Una aproximación al lenguaje de todas las fórmulas booleanas satisfacibles

Raudel Alejando Gómez Molina

Facultad de Matemática y Computación
Universidad de La Habana

22 de febrero de 2025

Tutor: MSc. Fernando Raul Rodriguez Flores

Teoría de lenguajes (Conceptos)

- Un alfabeto es un conjunto finito de símbolos

Teoría de lenguajes (Conceptos)

- Un alfabeto es un conjunto finito de símbolos
- Una cadena es una sucesión finita de símbolos del alfabeto

Teoría de lenguajes (Conceptos)

- Un alfabeto es un conjunto finito de símbolos
- Una cadena es una sucesión finita de símbolos del alfabeto
- Un lenguaje es un conjunto finito de cadenas

Teoría de lenguajes (Problemas)

- Problema de la palabra: una cadena pertenece a un lenguaje

Teoría de lenguajes (Problemas)

- Problema de la palabra: una cadena pertenece a un lenguaje
- Todo problema se puede reducir a un problema de la palabra

Teoría de lenguajes (Problemas)

- Problema de la palabra: una cadena pertenece a un lenguaje
- Todo problema se puede reducir a un problema de la palabra
- Todo problema se puede codificar como lenguaje formal

Teoría de lenguajes (Problemas)

- Problema de la palabra: una cadena pertenece a un lenguaje
- Todo problema se puede reducir a un problema de la palabra
- Todo problema se puede codificar como lenguaje formal
- Existen problemas para los cuales no se conoce una solución eficiente

Teoría de lenguajes (Problemas)

- Problema de la palabra: una cadena pertenece a un lenguaje
- Todo problema se puede reducir a un problema de la palabra
- Todo problema se puede codificar como lenguaje formal
- Existen problemas para los cuales no se conoce una solución eficiente
- Comprobar si una solución es válida es eficiente (clase NP)

Problema de la Satisfacibilidad booleana (SAT)

- El SAT es el primer problema demostrado como NP-Completo

Problema de la Satisfacibilidad booleana (SAT)

- El SAT es el primer problema demostrado como NP-Completo
- Pertenece a la clase NP

Problema de la Satisfacibilidad booleana (SAT)

- El SAT es el primer problema demostrado como NP-Completo
- Pertenece a la clase NP
- Todo problema en NP puede reducirse a él en tiempo polinomial

Problema de la Satisfacibilidad booleana (SAT)

- El SAT es el primer problema demostrado como NP-Completo
- Pertenece a la clase NP
- Todo problema en NP puede reducirse a él en tiempo polinomial
- Consiste en determinar si una fórmula booleana es satisfacible

Objetivo

Definir y construir el lenguaje de todas las fórmulas booleanas satisfacibles

Estructura de la presentación

1. Codificación de una fórmula booleana en una cadena
2. Asignar valores a una fórmula booleana mediante una cadena
3. Construcción de L_{S-SAT} mediante una transducción finita
4. Gramáticas de concatenación de rango
5. Construcción L_{S-SAT} utilizando una RCG
6. Recomendaciones

Contenido

Fórmula normal conjuntiva (CNF)

- Toda fórmula booleana tiene una fórmula equivalente en CNF

Fórmula normal conjuntiva (CNF)

- Toda fórmula booleana tiene una fórmula equivalente en CNF
- Una CNF es una conjunción de cláusulas

Fórmula normal conjuntiva (CNF)

- Toda fórmula booleana tiene una fórmula equivalente en CNF
- Una CNF es una conjunción de cláusulas
- Una cláusula es una disyunción de literales

Fórmula normal conjuntiva (CNF)

- Toda fórmula booleana tiene una fórmula equivalente en CNF
- Una CNF es una conjunción de cláusulas
- Una cláusula es una disyunción de literales
- Un literal es una variable booleana o su negación

Fórmula normal conjuntiva (CNF)

- Toda fórmula booleana tiene una fórmula equivalente en CNF
- Una CNF es una conjunción de cláusulas
- Una cláusula es una disyunción de literales
- Un literal es una variable booleana o su negación
- Asumimos que las fórmulas booleanas están en CNF

Estados de una variable en una cláusula

$$x_1 \vee x_2 \vee \neg x_3$$

Estados de una variable en una cláusula

$$x_1 \vee x_2 \vee \neg x_3$$

- x_1 está sin negar en la cláusula

Estados de una variable en una cláusula

$$x_1 \vee x_2 \vee \neg x_3$$

- x_1 está sin negar en la cláusula
- x_2 no está en la cláusula

Estados de una variable en una cláusula

$$x_1 \vee x_2 \vee \neg x_3$$

- x_1 está sin negar en la cláusula
- x_2 no está en la cláusula
- x_3 está negada en la cláusula

Codificación de una cláusula en una cadena

- a: la variable está sin negar en la cláusula

Codificación de una cláusula en una cadena

- a: la variable está sin negar en la cláusula
- b: la variable está negada en la cláusula

Codificación de una cláusula en una cadena

- a: la variable está sin negar en la cláusula
- b: la variable está negada en la cláusula
- c: la variable no está en la cláusula

Codificación de una cláusula en una cadena

- a: la variable está sin negar en la cláusula
- b: la variable está negada en la cláusula
- c: la variable no está en la cláusula

$$x_1 \vee x_2 \vee \neg x_3 \iff acb$$

Codificación de una fórmula booleana en una cadena

- Obtener la codificación de cada cláusula

Codificación de una fórmula booleana en una cadena

- Obtener la codificación de cada cláusula
- Establecer un separador para delimitar cada cláusula d

Codificación de una fórmula booleana en una cadena

- Obtener la codificación de cada cláusula
- Establecer un separador para delimitar cada cláusula d
- Concatenar cada cláusula seguida del separador d

Codificación de una fórmula booleana en una cadena

- Obtener la codificación de cada cláusula
- Establecer un separador para delimitar cada cláusula d
- Concatenar cada cláusula seguida del separador d

$$(x_1) \wedge (x_1 \vee \neg x_2 \vee x_3) \wedge (\neg x_2 \vee x_3) \Leftrightarrow acc\mathbf{d}aba\mathbf{d}cba\mathbf{d}$$

