1. Resumen

El problema de la satisfacibilidad booleana es un problema NP-Completo y consiste en determinar si existe alguna interpretación verdadera de una fórmula booleana dada. La teoría de lenguajes es una rama fundamental de la Ciencia de la Computación y la matemática que se enfoca en el estudio de los lenguajes formales.

Las gramáticas de concatenación de rango son un formalismo de gramáticas desarrollado en 1988 como una propuesta de Pierre Boullier, un investigador en el campo de la lingüística computacional. Su objetivo principal era proporcionar un modelo más general y expresivo que las gramáticas libres del contexto para describir lenguajes.

El objetivo de este trabajo es vincular el problema de la satisfacibilidad booleana y la teoría de lenguajes, utilizando las gramáticas de concatenación de rango para construir el lenguaje de todas las fórmulas booleanas satisfacibles. Para esto primeramente se describe como codificar una fórmula booleana cualquiera en una cadena, se define el lenguaje de todas las fórmulas booleanas satisfacibles y para terminar se construye una gramática de concatenación de rango que reconoce este lenguaje.

La construcción de la gramática de concatenación de rango que reconoce el lenguaje de todas las fórmulas booleanas satisfacibles permite demostrar que las gramáticas de concatenación de rango reconocen todos los problemas de la clase NP y que el problema de la palabra para estas gramáticas es NP-Duro.

2. Introducción

3. Gramáticas de Concatenación de Rango

Las gramáticas de concatenación de rango (*RCG*) [1] son un formalismo de gramáticas desarrollado en 1988 como una propuesta de Pierre Boullier, un investigador en el campo de la lingüística computacional. Su objetivo principal era proporcionar un modelo más general y expresivo que las gramáticas libres del contexto para describir lenguajes. Las RCG fueron diseñadas con el fin de analizar propiedades y características del lenguaje natural, como los números chinos y el orden aleatorio de algunas palabras alemanas [11].

3.1. Definiciones

En esta sección se define el concepto de rango, sustitución de rango, gramática de concatenación de rango [1] y gramática de concatenación de rango simple.

Definición 1. *Un* ***rango*** *es una tupla* (i, j) *que representa un intervalo de posiciones en una cadena, donde* i *y* j *son enteros no negativos tales que* $i \leq j$.

Por ejemplo, si los índices son indexados en 0, para la cadena *abcd*, el rango $(1, 2)$, representa la subcadena *bc*.

[1] En la literatura este tipo de RCG se toma como gramática de concatenación de rango positiva, pero como es la única que se usa en este trabajo se le llama solo gramática de concatenación de rango.

Definición 2. *Una **gramática de concatenación de rango** se define como una 5-tupla:*

$$G = (N, T, V, P, S),$$

donde:

- N*: Es un conjunto finito de **predicados o símbolos no terminales**: Cada predicado tiene una **aridad**, que indica la dimensión del vector de cadenas que reconoce y cada cadena del vector se asocia a un argumento del predicado.*
- T*: Es un conjunto finito de **símbolos terminales**.*
- V*: Es un conjunto finito de **variables**.*
- P*: Es un conjunto finito de **cláusulas**, de la forma:*

  $$A(x_1, x_2, \ldots, x_k) \rightarrow B_1(y_{1,1}, y_{1,2}, \ldots, y_{1,m_1}) \ldots B_n(y_{n,1}, y_{n,2}, \ldots, y_{n,m_n}),$$

  donde $A, B_i \in N$, $x_i, y_{i,j} \in (V \cup T)^*$, y k es la aridad de A.
- $S \in N$*: Es el **predicado inicial** de la gramática, que siempre tiene **aridad** 1.*

Por ejemplo, a continuación se muestra una gramática de concatenación de rango:

$$G^3_{copy} = (N, T, V, P, S),$$

donde:

- N=$\{A, S\}$.
- T=$\{a, b, c\}$.
- V=$\{X, Y, Z\}$.
- El conjunto de cláusulas P es el siguiente:
  1. $S(XYZ) \rightarrow A(X, Y, Z)$
  2. $A(aX, aY, aZ) \rightarrow A(X, Y, Z)$
  3. $A(bX, bY, bZ) \rightarrow A(X, Y, Z)$
  4. $A(cX, cY, cZ) \rightarrow A(X, Y, Z)$
  5. $A(\varepsilon, \varepsilon, \varepsilon) \rightarrow \varepsilon$
- El símbolo inicial es S.

Las RCG, a diferencia de las gramáticas convencionales no generan cadenas, su funcionamiento se basa en reconocer si una cadena pertenece o no al lenguaje.

Definición 3. *Una **sustitución de rango** es un mecanismo que reemplaza una variable por un rango de la cadena, respetando la estructura del argumento que se asocia a la cadena que se reconoce.*

Por ejemplo, dado el predicado $A(Xa)$ donde $X \in V$ y $a \in T$, la estructura del argumento de A es una variable X seguida del terminal a. Si el no terminal A recibe la cadena baa, X se puede asociar con el rango ba de la cadena original porque si $X = ba$, entonces $Xa = baa$.

Por otro lado, la variable X no puede tomar el valor baa, porque ningún caracter de la cadena de entrada coincidiría con el terminal a. De manera similar, X tampoco puede tomar el valor b porque el valor que se asigna a X no permite que el argumento Xa cubra la cadena completa.

En la próxima sección se describe el proceso de derivación de las RCG.

3.2. Proceso de derivación

La idea principal para realizar una derivación en la cláusula

$$A(x_1, x_2, \ldots, x_k) \rightarrow B_1(y_{1,1}, y_{1,2}, \ldots, y_{1,m_1}) \ldots B_n(y_{n,1}, y_{n,2}, \ldots, y_{n,m_n}),$$

de una RCG, se basa en tomar el vector de cadenas $[w_1, w_2, \ldots, w_k]$ que recibe el predicado A y asociar cada elemento del vector al argumento correspondiente: w_1 se asocia al argumento x_1, w_2 se asocia al argumento x_2 y así hasta que w_k se asocia a x_k.

Después de asociar los elementos del vector a los argumentos, se realizan todas las posibles sustituciones de rango para cada argumento y se asocia un rango a cada variable del predicado izquierdo.

A partir de los valores de las variables obtenidos en el paso anterior se construyen vectores de cadenas con los que se instancian las variables de los predicados del lado derecho de la cláusula.

A modo de ejemplo se puede considerar la producción $A(X, aYb) \rightarrow B(aXb, Y)$, donde X e Y son variables y a y b son símbolos terminales.

