, $|vwx| \leq N$
Existe un $i \geq 0$ tal que
$u \cdot v^i \cdot w \cdot x^i \cdot y$ no pertenece a $L^{2\text{pal}}$.

Tomamos $i = 0$.
Sea $\alpha = uv^0wx^0y = uwy$. P.D: $\alpha \notin L^{2\text{pal}}$.

Supongamos que $\alpha \in L^{2\text{pal}}$. Esto significa que $\alpha = \phi_1\phi_2\phi_3$, con $\phi_1 = \phi_2^{rev} = \phi_3$.

Dado que $|vwx| \leq N$, tenemos que vwx pertenece a $\mathcal{L}(a^*b^*)$ o $\mathcal{L}(b^*a^*)$, cualquier otra opción necesitaría abarcar más de N letras. En particular, se cumple el siguiente lema:
Si w_1 es el primer tercio de z, w_2 el segundo y w_3 el tercero, vwx puede abarcar a lo más 2 de ellos.
Por lo tanto, puede ocurrir uno de los siguientes 3 casos:

Caso 1: $vwx \in \mathcal{L}(a^*)$ (análogo a $vwx \in \mathcal{L}(b^*)$).
Como $vx \neq \epsilon$, tenemos que se eliminó al menos una a. Además, por el lema, se puede afirmar lo siguiente:
Entre ϕ_1, ϕ_2 y ϕ_3, hay al menos uno de ellos que tiene una menor cantidad de a's que otro, lo cual no puede ocurrir. Por contradicción, en este caso $\alpha \notin L^{2\text{pal}}$.

Caso 2: $vwx \in \mathcal{L}(a^+b^+)$.
Esto sólo ocurre cuando vwx está en el primer o último tercio de z.
Como se eliminan al menos una a y/o al menos una b, se sigue la misma lógica que el caso 1:
Si se eliminó al menos una a, por el lema, entre ϕ_1, ϕ_2 y ϕ_3, hay al menos uno con menos a's que el otro.
Si se eliminó al menos una b, entre ϕ_1, ϕ_2 y ϕ_3, hay al menos uno con menos b's que otro.
Ninguno de estos puede ocurrir, pues todos deberían ser iguales. Por contradicción, $\alpha \notin L^{2\text{pal}}$

Caso 3: $vwx \in \mathcal{L}(b^+a^+)$.
Esto sólo ocurre cuando vwx está en el segundo tercio de z.
Se eliminan al menos una b y/o al menos una a. Como vimos antes, se llega a una contradicción.
Por lo tanto $\alpha \notin L^{2\text{pal}}$.

En todos los casos posibles, α no pertenece a $L^{2\text{pal}}$. Por lo tanto, $L^{2\text{pal}}$ no es regular.

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

- **(1 punto)** Por definir una palabra tal que el diablo no pueda ganar
- **(1 punto)** Por casos donde vwx tenga una única letra (**Caso 1** en esta solución)
- **(1 punto)** Por casos donde vwx sea parte del tercio inicial o final de z (**Caso 2** en esta solución)
- **(1 punto)** Por casos donde vwx sea parte del tercio del medio de z (**Caso 3** en esta solución)

Pregunta 2

Para resolver el problema **Max-Var** utilizaremos el algoritmo CKY adaptado.
Sea $w = a_1 a_2 \dots a_n$ la palabra de entrada y $\mathcal{G} = (V, \Sigma, P, S)$ la gramática en CNF. Definimos el conjunto $C_{ij} = \{A \in V \mid A \Rightarrow^* a_i \dots a_j\}$ al igual que en clases, que corresponde a las variables que pueden derivar la subpalabra $a_i \dots a_j$. Durante el algoritmo, para todo $i < j \leq n$ mantendremos las funciones $H_{ij} : V \to \mathbb{N} \cup \{-\infty\}$ (esto es, una tabla de Hash $H_{i,j}$), donde $H_{ij}(A)$ es el número máximo de veces que aparece X sobre todos los árbol de derivación con raíz A para $a_i \dots a_j$. Notar que $\text{dom}(H_{ij}) = \text{keys}(H_{ij}) = C_{ij}$. Para simplificar el algoritmo, si no hay una derivación desde A (esto es, $A \notin C_{ij}$), entonces supondremos que $H_{ij}(A) = -\infty$. De hecho, asumiremos que inicialmente $H_{ij}(A) = -\infty$ para todo $i < j < n$ y $A \in V$. Supondremos también que $-\infty + c = -\infty$ para todo valor $c \in \mathbb{N}$.

Input: Gramática $\mathcal{G}$ en CNF, palabra $w = a_1 \dots a_n$, variable X
Output: $\max\{\text{count}_X(t) \mid t \in \text{der}_{\mathcal{G}}(w)\}$

$n \leftarrow |w|$
for $i \leftarrow 1$ **to** n **do**
  for $A \to a \in P$ **do**
    if $a = a_i$ **then**
      if $A = X$ **then**
        $H_{i,i}[A] \leftarrow 1$
      else
        $H_{i,i}[A] \leftarrow 0$
for $k \leftarrow 1$ **to** $n - 1$ **do**
  for $i \leftarrow 1$ **to** $n - k$ **do**
    for $j \leftarrow i$ **to** $i + k - 1$ **do**
      foreach $A \to BC \in P$ **do**
        if $A = X$ **then**
          $H_{i,i+k}[A] \leftarrow \max\{\, H_{i,i+k}[A],\ H_{i,j}[B] + H_{j+1,i+k}[C] + 1 \,\}$
        else
          $H_{i,i+k}[A] \leftarrow \max\{\, H_{i,i+k}[A],\ H_{i,j}[B] + H_{j+1,i+k}[C] \}$
return $H_{1,n}[S]$

Algorithm 1: Algoritmo Max-Var

Correctitud: La correctitud se demuestra por inducción sobre la longitud de las subcadenas.

- *Caso Base:* Para $A \to a_i$, el árbol tiene un solo nodo (la variable A). El conteo es 1 si $A = X$ y 0 en caso contrario. El algoritmo llena correctamente H_{ii}.
- *Hipótesis de Inducción:* Asumimos que para toda subcadena de longitud menor a L, los conjuntos C (keys) y los valores H están correctamente calculados.
- *Paso Inductivo:* Para una subcadena de longitud $L > 1$ generada por $A \to BC$, el número de X's es la suma de las X's en el subárbol izquierdo (B), el subárbol derecho (C) y la propia raíz A (si $A = X$). Al iterar sobre todos los posibles puntos de corte k y verificar $B \in C_{ik} \wedge C \in C_{k+1,j}$ (usando la H.I.), garantizamos encontrar la derivación válida que maximiza el conteo.

Complejidad: El algoritmo tiene tres for anidados que dependen de n (longitud, inicio, corte), lo que da un factor $O(n^3)$. Dentro del for más interno, se itera sobre las producciones de la gramática (P). Por lo tanto, el tiempo de ejecución es $O(|w|^3 \cdot |\mathcal{G}|)$.

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

- **(1 puntos)** Por definir correctamente H_{ij} y su relación con el conjunto de variables derivables C_{ij} (keys).
- **(1 puntos)** Por la correcta inicialización y cálculo del caso base, llenando H_{ii} y calculando el valor si $A = X$. Y por identificar el output correcto.
- **(1 puntos)** Por el paso recursivo correcto, verificando pertenencia a los conjuntos $B \in C_{ij}, C \in C_{j+1,i+k}$) y aplicando la maximización: $H_{i,j}[B] + H_{j+1,i+k}[C] + (1_{A=X})$.
- **(1 puntos)** Por demostrar la correctitud del algoritmo.