Lenguaje de todas las fórmulas booleanas satisfacibles

- $L_{FULL-SAT}$ lenguaje de todas las fórmulas booleanas en CNF

Lenguaje de todas las fórmulas booleanas satisfacibles

- $L_{FULL-SAT}$ lenguaje de todas las fórmulas booleanas en CNF
- L_{S-SAT} lenguaje de todas las fórmulas booleanas satisfacibles

Contenido

Asignar valores a una cláusula mediante una cadena binaria

- Se tiene una cadena q que representa una cláusula

Asignar valores a una cláusula mediante una cadena binaria

- Se tiene una cadena q que representa una cláusula
- Se tiene una cadena binaria w que representa una asignación

Asignar valores a una cláusula mediante una cadena binaria

- Se tiene una cadena q que representa una cláusula
- Se tiene una cadena binaria w que representa una asignación
- Se debe cumplir que $|q| = |w|$

Asignar valores a una cláusula mediante una cadena binaria

- Se tiene una cadena q que representa una cláusula
- Se tiene una cadena binaria w que representa una asignación
- Se debe cumplir que $|q| = |w|$
- Si el i-ésimo caracter de q es 1, $x_i = true$

Asignar valores a una cláusula mediante una cadena binaria

- Se tiene una cadena q que representa una cláusula
- Se tiene una cadena binaria w que representa una asignación
- Se debe cumplir que $|q| = |w|$
- Si el i-ésimo caracter de q es 1, $x_i = \mathit{true}$
- Si el i-ésimo caracter de q es 0, $x_i = \mathit{false}$

Asignar valores a una cláusula mediante una cadena binaria

$$w = 101 \qquad C = x_1 \vee x_2 \vee \neg x_3 \;\Leftrightarrow\; q = acb$$

Asignar valores a una cláusula mediante una cadena binaria

$$w = 101 \qquad C = x_1 \vee x_2 \vee \neg x_3 \;\Leftrightarrow\; q = acb$$

- $w_1 = 1 \Rightarrow x_1 = true$ C se evalúa positiva

Asignar valores a una cláusula mediante una cadena binaria

$$w = 101 \qquad C = x_1 \vee x_2 \vee \neg x_3 \;\Leftrightarrow\; q = acb$$

- $w_1 = 1 \Rightarrow x_1 = \mathit{true}$ C se evalúa positiva
- $w_2 = 0 \Rightarrow x_2 = \mathit{false}$ C se mantiene positiva

Asignar valores a una cláusula mediante una cadena binaria

$$w = 101 \qquad C = x_1 \vee x_2 \vee \neg x_3 \;\Leftrightarrow\; q = acb$$

- $w_1 = 1 \Rightarrow x_1 = \mathit{true}$ C se evalúa positiva
- $w_2 = 0 \Rightarrow x_2 = \mathit{false}$ C se mantiene positiva
- $w_3 = 1 \Rightarrow x_3 = \mathit{true}$ C se mantiene positiva

Asignar valores a una cláusula mediante una cadena binaria

$$w = 010 \qquad C = x_1 \vee x_2 \vee \neg x_3 \;\Leftrightarrow\; q = acb$$

Asignar valores a una cláusula mediante una cadena binaria

$$w = 010 \qquad C = x_1 \vee x_2 \vee \neg x_3 \;\Leftrightarrow\; q = acb$$

- $w_1 = 0 \Rightarrow x_1 = \mathit{false}$ C se evalúa negativa

Asignar valores a una cláusula mediante una cadena binaria

$$w = 010 \qquad C = x_1 \vee x_2 \vee \neg x_3 \;\Leftrightarrow\; q = acb$$

- $w_1 = 0 \Rightarrow x_1 = \mathit{false}$ C se evalúa negativa
- $w_2 = 1 \Rightarrow x_2 = \mathit{true}$ C se mantiene negativa

Asignar valores a una cláusula mediante una cadena binaria

$$w = 010 \qquad C = x_1 \vee x_2 \vee \neg x_3 \;\Leftrightarrow\; q = acb$$

- $w_1 = 0 \Rightarrow x_1 = \mathit{false}$ C se evalúa negativa
- $w_2 = 1 \Rightarrow x_2 = \mathit{true}$ C se mantiene negativa
- $w_3 = 0 \Rightarrow x_1 = \mathit{false}$ C se evalúa positiva

Asignar valores a una fórmula mediante una cadena

- Se tiene una cadena binaria w

Asignar valores a una fórmula mediante una cadena

- Se tiene una cadena binaria w
- w representa la asignación de valores para una cláusula

Asignar valores a una fórmula mediante una cadena

- Se tiene una cadena binaria w
- w representa la asignación de valores para una cláusula
- Si la fórmula booleana F tiene n cláusulas

Asignar valores a una fórmula mediante una cadena

- Se tiene una cadena binaria w
- w representa la asignación de valores para una cláusula
- Si la fórmula booleana F tiene n cláusulas
- $(wd)^n$ representa la asignación de variables para F

Asignar valores a una fórmula mediante una cadena

- Se tiene una cadena binaria w
- w representa la asignación de valores para una cláusula
- Si la fórmula booleana F tiene n cláusulas
- $(wd)^n$ representa la asignación de variables para F
- $r = 101\mathbf{d}101\mathbf{d}101\mathbf{d}$ y $e = acc\mathbf{d}aba\mathbf{d}cba\mathbf{d}$

$$(true) \wedge (true \vee \neg false \vee true) \wedge (\neg false \vee true) = true$$

Asignar valores a una fórmula mediante una cadena

- Se tiene una cadena binaria w
- w representa la asignación de valores para una cláusula
- Si la fórmula booleana F tiene n cláusulas
- $(wd)^n$ representa la asignación de variables para F
- $r = 101\mathbf{d}101\mathbf{d}101\mathbf{d}$ y $e = acc\mathbf{d}aba\mathbf{d}cba\mathbf{d}$

$$(true) \wedge (true \vee \neg false \vee true) \wedge (\neg false \vee true) = true$$