Cuando A recibe el vector $[a, abb]$, el primer argumento de A recibe a y el segundo recibe abb. El primer argumento de A es X y el segundo es aYb, por lo que $X = a$ y $aYb = abb$. En este caso la única sustitución de rango posible es $X = a$ y $Y = b$. Con estos valores se construyen los vectores con los que se instancia la parte derecha, que serían $aXb = aab$ y Y=b. Con este vector el predicado B se instancia como $B(aab, b)$, y por tanto, el predicado $A(a, abb)$ deriva como $B(aab, b)$.

Un vector de cadenas se reconoce por un predicado A si existe una secuencia de derivaciones que comienza en A y termina en la cadena vacía.

Por ejemplo, dada la cláusula $A(X_1, X_2, X_3) \rightarrow B_1(X_1)B_2(X_2)B_3(X_3)$, el vector $[w_1, w_2, w_3]$ se reconoce por A, si existe una secuencia de derivaciones para cada uno de los predicados $B_1(w_1)$, $B_2(w_2)$, $B_3(w_3)$ que derive en la cadena vacía.

A continuación se presenta un ejemplo de reconocimiento de la cadena $abcabcabc$ por la gramática G^3_{copy} presentada en la página 2.

La cadena $abcabcabc$ se reconoce por G^3_{copy}, ya que $S(abcabcabc)$ se puede derivar de la siguiente manera:

$$S(abcabcabc) \rightarrow A(abc, abc, abc) \rightarrow A(bc, bc, bc) \rightarrow A(c, c, c) \rightarrow A(\varepsilon, \varepsilon, \varepsilon) \rightarrow \varepsilon.$$

A continuación se muestran estas derivaciones paso a paso.

- **Primer paso:** En la primera cláusula $S(XYZ) \rightarrow A(X, Y, Z)$, existe una sustitución de rango que asocia las variables X, Y, Z a los valores $X = abc$, $Y = abc$ y $Z = abc$. De esta forma se deriva en el predicado $A(abc, abc, abc)$.

- **Segundo paso:** En la segunda cláusula $A(aX, aY, aZ) \rightarrow A(X, Y, Z)$, existe una sustitución de rango que asocia las variables X, Y, Z a los valores $X = bc$, $Y = bc$ y $Z = bc$. Con estos valores se deriva en el predicado $A(bc, bc, bc)$.

- **Tercer paso:** En la tercera cláusula $A(bX, bY, bZ) \rightarrow A(X, Y, Z)$, existe una sustitución de rango que asocia las variables X, Y, Z a los valores $X = c$, $Y = c$ y $Z = c$. Con estos valores se deriva en el predicado $A(c, c, c)$.

- **Cuarto paso:** En la cuarta cláusula $A(cX, cY, cZ) \rightarrow A(\varepsilon, \varepsilon, \varepsilon)$, existe una sustitución de rango que asocia las variables X, Y, Z a los valores $X = \varepsilon$, $Y = \varepsilon$ y $Z = \varepsilon$. Con estos valores se deriva en el predicado $A(\varepsilon, \varepsilon, \varepsilon)$.

- **Quinto paso:** Finalmente en el último paso se toma la última cláusula $A(\varepsilon, \varepsilon, \varepsilon) \rightarrow \varepsilon$ que deriva en la cadena vacía, por lo que de esta manera se reconoce la cadena *abcabcabc*.

A continuación se presentan algunas propiedades de las RCG relevantes para este trabajo.

3.3. Propiedades de las RCG

La motivación fundamental detrás de la creación de las RCG fue crear un formalismo modular. Esto significa que las principales operaciones sobre conjuntos: unión, intersección y complemento son cerradas para dicho formalismo [1]. Esta es precisamente la limitación de las CFG que se propone suplir con las RCG para el procesamiento del lenguaje natural [1].

En esta sección se describen las principales propiedades que demuestran que las RCG son un formalismo modular, además se presenta el problema de la palabra para las RCG, el cual se emplea en la sección 5 para determinar si una fórmula booleana es satisfacible.

Teorema 1. *Las RCG* ***son cerradas bajo unión, intersección y complemento****, la unión y la intersección de 2 RCG da como resultado un formalismo que pertenece a las RCG, mientras que el formalismo resultante del complemento de una RCG es también una RCG.*

Teorema 2. *Las RCG reconocen todos los problemas de la clase P.*

La demostración de los Teoremas 1 y 2 se realiza en [1].

En [1] se menciona que en la mayoría de los casos el problema de la palabra para las RCG es polinomial y se resuelve mediante un algoritmo de memorización sobre las cadenas asignadas a los argumentos de los predicados de la RCG. Como la cantidad máxima de rangos de la cadena es n^2 y la máxima aridad de un predicado es constante, este proceso de memorización cuenta con una cantidad polinomial de estados, y tiene una complejidad de $O(|P|n^{2h(l+1)})$ donde h es la máxima aridad en un predicado, l es la máxima cantidad de predicados en el lado derecho de una cláusula y n es la longitud de la cadena que se reconoce.

Sin embargo, existen casos en los que el problema de la palabra no es polinomial. En la siguiente sección se analiza un caso en el que este problema no es polinomial.

3.3.1. Problema de la palabra no polinomial para las RCG

El algoritmo de reconocimiento que se menciona en la sección anterior utiliza un proceso de memorización sobre los rangos de la cadena. La idea fundamental para esto y lo que acota la complejidad del algoritmo es que la cantidad de estados asociados a la memorización es igual a

la cantidad de rangos de la cadena, el cual es polinomial con respecto a la longitud de la cadena. Esto se cumple siempre que todos los argumentos que reciben todos los no terminales de la gramática sean subcadenas de la cadena original que se está analizando. Existen gramáticas de concatenación de rango en que esto no ocurre, como en la que se muestra a continuación.

La siguiente RCG reconoce el lenguaje $L = \{w \mid w \in \{0,1\}^*\}$. Esta gramática de concatenación de rango no tiene uso real porque existen otras RCG que reconocen el mismo lenguaje, pero ilustra una RCG donde se generan cadenas que no son subcadenas de la cadena de entrada durante el proceso de reconocimiento.

$$G_e = (N, T, V, P, S),$$

donde:

- N=$\{A, B, Eq, S\}$.
- T=$\{0, 1\}$.
- V=$\{X, Y\}$.
- El conjunto de cláusulas P es el siguiente:
  1. $S(X) \rightarrow A(X, X)$
  2. $A(1X, Y) \rightarrow B(X, 0, Y)$
  3. $A(1X, Y) \rightarrow B(X, 1, Y)$
  4. $A(0X, Y) \rightarrow B(X, 1, Y)$
  5. $A(0X, Y) \rightarrow B(X, 0, Y)$
  6. $B(1X, Y, Z) \rightarrow B(X, 1Y, Z)$
  7. $B(1X, Y, Z) \rightarrow B(X, 0Y, Z)$
  8. $B(0X, Y, Z) \rightarrow B(X, 0Y, Z)$
  9. $B(0X, Y, Z) \rightarrow B(X, 1Y, Z)$
  10. $B(\varepsilon, Y, Z) \rightarrow Eq(Y, Z)$
- El símbolo inicial es S.

