NOMBRE: Benjamín Farías Valdés

N.ALUMNO: 17642531

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

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

Tarea 4

Pregunta 1

1)

La gramática libre de contexto propuesta es la siguiente:

$$G: \quad S \longrightarrow aAaC \mid bAbC \mid CaAa \mid CbAb \mid aBa \mid bBb \mid aC \mid bC \mid aAb \mid bAa$$
$$A \longrightarrow \epsilon \mid aAa \mid bAb \mid a \mid b$$
$$B \longrightarrow \epsilon \mid aAb \mid bAa \mid aBa \mid bBb$$
$$C \longrightarrow \epsilon \mid a \mid b$$

2)

(No se alcanzó a demostrar formalmente)
Se debe demostrar que $\mathcal{L}(G) = L$. Esto se hará con la contención hacia ambos lados:

- $\mathcal{L}(G) \subseteq L$**:** La idea es que las reglas de S de la forma xC o Cx se encargan de generar todos los posibles palíndromos de largo par o impar (según lo visto en clase), los que al borrarles una letra central (cualquiera de las 2 en el caso de los pares) siempre seguirán siendo palíndromos. Esto significa que todos los palíndromos son también casi palíndromos. La variable C se encarga de añadir una letra cualquiera a algún extremo del palíndromo (de forma opcional), ya que en este caso se obtienen todos los casi palíndromos tales que al borrarles esa letra se cumple la propiedad.

  Por otro lado, las reglas de S que no contienen C se encargan de generar palíndromos con la posibilidad de equivocarse una vez, dejando un par de letras que no son iguales al momento de construir la palabra hacia adentro. Estos tipos de casi palíndromos son los que aplican la operación de cambiar una de esas 2 letras para que queden iguales y se satisfaga la propiedad.

  Luego, todos las palabras derivadas de S cumplen con pertenecer a L.

- $L \subseteq \mathcal{L}(G)$**:** Todo casi palíndromo se satisface mediante las operaciones de eliminación o intercambio. Por lo tanto, es posible generarlo con las reglas ya explicadas.

NOMBRE: Benjamín Farías Valdés

N.ALUMNO: 17642531

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

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

Tarea 4

Pregunta 2

1)

El algoritmo propuesto es el siguiente:

Algorithm 1 *Reachable*(V, E, S, X)

```
C = {S}
D = ∅
if X = S then
  return TRUE
end if
while C ≠ ∅ do
  take Y ∈ C
  C = C − {Y}
  D = D ∪ {Y}
  for each (A, B) ∈ E where A = Y do
    if B = X then
      return TRUE
    end if
    if B ∉ D then
      C = C ∪ {B}
    end if
  end for
end while
return FALSE
```

Para demostrar la correctitud de este algoritmo es necesario probar que entregará **TRUE** si y solo si X es alcanzable en G.

El algoritmo explora todas las variables que son alcanzables a partir de la variable inicial S, chequeando en cada una si es que corresponde a X, en cuyo caso retorna **TRUE**. Esto se desprende de que primero chequea a S en la condición del principio, y luego entra en un loop donde revisa todos los vértices $B \in V$ tales que $(A, B) \in E$, cambiando el vértice A en cada iteración y agregando los B a un conjunto de vértices que serán visitados posteriormente. Podemos aplicar inducción para demostrar que todos los vértices visitados representan a variables B alcanzables en G:

- **Caso Base:** Si $B = S$, entonces se cumple trivialmente que B es alcanzable por la propiedad refleja de la clausura de derivación ($S \overset{*}{\Rightarrow} S$).
- **Caso Inductivo:** Si B es alcanzable, entonces el algoritmo revisará todos aquellos vértices $B' \in V$ tales que $(B, B') \in E$. Por definición de aristas del enunciado, esto significa que existen $\alpha, \beta \in (V \cup \Sigma)^*$ tal que $B \to \alpha B' \beta \in P$. Por la propiedad vista en clases, tenemos entonces que los B' son alcanzables, ya que B es alcanzable y existe la producción $B \to \alpha B' \beta \in P$.

Luego, si el algoritmo retorna **TRUE**, significa que alguno de los vértices B revisados era justamente la variable X. Como demostramos que estos B son siempre alcanzables, tenemos que X también lo es. Por otro lado, si X no es alcanzable, entonces no se corresponderá con ningún B revisado por el algoritmo, ya que generaría una contradicción al ser alcanzable y no alcanzable a la vez, y por lo tanto saldrá del loop y retornará **FALSE**. Finalmente, llegamos a que si X es alcanzable el algoritmo retornará **TRUE**, y si no es alcanzable retornará **FALSE**, por lo que el algoritmo es correcto.

2)

(No se alcanzó a demostrar)

La idea es que se pueden tomar 2 gramáticas que cumplan con las reglas que definen al grafo, pero que en una la variable X sea generadora y en la otra no, llegando a una contradicción.