- $L_{0,1,d} = \{(wd)^+ \mid w \in \{0,1\}^+\}$ lenguaje de todas las interpretaciones

Contenido

Construcción de L_{S-SAT} mediante una transducción finita

- Cadena binaria w generar todas las cláusulas satisfacibles por w

Construcción de L_{S-SAT} mediante una transducción finita

- Cadena binaria w generar todas las cláusulas satisfacibles por w
- Cadena $r \in L_{0,1,d}$ generar todas las cadenas $e \in L_{FULL-SAT}$

Construcción de L_{S-SAT} mediante una transducción finita

- Cadena binaria w generar todas las cláusulas satisfacibles por w
- Cadena $r \in L_{0,1,d}$ generar todas las cadenas $e \in L_{FULL-SAT}$
- e representa una fórmula booleana satisfacible por r

Construcción de L_{S-SAT} mediante una transducción finita

- Cadena binaria w generar todas las cláusulas satisfacibles por w
- Cadena $r \in L_{0,1,d}$ generar todas las cadenas $e \in L_{FULL-SAT}$
- e representa una fórmula booleana satisfacible por r
- Transductor finito (autómata finito que escribe símbolos)

Transductor T_{CLAUSE}

Entrada y Salida

- Cadena binaria w
- Todas las cláusulas satisfacibles por w

Transductor T_{CLAUSE}

Entrada y Salida

- Cadena binaria w
- Todas las cláusulas satisfacibles por w

Estados

- q_0: estado inicial
- q_p: estado positivo (estado de aceptación)
- q_n: estado negativo

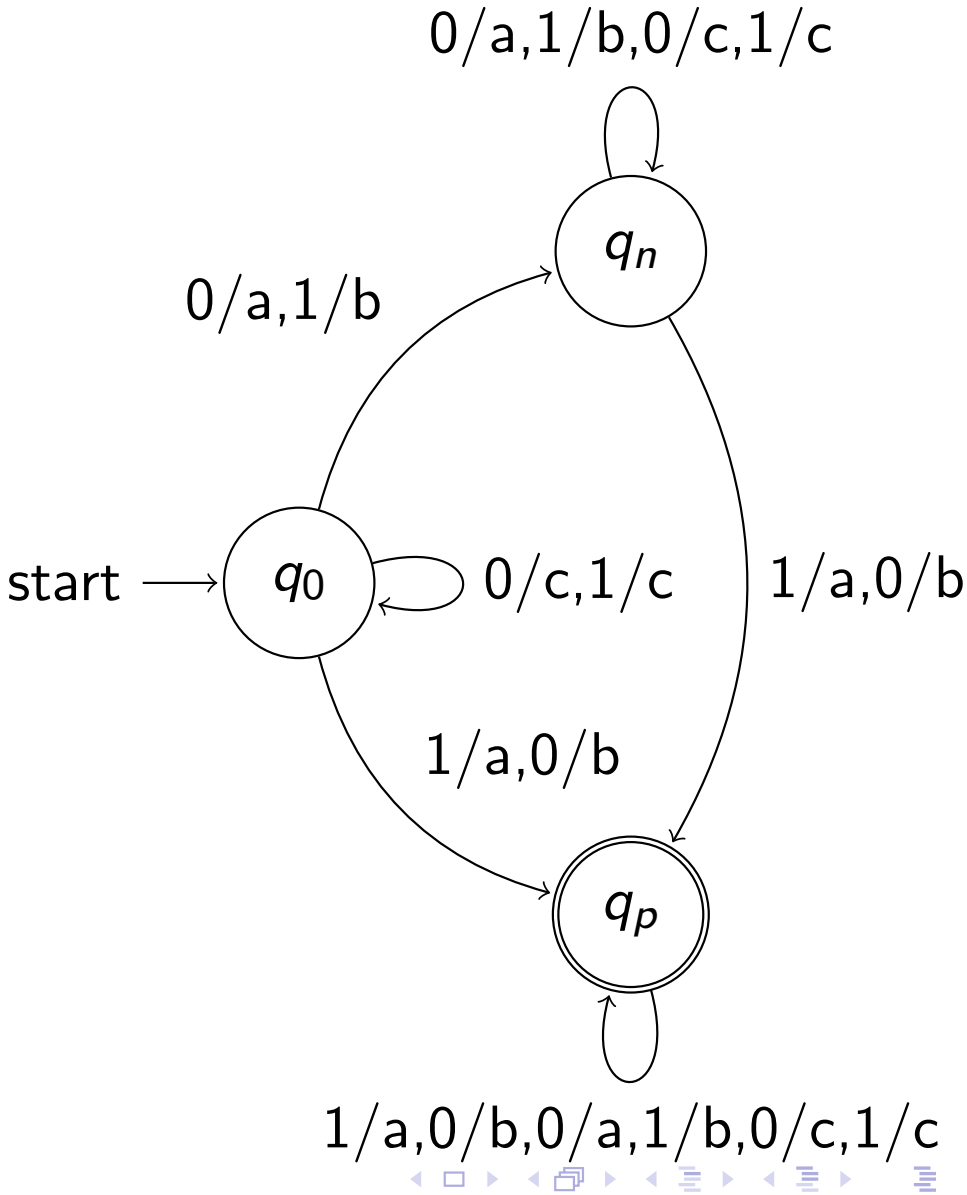
Transductor T_{CLAUSE}

Entrada y Salida

- Cadena binaria w
- Todas las cláusulas satisfacibles por w

Estados

- q_0: estado inicial
- q_p: estado positivo (estado de aceptación)
- q_n: estado negativo



Transductor T_{SAT}

Entrada y Salida

- Cadena $r \in L_{0,1,d}$
- Todas las fórmulas satisfacibles por r

Transductor T_{SAT}

Entrada y Salida

- Cadena $r \in L_{0,1,d}$
- Todas las fórmulas satisfacibles por r

Estados

- q_0: estado inicial (estado de aceptación)
- q_p: estado positivo
- q_n: estado negativo

