Una aproximación al lenguaje de todas las fórmulas booleanas satisfacibles

Raudel Alejandro Gómez Molina

Facultad de Matemática y Computación
Universidad de La Habana

16 de noviembre de 2025

Tutor: MSc. Fernando Raul Rodriguez Flores

Teoría de lenguajes (Conceptos)

Alfabeto

Un alfabeto es un conjunto finito de símbolos

$$\Sigma = \{0, 1\}$$

Teoría de lenguajes (Conceptos)

Alfabeto

Un alfabeto es un conjunto finito de símbolos

$$\Sigma = \{0, 1\}$$

Cadena

Una cadena es una sucesión finita de símbolos del alfabeto

$$w = 0001111$$

Teoría de lenguajes (Conceptos)

Alfabeto

Un alfabeto es un conjunto finito de símbolos

$$\Sigma = \{0, 1\}$$

Cadena

Una cadena es una sucesión finita de símbolos del alfabeto

$$w = 0001111$$

Lenguaje

Un lenguaje es un conjunto finito de cadenas

$$L = \{0^n 1^m \mid n, m \in \mathbb{N}\}$$

Teoría de lenguajes (Problemas)

Problema de la palabra

- Determinar si una cadena pertenece a un lenguaje ¿w pertenece a L?

Teoría de lenguajes (Problemas)

Problema de la palabra

- Determinar si una cadena pertenece a un lenguaje ¿w pertenece a L?
- $10001 \in \{1w \mid w \in \{0,1\}^*\}$

Teoría de lenguajes (Problemas)

Problema de la palabra

- Determinar si una cadena pertenece a un lenguaje ¿w pertenece a L?
- $10001 \in \{1w \mid w \in \{0,1\}^*\}$
- $00001 \notin \{1w \mid w \in \{0,1\}^*\}$

Teoría de lenguajes (Problemas)

Problema de la palabra

- Determinar si una cadena pertenece a un lenguaje ¿w pertenece a L?
- $10001 \in \{1w \mid w \in \{0,1\}^*\}$
- $00001 \notin \{1w \mid w \in \{0,1\}^*\}$

Problema en Ciencia de la Computación

- Todo problema se puede reducir a un problema de la palabra

Teoría de lenguajes (Problemas)

Problema de la palabra

- Determinar si una cadena pertenece a un lenguaje ¿w pertenece a L?
- $10001 \in \{1w \mid w \in \{0,1\}^*\}$
- $00001 \notin \{1w \mid w \in \{0,1\}^*\}$

Problema en Ciencia de la Computación

- 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

- Determinar si una cadena pertenece a un lenguaje ¿w pertenece a L?
- $10001 \in \{1w \mid w \in \{0,1\}^*\}$
- $00001 \notin \{1w \mid w \in \{0,1\}^*\}$

Problema en Ciencia de la Computación

- Todo problema se puede reducir a un problema de la palabra
- Todo problema se puede codificar como lenguaje formal
- Determinar si dos números son primos relativos

Teoría de lenguajes (Problemas)

Problema de la palabra

- Determinar si una cadena pertenece a un lenguaje ¿w pertenece a L?
- $10001 \in \{1w \mid w \in \{0,1\}^*\}$
- $00001 \notin \{1w \mid w \in \{0,1\}^*\}$

Problema en Ciencia de la Computación

- Todo problema se puede reducir a un problema de la palabra
- Todo problema se puede codificar como lenguaje formal
- Determinar si dos números son primos relativos
- Determinar si un arreglo está ordenado

Clases de problemas

- Existen problemas para los cuales no se conoce una solución eficiente

Clases de problemas

- Existen problemas para los cuales no se conoce una solución eficiente
- Comprobar si una solución es válida es eficiente (clase NP)

Clases de problemas

- Existen problemas para los cuales no se conoce una solución eficiente
- Comprobar si una solución es válida es eficiente (clase NP)
- El SAT es el primer problema demostrado como NP-Completo

Clases de problemas

- Existen problemas para los cuales no se conoce una solución eficiente
- Comprobar si una solución es válida es eficiente (clase NP)
- El SAT es el primer problema demostrado como NP-Completo
- Pertenece a la clase NP

Clases de problemas

- Existen problemas para los cuales no se conoce una solución eficiente
- Comprobar si una solución es válida es eficiente (clase NP)
- El SAT es el primer problema demostrado como NP-Completo
- Pertenece a la clase NP
- Se puede reducir a cualquier problema en NP en tiempo polinomial

Problema de la Satisfacibilidad booleana (SAT)

- Consiste en determinar si una fórmula booleana es satisfacible

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

Problema de la Satisfacibilidad booleana (SAT)

- Consiste en determinar si una fórmula booleana es satisfacible

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

- Existen instancias polinomiales 2-SAT, Horn-SAT y XOR-SAT

Estructura de la presentación

Definir y construir el lenguaje de todas las fórmulas booleanas satisfacibles

Estructura de la presentación

Definir y construir el lenguaje de todas las fórmulas booleanas satisfacibles

Definir L_{S-SAT}

- Codificar una fórmula booleana
- Definir L_{S-SAT}

Estructura de la presentación

Definir y construir el lenguaje de todas las fórmulas booleanas satisfacibles

Definir L_{S-SAT}

- Codificar una fórmula booleana
- Definir L_{S-SAT}

Construir L_{S-SAT} mediante transducción finita

- Asignar valores a una fórmula booleana mediante una cadena
- Construir L_{S-SAT} mediante un transductor finito