Para procesar una cadena w, la gramática anterior genera todas las posibles cadenas q, tales que $|w| = |q|$ y luego comprueba si $w = q$.

Esta gramática no tiene caso de uso, ya que para toda cadena w siempre va a existir una cadena q tal que $w = q$, por lo que se puede modelar con solamente la cláusula $S(X) \rightarrow \varepsilon$. Sin embargo, la complejidad del reconocimiento de G es mayor que 2^n (con n igual al tamaño de la cadena de entrada), ya que esta es la cantidad de cadenas posibles que puede recibir el segundo argumento del predicado B, porque la gramática es ambigua y en cada derivación de B existen 2 posibles decisiones, se añade un 1 delante al valor de la Y o se añade un 0.

En la próxima sección se muestra como codificar una fórmula booleana como una cadena y se definen el lenguaje de todas las fórmulas booleanas satisfacibles.

4. Codificación de una fórmula booleana a una cadena

Una fórmula booleana F, con v variables en CNF tiene la siguiente estructura:

$$F = X_1 \wedge X_2 \wedge \ldots \wedge X_n$$

donde cada cláusula X_i es una disyunción de literales

$$X_i = L_{i1} \vee L_{i2} \vee \ldots \vee L_{im},$$

cada literal L_{ij} es una variable booleana o su negación. También se asume que $m \leq v$.

Si se tiene una fórmula booleana F en forma normal conjuntiva se puede considerar que cada una de las v variables aparece en cada cláusula en uno de tres posibles estados: sin negar, negada, o no aparece.

Por ejemplo, en la primera cláusula de la siguiente fórmula booleana en CNF con 3 variables:

$$F = (x_1 \vee \neg x_2) \wedge (\neg x_1 \vee x_2 \vee x_3) \wedge (x_1 \vee \neg x_2 \vee x_3)$$

la variable x_1 aparece sin negar, la variable x_2 aparece negada, y la variable x_3 no aparece.

El hecho de que se pueda asumir que en todas las cláusulas aparecen todas variables permite representar una cláusula de una fórmula con v variables como una cadena de v símbolos, donde el símbolo en la posición i indica el estado de la variable x_i en la cláusula.

En este trabajo se propone usar los símbolos a, b y c para indicar el estado de una variable en una cláusula, usando el siguiente convenio:

- a: indica que la variable aparece sin negar,
- b: indica que la variable aparece negada,
- c: indica que la variable no aparece.

Con este convenio, la primera cláusula de F se puede representar mediante la cadena abc.

Una vez que se tiene cómo representar una cláusula es posible representar varias cláusulas usando otro símbolo como separador. En este trabajo se propone usar d para indicar el final de una cláusula. De esta forma, una fórmula lógica con v variables y k cláusulas se puede representar mediante k bloques de longitud v, donde cada bloque está formado por los símbolos a, b, o c, y cada bloque se separa del siguiente por el símbolo d.

Con este convenio, la fórmula

$$F = (x_1 \vee \neg x_2) \wedge (\neg x_1 \vee x_2 \vee x_3) \wedge (x_1 \vee \neg x_2 \vee x_3)$$

se representa mediante la cadena:

$$abc\mathbf{d}baa\mathbf{d}aba\mathbf{d},$$

donde los símbolos **d** aparecen en negrita para facilitar la interpretación de la cadena como fórmula en forma normal conjuntiva.

Para que una cadena e se pueda interpretar como una fórmula booleana debe cumplir con las siguientes condiciones: tener n bloques separados por d, cada bloque de la misma longitud v y cada bloque solo debe estar formado por los caracteres a, b y c.

Una cadena e que cumpla con estas características se puede interpretar como una fórmula booleana con n cláusulas y v variables, donde la estructura de cada cláusula depende de los caracteres correspondientes al bloque de a, b y c que se asocia a dicha cláusula.

Por ejemplo, la cadena $w = acc\mathbf{d}aba\mathbf{d}cba\mathbf{d}$, tiene 3 bloques separados por d, los cuales son acc, aba y cba, los 3 tienen tamaño 3 y solo tienen los caracteres a, b y c. Por tanto w se puede interpretar como la siguiente fórmula booleana:

$$(x_1) \wedge (x_1 \vee \neg x_2 \vee x_3) \wedge (\neg x_2 \vee x_3).$$

Una vez definida la transformación de una fórmula booleana en una cadena, se puede definir el lenguaje de todas las fórmulas booleanas en CNF.

Definición 4. *El **lenguaje de todas las fórmulas booleanas en CNF** se define como:*

$$L_{FULL-SAT} = \{q_1 d q_2 d \dots q_n d \mid q_i \in \{a,b,c\}^+,\ |q_i| = |q_j|\ \forall i,j = 1 \dots n,\ \ y\ n \in \mathbb{N}\}.$$

A partir de $L_{FULL-SAT}$ se puede definir el lenguaje de todas las fórmulas booleanas satisfacibles en CNF, el cual se define a continuación.

Definición 5. *El **lenguaje de todas las fórmulas booleanas en CNF que son satisfacibles** L_{S-SAT} se define como todas las cadenas $e \in L_{FULL-SAT}$, tales que la fórmula booleana que representa e, sea satisfacible.*

Por ejemplo, la fórmula

$$x_1 \wedge x_2 \wedge x_3,$$

es satisfacible por los valores $x_1 = true$, $x_2 = true$ y $x_3 = true$, por lo que la cadena $acc\mathbf{d}cac\mathbf{d}cca\mathbf{d}$ pertenece a L_{S-SAT}. Por otro lado, la fórmula

$$x_1 \wedge x_2 \wedge \neg x_1,$$

no es satisfacible para ninguna asignación de los valores de sus variables, por lo que la cadena $ac\mathbf{d}ca\mathbf{d}bc\mathbf{d}$ no pertenece a L_{S-SAT}.

En la próxima sección se muestra como construir una RCG que reconoce el lenguaje L_{S-SAT}.

5. Construcción de L_{S-SAT} mediante una RCG

En esta sección se presenta una RCG que reconoce las fórmulas booleanas satisfacibles. Esto permite demostrar que las RCG reconocen todos los problemas de la clase NP, en su representación como lenguaje formal.

La idea para reconocer las fórmulas satisfacibles tiene dos partes: mientras se reconoce la primera cláusula se generan todas las posibles interpretaciones de la variable que la hacen verdadera, y después, en la segunda parte, se comprueba si alguna de estas interpretaciones satisface al resto de las cláusulas.