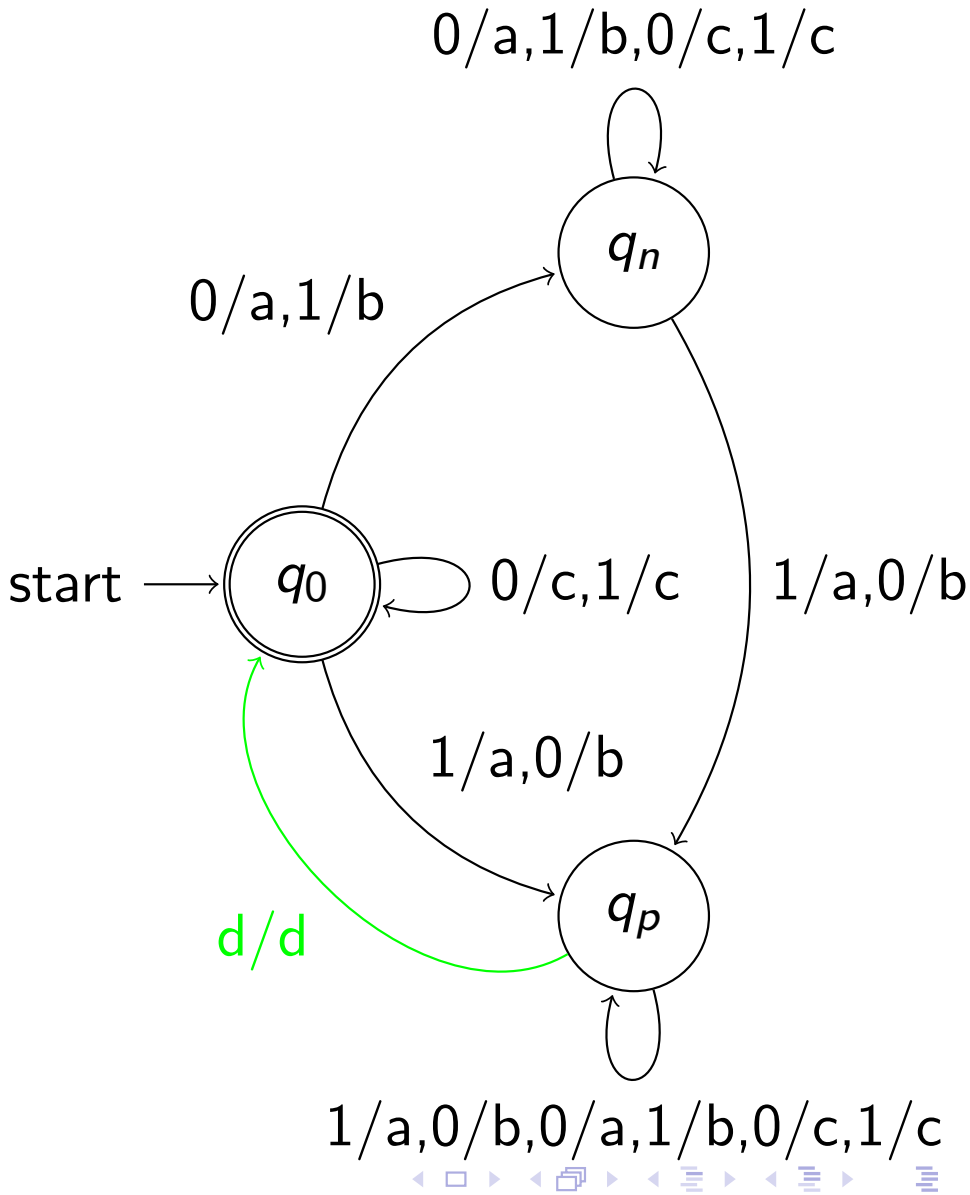
Transductor T_{SAT}

Entrada y Salida

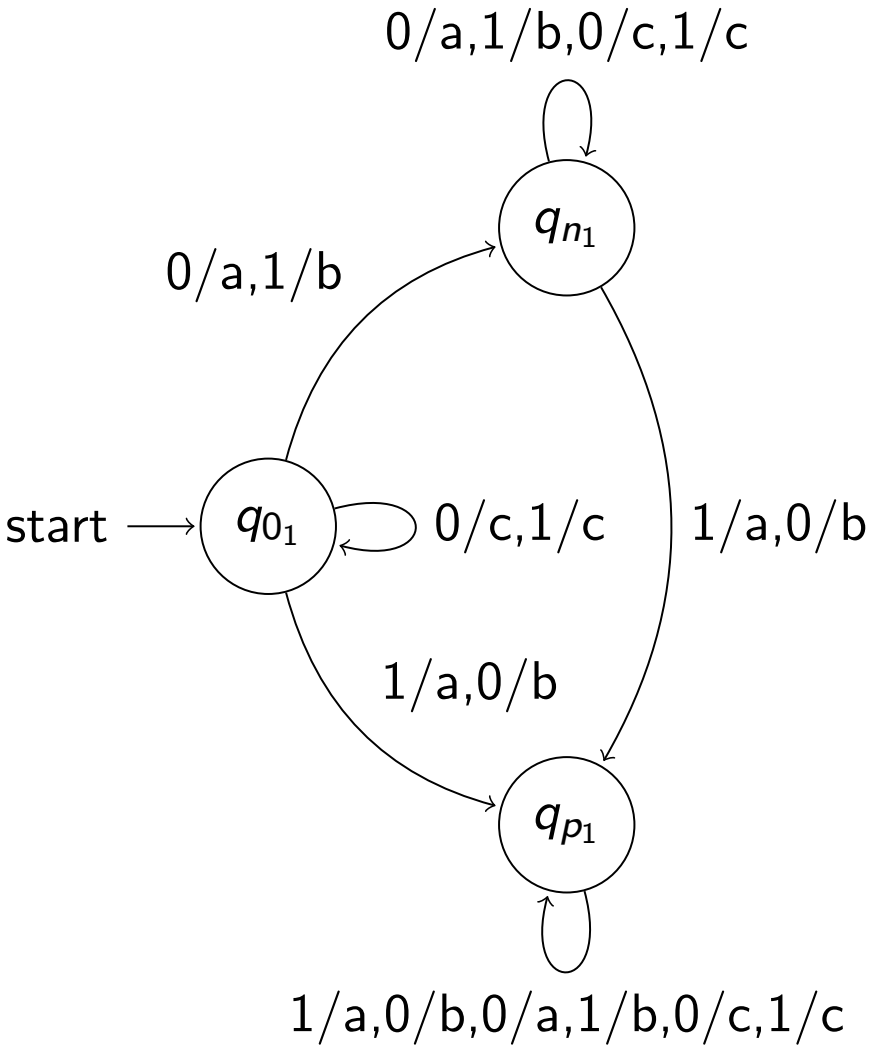
- Cadena $r \in L_{0,1,d}$
- Todas las fórmulas satisfacibles por r