Para definir la gramática, sus producciones se agrupan en 4 grupos, en dependencia de las tareas que cumplen durante el reconocimiento. A cada uno de estos grupos se les llamará *fase*. A continuación se describe qué función cumplen las producciones de cada fase.

- **Primera fase:** representa la derivación inicial de la gramática.

- **Segunda fase:** se encarga de generar todas las posibles cadenas de 0 y 1 que representan interpretaciones de las variables que satisfacen la primera cláusula. En esta fase se definen 2 estados: positivo (significa que la cadena de 0 y 1 generada ya satisface la primera cláusula) y negativo (significa que la cadena de 0 y 1 generada aún no satisface la primera cláusula). Estos estados se representan por los predicados P y N, respectivamente.
- **Tercera fase:** comprueba que la interpretación que se define en la fase anterior satisfaga el resto de las cláusulas.
- **Cuarta fase:** define el algoritmo para determinar si una interpretación satisface una cláusula dada. En esta fase se definen 2 estados: positivo (significa que la interpretación ya satisface la cláusula actual) y negativo (significa que la interpretación aún no satisface la cláusula actual). Estos estados se representan por los predicados Cp y Cn respectivamente.

Seguidamente, se define la siguiente RCG que reconoce el lenguaje L_{S-SAT}:

$$G_{S-SAT} = (N, T, V, P, S),$$

donde:

- $N = \{S, A, B, C, P, N, Cp, Cn\}$
- $T = \{a, b, c, d\}$.
- $V = \{X, Y, X_1, X_2\}$.
- El **símbolo inicial** es S.

A continuación se desglosa el conjunto de **cláusulas** P de acuerdo a las fases descritas.

- **Primera fase:** Representa la cláusula de derivación inicial de la gramática:

  1. $S(X) \to A(X)$.

- **Segunda fase:** El siguiente conjunto de cláusulas genera una cadena de 0 y 1 que cuando se le asigna a las variables de la fórmula booleana, satisface la primera cláusula.

  2. $A(aX) \to P(X, 1)$
  3. $A(aX) \to N(X, 0)$
  4. $A(bX) \to N(X, 1)$
  5. $A(bX) \to P(X, 0)$
  6. $A(cX) \to N(X, 1)$
  7. $A(cX) \to N(X, 0)$
  8. $P(aX, Y) \to P(X, Y1)$
  9. $P(aX, Y) \to P(X, Y0)$
  10. $P(bX, Y) \to P(X, Y1)$
  11. $P(bX, Y) \to P(X, Y0)$
  12. $P(cX, Y) \to P(X, Y1)$
  13. $P(cX, Y) \to P(X, Y0)$
  14. $P(dX, Y) \to B(X, Y)$
  15. $N(aX, Y) \to P(X, Y1)$
  16. $N(aX, Y) \to N(X, Y0)$
  17. $N(bX, Y) \to N(X, Y1)$
  18. $N(bX, Y) \to P(X, Y0)$
  19. $N(cX, Y) \to N(X, Y1)$
  20. $N(cX, Y) \to N(X, Y0)$

El no terminal A representa el predicado por donde inician las derivaciones de esta fase, P representa que con los valores de las variables que se han generado, la cláusula ya tiene un valor de verdad positivo y N representa que con esos mismos valores la fórmula booleana aún tiene un valor de verdad negativo.

Del no terminal A se deriva a los predicados P y N en dependencia del valor asignado a la variable del literal que se encuentra al inicio del rango actual. El predicado P deriva hacia sí mismo independientemente del símbolo, exceptuando el símbolo d, caso en el que se deriva en B y se procede a la siguiente fase.

Por último, del no terminal N se deriva a los predicados P y N en dependencia del valor asignado a la variable del literal que se encuentra al inicio del rango actual.

- **Tercera fase:** El siguiente conjunto de cláusulas comprueba que la asignación de variables que se realiza en la fase anterior sea verdadera para las restantes cláusulas.

  21. $B(X_1 dX_2, Y) \rightarrow C(X_1, Y)B(X_2, Y)$
  22. $B(\varepsilon, Y) \rightarrow \varepsilon$

  El predicado B permite identificar las cláusulas restantes, mientras que el predicado C comprueba que cada cláusula identificada por el predicado B sea satisfacible con los valores de las variables que recibe en su segundo argumento. Este comportamiento se define en la cuarta fase.

- **Cuarta fase:** En esta fase se define el comportamiento de C, que recibe una cláusula y una interpretación de las variables y comprueba que dicha interpretación sea verdadera para la cláusula analizada.

  23. $C(X, Y) \rightarrow Cn(X, Y)$
  24. $Cn(aX, 1Y) \rightarrow Cp(X, Y)$
  25. $Cn(aX, 0Y) \rightarrow Cn(X, Y)$
  26. $Cn(bX, 1Y) \rightarrow Cn(X, Y)$
  27. $Cn(bX, 0Y) \rightarrow Cp(X, Y)$
  28. $Cn(cX, 1Y) \rightarrow Cn(X, Y)$
  29. $Cn(cX, 0Y) \rightarrow Cn(X, Y)$
  30. $Cp(aX, 1Y) \rightarrow Cp(X, Y)$
  31. $Cp(aX, 0Y) \rightarrow Cp(X, Y)$
  32. $Cp(bX, 1Y) \rightarrow Cp(X, Y)$
  33. $Cp(bX, 0Y) \rightarrow Cp(X, Y)$
  34. $Cp(cX, 1Y) \rightarrow Cp(X, Y)$
  35. $Cp(cX, 0Y) \rightarrow Cp(X, Y)$
  36. $Cp(\varepsilon, \varepsilon) \rightarrow \varepsilon$

  Este funcionamiento sigue la misma idea que el descrito en la segunda fase: tiene un predicado que representa un estado positivo (Cp) y un predicado que representa un estado positivo (Cn). La diferencia es que no se genera la cadena, sino que se comprueba con el patrón que se construye en la segunda fase y que cada uno de los no terminales de esta fase recibe como argumento.

A continuación se demuestra que el lenguaje que reconoce G_{S-SAT} es exactamente igual al lenguaje que representa todas las fórmulas booleanas satisfacibles descritas mediante el lenguaje $L_{FULL-SAT}$.

5.1. La gramática G_{S-SAT} reconoce el lenguaje L_{S-SAT}

En esta sección se demuestra que G_{S-SAT} reconoce el lenguaje L_{S-SAT}.

Teorema 3. *Dada una cadena $e \in L_{FULL-SAT}$, G_{S-SAT} reconoce la cadena e si y solo si la fórmula booleana asociada a e es satisfacible.*

Para la demostración del Teorema 3 se usarán los siguientes lemas.

Lema 1. *Dadas las cadenas $q \in \{a, b, c\}^+$ y $w \in \{0, 1\}^+$, el predicado C de la gramática G_{S-SAT}, reconoce el vector $[q, w]$ si y solo si w satisface a la cláusula que representa q.*

Lema 2. *Dadas las cadenas $e \in L_{FULL-SAT}$ y $w \in \{0, 1\}^+$, el predicado B de la gramática G_{S-SAT}, reconoce el vector $[e, w]$ si y solo si w satisface a todas las cláusulas de e.*

Lema 3. *Dada una cadena $e \in L_{FULL-SAT}$, el conjunto de cadenas W formado por todas las cadenas $w \in \{0, 1\}^+$ tales que existe una secuencia de derivaciones desde del predicado $A(e)$ hasta $B(z_e, w)$, donde z_e es igual a la cadena e sin su primera cláusula, es exactamente igual a al conjunto de todas las interpretaciones que hacen verdadera la primera cláusula de e.*

La idea de la demostración del Teorema 3 es probar que dadas una cadena $q \in \{a, b, c\}^+$ y $w \in \{0, 1\}^+$, $C(q, w)$ se reconoce por la gramática si y solo si w satisface la cláusula que representa q, esto se plantea en el Lema 1.

Después de la demostración del Lema 1, dadas una cadena $e \in L_{FULL-SAT}$ y $w \in \{0, 1\}^+$, se hace una inducción sobre la cantidad de cláusulas de e para demostrar que G_{S-SAT} reconoce $B(e, w)$ si y solo si w satisface todas las cláusulas de e, esto se plantea en el Lema 2. Posteriormente, se prueba que durante la segunda fase se generan todas las cadenas binarias que satisfacen la primera cláusula de la cadena de entrada, esto se plantea en el Lema 3. Por último, se demuestra que el lenguaje que reconoce G_{S-SAT} es igual a L_{S-SAT}.

A continuación se demuestra el Lema 1 y con eso se garantiza la primera parte de la demostración.

Demostración del Lema 1.

Se definen las cadenas $q \in \{a, b, c\}^+$ y $w \in \{0, 1\}^+$, y la cláusula asociada a q F_q. Para demostrar que el predicado C de la gramática G_{S-SAT}, reconoce el vector $[q, w]$ si y solo si w satisface a F_q, primero se demuestra que $C(q, w)$ se reconoce por G_{S-SAT} si F_q es satisfacible por w y luego se prueba que si $C(q, w)$ se reconoce, entonces w satisface a F_q.

Para demostrar que $C(q, w)$ se reconoce por G_{S-SAT} si F_q es satisfacible por w, suponga que F_q es satisfacible por w, entonces se debe demostrar que existe una secuencia de derivaciones desde $C(q, w)$ hasta la cadena vacía.

Como w satisface a F_q, existe al menos un índice i menor que la longitud de w tal que $w_i = 1$ y $q_i = a$, o $w_i = 0$ y $q_i = b$. Si ese índice no existe w no puede satisfacer a F_q.

En la fase 4, del predicado C se deriva directamente al predicado Cn. Las únicas derivaciones de la gramática donde se deriva del predicado Cn a Cp son la combinación de una a y un 1 o de una b y un 0 y como w satisface F_q esta combinación existe en el índice i de ambas cadenas. Esto significa que al procesar los primeros i índices de q y w, la gramática deriva en el predicado Cp.

Al procesar los restantes $|q| - i$ índices, Cp siempre deriva en sí mismo o en la cadena vacía. De esta forma se demuestra que existe una secuencia de derivaciones desde $C(q, w)$ hasta la cadena vacía.

Para finalizar la demostración del Lema 1, es necesario probar que si $C(q, w)$ se reconoce, entonces w satisface a F_q.

Por la estructura de la gramática, si existe una secuencia de derivaciones desde $C(q, w)$ hasta la cadena vacía entonces hay una derivación desde Cn hacia Cp, sin pérdida de la generalidad esta derivación ocurre en el índice i de ambas cadenas. Esta derivación solo es posible por una combinación de una a y un 1 o de una b y un 0, por lo tanto una de estas combinaciones existe en el índice i. Por lo que cuando se le asignan los valores de w a las variables de F_q la variable con índice i está sin negar con valor *true* o está negada con valor *false*, lo cual implica que w satisface F_q, por tanto se demuestra el Lema 1. □

Una vez demostrado que $C(q, w)$ se reconoce si y solo si w satisface a q, se demuestra el Lema 2.

Demostración del Lema 2.

Dadas las cadenas $e \in L_{FULL-SAT}$ y $w \in \{0, 1\}^+$, para demostrar que el predicado B de la gramática G_{S-SAT}, reconoce el vector $[e, w]$ si y solo si w satisface a todas las cláusulas de e, se hará una inducción sobre la cantidad de cláusulas n de la fórmula booleana que representa e. En el caso base y en el paso inductivo se prueban ambos sentidos de la demostración.

Sea $n = 1$ y sea la cláusula de la gramática $B(X_1 dX_2, Y) \to C(X_1, Y)B(X_2, Y)$. Como e tiene un solo caracter d al final de la cadena, se cumple que al realizar la sustitución de rango, en el argumento $X_1 dX_2 = e$ los rangos asociados a las variables X_1 y X_2 son e sin su último caracter y la cadena vacía respectivamente. Por tanto $B(e, w)$ se reconoce por la gramática si y solo si $C(X_1, w)$ se reconoce, porque $B(\varepsilon, w)$ deriva en la cadena vacía. Por el Lema 1 $C(X_1, w)$ se reconoce si y solo si w satisface a X_1, por lo que se demuestra el caso base.

Una vez demostrado el caso base se asume que si la cantidad de cláusulas de la fórmula booleana que representa e es n y $n = k$, el predicado B reconoce el vector $[e, w]$ si y solo si w satisface a todas las cláusulas de e, y se demuestra para $n = k + 1$.

Dada la cláusula de la gramática $B(X_1 dX_2, Y) \to C(X_1, Y)B(X_2, Y)$, en todas las posibles sustituciones de rango de X_1 y X_2, $C(X_1, w)$ solo se reconoce si $|X_1| = |w|$, por lo tanto, el caso de sustitución de rango que ocupa a la demostración es cuando $|X_1| = |w|$, porque para el resto de las sustituciones de rango $C(X_1, w)$ no se reconoce por la gramática. Entonces X_1 es igual a la subcadena que contiene la primera cláusula de e y X_2 es igual a la subcadena que contiene el resto de las cláusulas de e.