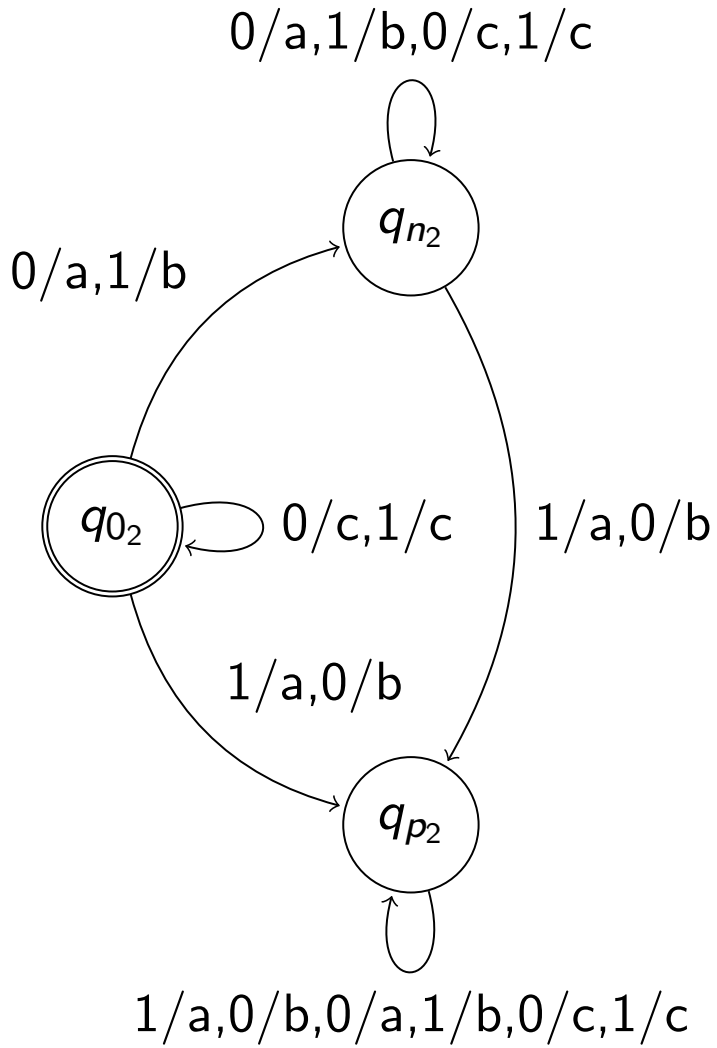
Estados

- q_0: estado inicial (estado de aceptación)
- q_p: estado positivo
- q_n: estado negativo