La cadena w satisface todas las cláusulas de e si y solo si satisface a la primera cláusula de e y el resto de las cláusulas de e, que en este caso están asociadas a las variables X_1 y X_2 respectivamente. Precisamente $B(e, w)$ se reconoce si y solo si se reconoce $C(X_1, w)$ y $B(X_2, w)$.

$C(X_1, w)$ se reconoce si y solo si w satisface a X_1, por el Lema 1 y $B(X_2, w)$ se reconoce si y solo si w satisface todas las cláusulas de X_2 por hipótesis de inducción, ya que X_2 tiene k cláusulas. Por tanto, se demuestra el Lema 2 y esto significa que B reconoce el vector $[e, w]$ si y solo si w satisface a todas las cláusulas de e. □

Con la demostración del lema anterior se tiene que el predicado B reconoce una cadena que representa una fórmula booleana F y una cadena que representa una asignación de valores a las variables de F, si esta asignación satisface todas las cláusulas de F. Para completar la demostración de la gramática es necesario demostrar que al predicado B llegan todas las posibles interpretaciones que hacen verdadera la primera cláusula de la cadena de entrada y a ello se dedica la siguiente demostración.

Demostración del Lema 3.

Se definen las cadenas $e \in L_{FULL-SAT}$ y z_e, donde z_e es igual a la cadena e sin su primera cláusula. Además, se define el conjunto de cadenas W formado por todas las cadenas $w \in \{0,1\}^+$, tales que existe una secuencia de derivaciones desde del predicado $A(e)$ hasta $B(z_e, w)$, y el conjunto de todas las interpretaciones W' que hacen verdadera la primera cláusula de e, a la cual se le denomina F_{1e}. Para demostrar que $W = W'$, se debe demostrar que W es subconjunto de W' y que W' es subconjunto de W.

Para demostrar que $W' \subseteq W$, se toma una cadena w' tal que $w' \in W'$, es decir, w' satisface a F_{1e}. Por tanto en F_{1e}, cuando se le asignan los valores de w' a las variables de F_{1e}, la variable con índice i está sin negar con valor *true* o negada con valor *false*, lo que representa una combinación de una a y un 1 o de una b y un 0 en la i-ésima derivación de la segunda fase.

Por la estructura de la gramática del predicado A, solo hay derivaciones hacia P con una de estas 2 combinaciones, el resto son hacia el predicado N y del predicado N solo hay derivaciones a P con una de las combinaciones anteriores. Por tanto, como existe una combinación de una a y un 1 o de una b y un 0 en el índice i de ambas cadenas, las primeras i derivaciones de la gramática llevan del predicado A al predicado P. Como el predicado P solo tiene derivaciones hacia sí mismo o hacia $B(z_e, w')$, en las próximas $|e| - i$ derivaciones la gramática deriva en $B(z_e, w')$, por lo que se cumple que $w' \in W$.

Para demostrar que $W \subseteq W'$, se toma una cadena w tal que $w \in W$, por lo que existe una secuencia de derivaciones desde $A(e)$ a $B(z_e, w)$. Por la estructura de la gramática solo se puede derivar al predicado B desde el predicado P, y a su vez de este predicado solo se puede derivar mediante una combinación de una a y un 1 o de una b y un 0 en la gramática. Por tanto, cuando se le asignan los valores de w a las variables de F_{1e}, la variable con índice i está sin negar con valor *true* o negada con valor *false*. Entonces se cumple que w satisface a F_{1e} por lo que $w \in W'$. Con esto se demuestra que $W' = W$, por tanto se cumple el Lema 3. □

Con la demostración de los Lemas 1, 2 y 3 se puede demostrar el Teorema 3.

Demostración del Teorema 3.

Para demostrar que el lenguaje que reconoce G_{S-SAT} es exactamente igual al lenguaje que representa todas las fórmulas booleanas satisfacibles se define el lenguaje $L_{G_{S-SAT}}$ que representa el lenguaje de todas las cadenas que se reconocen por G_{S-SAT}, es necesario demostrar que $L_{S-SAT} = L_{G_{S-SAT}}$.

Para demostrar que $L_{S-SAT} \subseteq L_{G_{S-SAT}}$, sea una fórmula booleana satisfacible F y sea e su representación como cadena en el lenguaje $L_{FULL-SAT}$, entonces existe una cadena binaria w, con longitud igual a la cantidad de variables de F, que satisface a F.

Como w satisface a F entonces w pertenece al conjunto de cadenas que satisfacen a la primera cláusula de F. Por el Lema 3 existe una secuencia de derivaciones desde el predicado $S(e)$ hasta $B(z_e, w)$, y como w satisface todas las cláusulas de F entonces $B(z_e, w)$ deriva en la cadena vacía, por el Lema 2, por lo que e se reconoce por G_{S-SAT}, lo cual implica que $L_{S-SAT} \subseteq L_{G_{S-SAT}}$.

Para completar la demostración es necesario demostrar que $L_{G_{S-SAT}} \subseteq L_{S-SAT}$. Sea una cadena e que se reconoce por G_{S-SAT} y sea F la fórmula booleana asociada a e, entonces existe una cadena binaria w tal que existe una secuencia de derivaciones desde $A(e)$ a $B(z_e, w)$ y de $B(z_e, w)$ a la cadena vacía. Por el Lema 3 w satisface a la primera cláusula de F y por el Lema 2 w satisface a las restantes cláusulas de F también. Luego w satisface a F, por lo que F es satisfacible. Esto implica que $L_{G_{S-SAT}} \subseteq L_{S-SAT}$ y con esto se demuestra que $L_{G_{S-SAT}} = L_{S-SAT}$. □

En la siguiente sección se presenta un ejemplo del reconocimiento de una cadena por G_{S-SAT}.

5.2. Ejemplo de reconocimiento de G_{S-SAT}

En esta sección se presentan 2 ejemplos del funcionamiento de G_{S-SAT}. En el primero se muestra cómo se reconoce la cadena asociada a la fórmula booleana $(x_1 \vee x_2) \wedge (x_1) \wedge (\neg x_2)$ y en el segundo se muestra cómo no se reconoce la cadena asociada a la fórmula booleana $x_1 \wedge \neg x_1$.

La cadena asociada a $(x_1 \vee x_2) \wedge (x_1) \wedge (\neg x_2)$ es $aa\mathbf{d}ac\mathbf{d}cb\mathbf{d}$ y una posible secuencia de derivaciones asociada a esta cadena en G_{S-SAT} es la siguiente:

1. $S(aa\mathbf{d}ac\mathbf{d}cb\mathbf{d}) \rightarrow A(aa\mathbf{d}ac\mathbf{d}cb\mathbf{d})$
2. $A(aa\mathbf{d}ac\mathbf{d}cb\mathbf{d}) \rightarrow P(a\mathbf{d}ac\mathbf{d}cb\mathbf{d}, 1)$
3. $P(a\mathbf{d}ac\mathbf{d}cb\mathbf{d}, 1) \rightarrow P(\mathbf{d}ac\mathbf{d}cb\mathbf{d}, 10)$
4. $P(\mathbf{d}ac\mathbf{d}cb\mathbf{d}, 10) \rightarrow B(ac\mathbf{d}cb\mathbf{d}, 10)$
5. Para la producción
$$B(X_1 dX_2, Y) \rightarrow C(X_1, Y)B(X_2, Y),$$
con el vector $[ac\mathbf{d}cb\mathbf{d}, 10]$, se le asignan los valores $X_1 = ac$, $X_2 = cb\mathbf{d}$, $Y = 10$, con lo que se tiene la derivación:
$$B(ac\mathbf{d}cb\mathbf{d}, 10) \rightarrow C(ac, 10)B(cb\mathbf{d}, 10)$$
6. Para la producción
$$B(X_1 dX_2, Y) \rightarrow C(X_1, Y)B(X_2, Y),$$
con el vector $[cb\mathbf{d}, 10]$, se le asignan los valores $X_1 = cb$, $X_2 = \varepsilon$, $Y = 10$, con lo que se tiene la derivación:
$$B(cb\mathbf{d}, 10) \rightarrow C(cb, 10)B(\varepsilon, 10).$$
7. $B(\varepsilon, 10) \rightarrow \varepsilon$.
8. $C(ac, 10) \rightarrow Cn(ac, 10)$,
9. $Cn(ac, 10) \rightarrow Cp(c, 0)$,
10. $Cp(c, 0) \rightarrow Cp(\varepsilon, \varepsilon)$,
11. $Cp(\varepsilon, \varepsilon) \rightarrow \varepsilon$.
12. $C(cb, 10) \rightarrow Cn(cb, 10)$,
13. $Cn(cb, 10) \rightarrow Cn(b, 0)$,
14. $Cn(b, 0) \rightarrow Cp(\varepsilon, \varepsilon)$,
15. $Cp(\varepsilon, \varepsilon) \rightarrow \varepsilon$.

Como en todas las instancias de los no terminales existe una sustitución de rango que provoca que todos los predicados deriven en la cadena vacía, entonces $aa\mathbf{d}ac\mathbf{d}cb\mathbf{d}$ se reconoce por G_{S-SAT}. Esto coincide con el hecho de que $(x_1 \vee x_2) \wedge (x_1) \wedge (\neg x_2)$ es satisfacible con los valores $x_1 = true$ y $x_2 = false$.

Seguidamente, se presenta una cadena que representa una fórmula que no es satisfacible y por tanto la cadena asociada a dicha fórmula no se reconoce por G_{S-SAT}. La cadena asociada a $x_1 \wedge \neg x_1$ es $a\mathbf{d}b\mathbf{d}$, la secuencia de derivaciones asociada a esta cadena en G_{S-SAT} es la siguiente:

1. $S(a\mathbf{d}b\mathbf{d}) \rightarrow A(a\mathbf{d}b\mathbf{d})$
2. $A(a\mathbf{d}b\mathbf{d}) \rightarrow P(\mathbf{d}b\mathbf{d}, 1)$
3. $A(a\mathbf{d}b\mathbf{d}) \rightarrow N(\mathbf{d}b\mathbf{d}, 0)$

En la anterior secuencia de derivaciones hay 2 caminos posibles, cuando $A(a\mathbf{d}b\mathbf{d})$ deriva como $P(\mathbf{d}b\mathbf{d}, 1)$ o cuando $A(a\mathbf{d}b\mathbf{d})$ deriva como $N(\mathbf{d}b\mathbf{d}, 0)$. La siguiente secuencia de derivaciones corresponda al predicado P con el vector $[\mathbf{d}b\mathbf{d}, 1]$:

2. $P(\mathbf{d}b\mathbf{d}, 1) \rightarrow B(b\mathbf{d}, 1)$
3. Para la producción
$$B(X_1 dX_2, Y) \rightarrow C(X_1, Y)B(X_2, Y),$$
con el vector $[b\mathbf{d}, 1]$, la única asignación de valores posibles es: $X_1 = b$, $X_2 = \varepsilon$, $Y = 1$, con lo que se tiene la derivación:
$$B(b\mathbf{d}, 1) \rightarrow C(b, 1)B(\varepsilon, 1)$$
4. $B(\varepsilon, 1) \rightarrow \varepsilon$
5. $C(b, 1) \rightarrow Cn(b, 1)$
6. $Cn(b, 1) \rightarrow Cn(\varepsilon, \varepsilon)$

Cuando se realizan todas las posibles derivaciones para el predicado P con el vector $[\mathbf{d}b\mathbf{d}, 1]$, no se logra derivar en la cadena vacía, por lo tanto el predicado P no reconoce el vector $[\mathbf{d}b\mathbf{d}, 1]$.

Para el predicado N, con el vector $[\mathbf{d}b\mathbf{d}, 0]$, no existe ninguna derivación posible, por lo que N no reconoce el vector $[\mathbf{d}b\mathbf{d}, 0]$.

Después de realizarse todas las posibles derivaciones y sustituciones de rango, G_{S-SAT} no reconoce la cadena $a\mathbf{d}b\mathbf{d}$. Esto coincide con el hecho de que $x_1 \wedge \neg x_1$ no es satisfacible para ninguna asignación de valores a sus variables.

Como G_{S-SAT} reconoce las fórmulas booleanas satisfacibles, para determinar si una fórmula es satisfacible se debe determinar si su cadena asociada se reconoce por G_{S-SAT}. A continuación se analiza la complejidad del problema de la palabra para G_{S-SAT}.

5.3. Análisis de la complejidad computacional del reconocimiento en G_{S-SAT}

Como se mencionó en el sección 3 no todas las RCG tienen un algoritmo de reconocimiento polinomial y G_{S-SAT} es un ejemplo de ello.

La fase de la gramática que provoca que el algoritmo de reconocimiento sea exponencial es la segunda, ya que se generan todas las cadenas binarias que representan la asignación de valores para las variables booleanas. Estas cadenas se pasan como argumentos a los predicados de las fases posteriores, mediante las derivaciones de la gramática.

Si se analiza el algoritmo de reconocimiento descrito en [1], un factor en la complejidad del algoritmo de reconocimiento es la cantidad de rangos posibles para una cadena que se reconoce por un predicado. En este caso la cadena que representa los valores de las variables de la fórmula booleana puede tomar 2^n valores distintos, donde n es la cantidad de variables en la fórmula booleana, ya que dicha cadena se genera durante la segunda fase, donde la gramática es ambigua y en cada derivación hay decisiones que generan valores distintos.