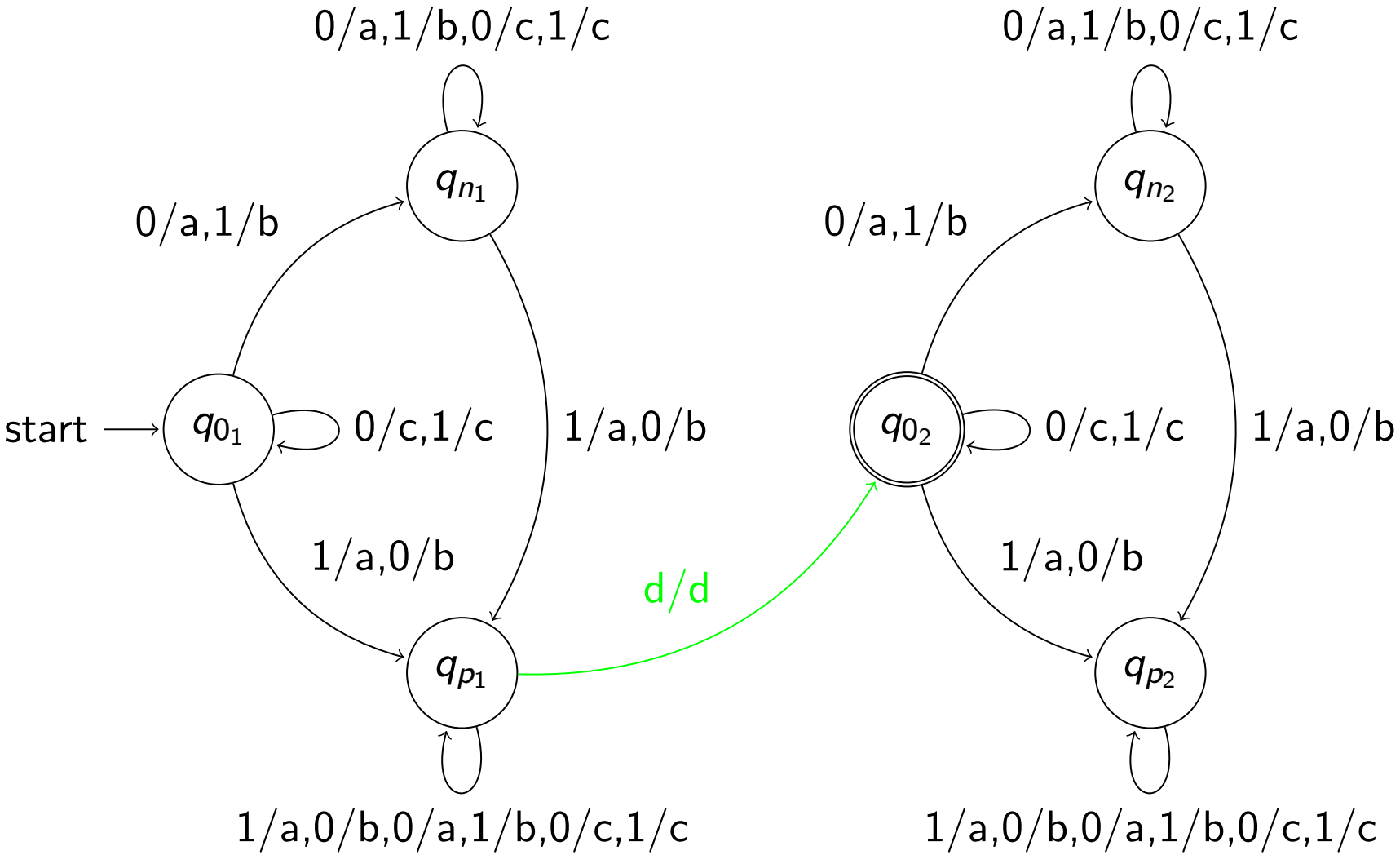


Transductor T_{SAT}

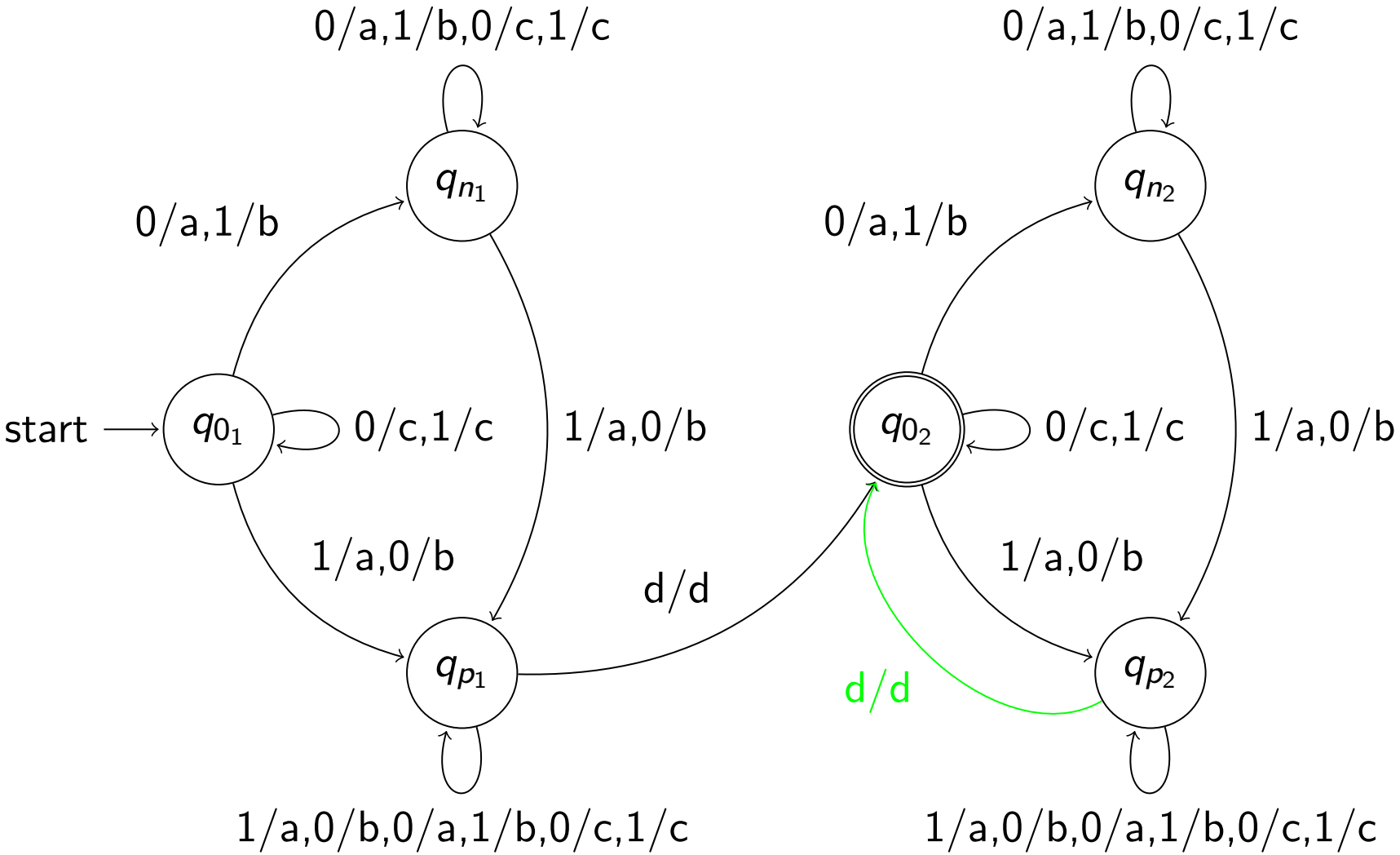




Transductor T_{SAT}



Transductor T_{SAT}



Construcción de L_{S-SAT} mediante T_{SAT}

$$L_{S-SAT} = \{e \mid \exists r \in L_{0,1,d} \text{ y } e \in T_{SAT}(r)\}$$

Construcción de L_{S-SAT} mediante T_{SAT}

$$L_{S-SAT} = \{e \mid \exists r \in L_{0,1,d} \text{ y } e \in T_{SAT}(r)\}$$

- Es necesario un formalismo que genere $L_{0,1,d}$

Construcción de L_{S-SAT} mediante T_{SAT}

$$L_{S-SAT} = \{e \mid \exists r \in L_{0,1,d} \text{ y } e \in T_{SAT}(r)\}$$

- Es necesario un formalismo que genere $L_{0,1,d}$
- Sea cerrado bajo transducción finita

Construcción de L_{S-SAT} mediante T_{SAT}

$$L_{S-SAT} = \{e \mid \exists r \in L_{0,1,d} \text{ y } e \in T_{SAT}(r)\}$$

- Es necesario un formalismo que genere $L_{0,1,d}$
- Sea cerrado bajo transducción finita
- Resolver el problema de la palabra

Resultados derivados de T_{SAT}

- Formalismo que genere $L_{0,1,d}$ $(G_{0,1,d})$

Resultados derivados de T_{SAT}

- Formalismo que genere $L_{0,1,d}$ ($G_{0,1,d}$)
- Si $G_{0,1,d}$ es cerrado bajo transducción finita

Resultados derivados de T_{SAT}

- Formalismo que genere $L_{0,1,d}$ ($G_{0,1,d}$)
- Si $G_{0,1,d}$ es cerrado bajo transducción finita
- El problema de la palabra de $G_{0,1,d}$ es NP-Duro

Resultados derivados de T_{SAT}

- Formalismo que genere $L_{0,1,d}$ ($G_{0,1,d}$)
- Si $G_{0,1,d}$ es cerrado bajo transducción finita
- El problema de la palabra de $G_{0,1,d}$ es NP-Duro
- Todo formalismo que genere $L_{0,1,d}$ tiene tamaño $O(1)$ (Conjetura)

Contenido

Gramáticas de concatenación de rango (RCG)

- Un rango es un intervalo de la cadena

Gramáticas de concatenación de rango (RCG)

- Un rango es un intervalo de la cadena
- A los no terminales se les llama predicados

Gramáticas de concatenación de rango (RCG)

- Un rango es un intervalo de la cadena
- A los no terminales se les llama predicados
- Cada predicado tiene una secuencia de argumentos

Gramáticas de concatenación de rango (RCG)

- Un rango es un intervalo de la cadena
- A los no terminales se les llama predicados
- Cada predicado tiene una secuencia de argumentos
- Cada predicado recibe un vector de cadenas

Gramáticas de concatenación de rango (RCG)

- Un rango es un intervalo de la cadena
- A los no terminales se les llama predicados
- Cada predicado tiene una secuencia de argumentos
- Cada predicado recibe un vector de cadenas
- Cada argumento está formado por variables y no terminales

$$A(aX, cbYZ)$$

Gramáticas de concatenación de rango (RCG)

- Un rango es un intervalo de la cadena
- A los no terminales se les llama predicados
- Cada predicado tiene una secuencia de argumentos
- Cada predicado recibe un vector de cadenas
- Cada argumento está formado por variables y no terminales

$$A(aX, cbYZ)$$

- A las producciones se les llama cláusulas

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

Gramáticas de concatenación de rango (RCG)

- Un rango es un intervalo de la cadena
- A los no terminales se les llama predicados
- Cada predicado tiene una secuencia de argumentos
- Cada predicado recibe un vector de cadenas
- Cada argumento está formado por variables y no terminales

$$A(aX, cbYZ)$$

- A las producciones se les llama cláusulas

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

- Las RCG no generan, por el contrario, reconocen cadenas

Sustitución de rango

Asocia una variable de un argumento a un rango de una cadena respetando la estructura del argumento

Sustitución de rango

Asocia una variable de un argumento a un rango de una cadena respetando la estructura del argumento

$$XYZ = abc$$

Sustitución de rango

Asocia una variable de un argumento a un rango de una cadena respetando la estructura del argumento

$$XYZ = abc$$

X	Y	Z
a	b	c
ab	ε	c
ab	c	ε
abc	ε	ε
ε	ab	c
ε	abc	ε
ε	ε	abc
a	ε	bc
⋮	⋮	⋮
a	bc	ε

Derivación en las RCG

- Cada cadena del vector del predicado izquierdo se asocia a un argumento

Derivación en las RCG

- Cada cadena del vector del predicado izquierdo se asocia a un argumento
- Sustitución de rango para cada argumento

Derivación en las RCG

- Cada cadena del vector del predicado izquierdo se asocia a un argumento
- Sustitución de rango para cada argumento
- Se obtienen los valores de las variables

Derivación en las RCG

- Cada cadena del vector del predicado izquierdo se asocia a un argumento
- Sustitución de rango para cada argumento
- Se obtienen los valores de las variables
- Se construyen los vectores y se evalúa en los predicados derechos

Derivación en las RCG

- Cada cadena del vector del predicado izquierdo se asocia a un argumento
- Sustitución de rango para cada argumento
- Se obtienen los valores de las variables
- Se construyen los vectores y se evalúa en los predicados derechos
- Se repite el mismo proceso en cada uno de los predicados derechos

Reconocer un vector y Problema de la palabra

- Si existe una secuencia de derivaciones y sustituciones de rango

Reconocer un vector y Problema de la palabra

- Si existe una secuencia de derivaciones y sustituciones de rango
- Tales que desde el predicado se deriva en la cadena vacía

- Para la mayoría de las RCG el problema de la palabra es polinomial

Reconocer un vector y Problema de la palabra

- Si existe una secuencia de derivaciones y sustituciones de rango
- Tales que desde el predicado se deriva en la cadena vacía

- Para la mayoría de las RCG el problema de la palabra es polinomial
- RCG ambiguas cuyo problema de la palabra no es polinomial

Contenido

$L_{0,1,d}$ como lenguaje de concatenación de rango

Producciones de la gramática $G_{0,1,d}$

- $S(X) \rightarrow A(X)$

$L_{0,1,d}$ como lenguaje de concatenación de rango

Producciones de la gramática $G_{0,1,d}$

- $S(X) \to A(X)$
- $A(XdY) \to B(Y,X)C(X)$

$L_{0,1,d}$ como lenguaje de concatenación de rango

Producciones de la gramática $G_{0,1,d}$

- $S(X) \rightarrow A(X)$
- $A(XdY) \rightarrow B(Y, X)C(X)$
- $B(XdY, P) \rightarrow B(Y, P)C(X)Eq(X, P)$