Como se pueden generar 2^n cadenas, cada cadena tiene n^2 rangos y cada cadena generada se pasa como argumento a algún predicado de la gramática, la cantidad de rangos totales que se deben analizar durante el proceso de reconocimiento sería $n^2 2^n$.

Aunque esta es una cota burda, ya que para cada cadena no se utilizan todos los posibles rangos en el proceso de derivación de la gramática, el algoritmo sigue siendo exponencial con respecto al tamaño de la cadena de entrada.

El resto de las fases de la gramática tienen una complejidad de m^2 donde m es la cantidad de caracteres en la cadena de entrada, por lo que la complejidad total sería $O(2^n m^2)$.

En la siguiente sección se demuestra que no es necesaria la transducción del lenguaje $L_{0,1,d}$ mediante T_{SAT} para definir L_{S-SAT} y que las RCG reconocen todos los problemas de la clase NP.

5.4. Resultados derivados de la gramática G_{S-SAT}

Por el Teorema 1 se tiene que las RCG reconocen todos los problemas de la clase P. Como se mostró en la sección anterior con la gramática G_{S-SAT}, existe una RCG que reconoce el lenguaje de las fórmulas satisfacibles, y como el SAT se puede reducir a cualquier problema en NP en una complejidad polinomial, entonces para todo problema en NP también existe una RCG que lo reconoce en su representación como lenguaje formal.

En la próxima sección se utilizan las gramáticas de concatenación de rango para determinar si una fórmula booleana asociada al 2-SAT es satisfacible.

6. Instancias de SAT polinomiales empleando RCG

En esta sección se presenta una RCG que es capaz de reconocer problemas SAT satisfacibles que pertenecen al 2-SAT, es decir, problemas SAT donde cada cláusula tiene a lo sumo 2 literales. La idea detrás de esta gramática es obtener una RCG que reconozca cuándo la fórmula booleana pertenece al conjunto de fórmulas booleanas de 2-SAT y luego intersectar dicha gramática con G_{S-SAT}. Para ello se define la siguiente RCG:

$$G_{2-SAT} = (N, T, V, P, S),$$

donde:

- $N = \{S, A, A_0, A_1, A_2, A_3\}$
- $T = \{a, b, c, d\}$.
- $V = \{X, Y, X_1, X_2\}$.
- El conjunto de cláusulas P es el siguiente:

1. $S(X) \to A(X)$
2. $A(X_1 dX_2) \to A_0(X_1)A(X_2)$
3. $A(\varepsilon) \to \varepsilon$
4. $A_0(aX) \to A_1(X)$
5. $A_0(bX) \to A_1(X)$
6. $A_0(cX) \to A_0(X)$
7. $A_1(aX) \to A_2(X)$
8. $A_1(bX) \to A_2(X)$
9. $A_1(cX) \to A_1(X)$
10. $A_2(aX) \to A_3(X)$
11. $A_2(bX) \to A_3(X)$
12. $A_2(cX) \to A_2(X)$
13. $A_2(\varepsilon) \to \varepsilon$

- El **símbolo inicial** es S.

El funcionamiento de la gramática anterior es el siguiente: la segunda cláusula permite reconocer todas las cláusulas asociadas a la cadena original. Las cláusulas de la 4 a la 13 permiten contar la cantidad de a o b en una cláusula, o sea, la cantidad de literales de cada cláusula. Para esto se definen 4 estados: A_0, A_1, A_2 y A_3. A_0 representa que se reconocieron 0 a o b, A_1 representa que se reconocieron una a o b, A_2 representa que se reconocieron 2 a o b y A_3 representa que se reconocen más de 2 a o b.

Si se observa el enfoque seguido en la construcción de G_{S-SAT}, en la representación del SAT como cadena se trabaja con una instancia del SAT general. Por lo que no se tienen en cuenta las propiedades específicas del problema, que en el caso de las instancias polinomiales, es lo que permite que el algoritmo para las mismas sea polinomial.

Finalmente, la gramática que reconoce los problemas 2-SAT satisfacibles sería:

$$G_{S-2-SAT} = G_{S-SAT} \cap G_{2-SAT}.$$

7. Problemas propuestos

Como las RCG reconocen todos los problemas de la clase P, entonces es posible diseñar una RCG para el 2-SAT y para cada instancia polinomial del SAT, donde el problema de la palabra sea polinomial para dicha gramática. Lo anterior queda propuesto como un problema abierto, porque tiene que existir una RCG que reconozca el 2-SAT en tiempo polinomial. La estructura de esta gramática puede conducir a encontrar nuevas instancias polinomiales del SAT.

Referencias

[1] Boullier, Pierre. *Proposal for a Natural Language Processing Syntactic Backbone*. Research Report RR-3342, INRIA, 1998.

[2] Boullier, Pierre. *A Cubic Time Extension of Context-Free Grammars*. Research Report RR-3611, INRIA, 1999.

[3] Eberhard Bertsch and Mark-Jan Nederhof *On the Complexity of Some Extensions of RCG Parsing*. International Workshop/Conference on Parsing Technologies, 2001.

[4] Boullier, Pierre. *Counting with range concatenation grammar*. Theor. Comput. Sci., 2003.

[5] Ibarra, Oscar H. *Simple matrix languages. Information and Control*, Vol. 17, No. 4, pp. 359-394, 1970.

[6] Castaño, José M. *Global Index Languages*. Ph.D. Thesis, The Faculty of the Graduate School of Arts and Sciences, Brandeis University, 2004.

[7] Hopcroft, John E., Motwani, Rajeev, y Ullman, Jeffrey D. *Introduction to Automata Theory, Languages, and Computation*. 3ª edición, Addison-Wesley, 2006. ISBN: 9780321455369.

[8] Fernández Arias, Alina. *El problema de la satisfacibilidad booleana libre del contexto*. Facultad de Matemática y Computación, Universidad de La Habana, 2007.

[9] Aguilera López, Manuel. *Problema de la Satisfacibilidad Booleana de Concatenación de Rango Simple*. Facultad de Matemática y Computación, Universidad de La Habana, 2016.

[10] Rodríguez Salgado, José Jorge. *Gramáticas Matriciales Simples. Primera aproximación para una solución al problema SAT*. Facultad de Matemática y Computación, Universidad de La Habana, 2019.

[11] Boullier, Pierre. *Chinese Numbers, MIX, Scrambling, and Range Concatenation Grammars*. Proceedings of the Ninth Conference of the European Chapter of the Association for Computational Linguistics, pages 53–60. Association for Computational Linguistics, 1999.