$L_{0,1,d}$ como lenguaje de concatenación de rango

Producciones de la gramática $G_{0,1,d}$

- $S(X) \to A(X)$
- $A(XdY) \to B(Y, X)C(X)$
- $B(XdY, P) \to B(Y, P)C(X)Eq(X, P)$
- $B(\varepsilon, P) \to \varepsilon$

- C comprueba que la cadena está formada por 0 y 1

$L_{0,1,d}$ como lenguaje de concatenación de rango

Producciones de la gramática $G_{0,1,d}$

- $S(X) \to A(X)$
- $A(XdY) \to B(Y, X)C(X)$
- $B(XdY, P) \to B(Y, P)C(X)Eq(X, P)$
- $B(\varepsilon, P) \to \varepsilon$

- C comprueba que la cadena está formada por 0 y 1
- Eq comprueba que 2 cadenas sean iguales

Construcción de L_{SAT} mediante una RCG

Las producciones de la gramática G_{S-SAT} se agrupan en 4 grupos (fases):

1. Derivación inicial de la gramática

Construcción de L_{SAT} mediante una RCG

Las producciones de la gramática G_{S-SAT} se agrupan en 4 grupos (fases):

1. Derivación inicial de la gramática
2. Todas las posibles interpretaciones (verdadera la primera cláusula)

Construcción de L_{SAT} mediante una RCG

Las producciones de la gramática G_{S-SAT} se agrupan en 4 grupos (fases):

1. Derivación inicial de la gramática
2. Todas las posibles interpretaciones (verdadera la primera cláusula)
3. La interpretación generada satisface el resto de las cláusulas

Construcción de L_{SAT} mediante una RCG

Las producciones de la gramática G_{S-SAT} se agrupan en 4 grupos (fases):

1. Derivación inicial de la gramática
2. Todas las posibles interpretaciones (verdadera la primera cláusula)
3. La interpretación generada satisface el resto de las cláusulas
4. La interpretación generada satisface una cláusula

Producciones de G_{S-SAT}

Producciones agrupadas por fases

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

Producciones de G_{S-SAT}

Producciones agrupadas por fases

1. $S(X) \rightarrow A(X)$
2. Predicados A, P (estado positivo) y N (estado negativo)

$$H(\_X, Y) \rightarrow J(X, Y\_)$$

Producciones de G_{S-SAT}

Producciones agrupadas por fases

1. $S(X) \to A(X)$
2. Predicados A, P (estado positivo) y N (estado negativo)

$$H(\_X, Y) \to J(X, Y\_)$$

3. $B(X_1 d X_2, Y) \to C(X_1, Y)B(X_2, Y)$ y $B(\varepsilon, Y) \to \varepsilon$

Producciones de G_{S-SAT}

Producciones agrupadas por fases

1. $S(X) \rightarrow A(X)$
2. Predicados A, P (estado positivo) y N (estado negativo)

$$H(\_X, Y) \rightarrow J(X, Y\_)$$

3. $B(X_1 dX_2, Y) \rightarrow C(X_1, Y)B(X_2, Y)$ y $B(\varepsilon, Y) \rightarrow \varepsilon$
4. Predicados C, Cp (estado positivo) y Cn estado negativo

$$H(\_X, \_Y) \rightarrow J(X, Y)$$

Resultados derivados de G_{S-SAT}

- $L_{S-SAT} = L_{G_{S-SAT}}$

Resultados derivados de G_{S-SAT}

- $L_{S-SAT} = L_{G_{S-SAT}}$
- Problema de la palabra G_{S-SAT} no es polinomial

Resultados derivados de G_{S-SAT}

- $L_{S-SAT} = L_{G_{S-SAT}}$
- Problema de la palabra G_{S-SAT} no es polinomial
- No es necesaria la transducción finita de T_{SAT} para construir L_{S-SAT}

Resultados derivados de G_{S-SAT}

- $L_{S-SAT} = L_{G_{S-SAT}}$
- Problema de la palabra G_{S-SAT} no es polinomial
- No es necesaria la transducción finita de T_{SAT} para construir L_{S-SAT}
- Las RCG reconocen todos los problemas de la clase NP

Contenido

Recomendaciones

- Otro formalismo que genere $L_{0,1,d}$ (cerrado bajo transducción finita)

Recomendaciones

- Otro formalismo que genere $L_{0,1,d}$ (cerrado bajo transducción finita)
- Todo formalismo que genere $L_{0,1,d}$ tiene un tamaño $O(1)$

Recomendaciones

- Otro formalismo que genere $L_{0,1,d}$ (cerrado bajo transducción finita)
- Todo formalismo que genere $L_{0,1,d}$ tiene un tamaño $O(1)$
- Tipo de formalismo se obtiene al aplicarle el transductor T_{SAT} a $G_{0,1,d}$

Recomendaciones

- Otro formalismo que genere $L_{0,1,d}$ (cerrado bajo transducción finita)
- Todo formalismo que genere $L_{0,1,d}$ tiene un tamaño $O(1)$
- Tipo de formalismo se obtiene al aplicarle el transductor T_{SAT} a $G_{0,1,d}$
- Por qué las RCG no sean cerradas bajo transducción finita?

Recomendaciones

- Otro formalismo que genere $L_{0,1,d}$ (cerrado bajo transducción finita)
- Todo formalismo que genere $L_{0,1,d}$ tiene un tamaño $O(1)$
- Tipo de formalismo se obtiene al aplicarle el transductor T_{SAT} a $G_{0,1,d}$
- Por qué las RCG no sean cerradas bajo transducción finita?
- RCG que reconozca los SAT solubles en tiempo polinomial (2-SAT)

Una aproximación al lenguaje de todas las fórmulas booleanas satisfacibles

Raudel Alejando Gómez Molina

Facultad de Matemática y Computación
Universidad de La Habana

22 de febrero de 2025

Tutor: MSc. Fernando Raul Rodriguez Flores