Estructura de la presentación

Definir y construir el lenguaje de todas las fórmulas booleanas satisfacibles

Definir L_{S-SAT}

- Codificar una fórmula booleana
- Definir L_{S-SAT}

Construir L_{S-SAT} mediante transducción finita

- Asignar valores a una fórmula booleana mediante una cadena
- Construir L_{S-SAT} mediante un transductor finito

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
- Asumimos que las fórmulas booleanas están en CNF

Estados de una variable en una cláusula

$$(x_1 \vee \neg x_3)$$

Estados de una variable en una cláusula

$$(x_1 \vee \neg x_3)$$

- x_1 está sin negar en la cláusula
- x_3 está negada en la cláusula

Estados de una variable en una cláusula

$$(x_1 \vee \neg x_3)$$

- 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

$$x_1 \vee \neg x_3$$

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

$$x_1 \vee \neg x_3$$

$$(x_1$$

$$a$$

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

$$x_1 \vee \neg x_3$$

$$(x_1 \quad \not\vee \quad \not{x_2}$$

$$a \qquad\qquad c$$

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

$$x_1 \vee \neg x_3$$

$$(x_1 \quad \cancel{\vee} \quad \cancel{x_2} \quad \vee \quad \neg x_3)$$

$$a \qquad\qquad c \qquad\qquad b$$

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

$$x_1 \vee \neg x_3$$

$$(x_1 \quad \not\vee \quad \not{x_2} \quad \vee \quad \neg x_3)$$

$$a \qquad\qquad c \qquad\qquad b$$

$$acb$$

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

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

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

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

(x_1)

$acc\mathbf{d}$

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*

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

(x_1)	$\wedge$	$(x_1 \vee \neg x_2 \vee x_3)$
*acc***d**		*aba***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*

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

(x_1)	$\wedge$	$(x_1 \vee \neg x_2 \vee x_3)$	$\wedge$	$(\neg x_2 \vee x_3)$
*acc***d**		*aba***d**		*cba***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

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

$$\begin{array}{ccccc} (x_1) & \wedge & (x_1 \vee \neg x_2 \vee x_3) & \wedge & (\neg x_2 \vee x_3) \\ acc\mathbf{d} & & aba\mathbf{d} & & cba\mathbf{d} \end{array}$$

$$acc\mathbf{d}aba\mathbf{d}cba\mathbf{d}$$

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

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

$$x_1 \wedge x_2 \wedge x_3$$

$$acc\mathbf{d}cac\mathbf{d}cca\mathbf{d} \in L_{S-SAT}$$

Estructura de la presentación

Definir y construir el lenguaje de todas las fórmulas booleanas satisfacibles

Definir L_{S-SAT}

- Codificar una fórmula booleana
- Definir L_{S-SAT}

Construir L_{S-SAT} mediante transducción finita

- Asignar valores a una fórmula booleana mediante una cadena
- Construir L_{S-SAT} mediante un transductor finito

Asignar valores a una cláusula mediante una cadena binaria

- Se tiene una cadena q que representa una cláusula

- $q = acb$

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

- $q = acb$
- $w = 101$

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 $|q| = |w|$

- $q = acb$
- $w = 101$

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 es 0, $x_i = \mathit{false}$

- $q = acb$
- $w = 101$
- $x_1 = \mathit{true}$, $x_2 = \mathit{false}$ y $x_3 = \mathit{true}$

Asignar valores a una cláusula mediante una cadena binaria

$$w = 101 \qquad q = acb \;\Leftrightarrow\; C = (x_1 \not\vee \not{x_2} \vee \neg x_3)$$

Asignar valores a una cláusula mediante una cadena binaria

$$w = 101 \qquad q = acb \Leftrightarrow C = (x_1 \not\vee \not x_2 \vee \neg x_3)$$

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

Asignar valores a una cláusula mediante una cadena binaria

$$w = 101 \qquad q = acb \Leftrightarrow C = (x_1 \not\vee \not{x_2} \vee \neg x_3)$$

- $w_1 = 1 \Rightarrow x_1 = \mathit{true}$ 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

$$w = 101 \qquad q = acb \Leftrightarrow C = (x_1 \not\vee \not{x_2} \vee \neg x_3)$$

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

Asignar valores a una fórmula mediante una cadena

- w (cadena binaria) es la asignación de valores para una cláusula

- $w = 101$

Asignar valores a una fórmula mediante una cadena

- w (cadena binaria) es la asignación de valores para una cláusula
- Si la fórmula booleana F tiene n cláusulas

- $w = 101$
- $e = acc\mathbf{d}aba\mathbf{d}cba\mathbf{d}$

Asignar valores a una fórmula mediante una cadena

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

- $w = 101$
- $e = acc\mathbf{d}aba\mathbf{d}cba\mathbf{d}$
- $r = 101\mathbf{d}101\mathbf{d}101\mathbf{d}$

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

Asignar valores a una fórmula mediante una cadena

- w (cadena binaria) es la asignación de valores para una cláusula
- Si la fórmula booleana F tiene n cláusulas
- $(wd)^n$ representa la asignación de variables para F
- $L_{0,1,d} = \{(wd)^+ \mid w \in \{0,1\}^+\}$ lenguaje de todas las interpretaciones

- $w = 101$
- $e = acc\mathbf{d}aba\mathbf{d}cba\mathbf{d}$
- $r = 101\mathbf{d}101\mathbf{d}101\mathbf{d}$

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

Transductor finito

- Transductor finito

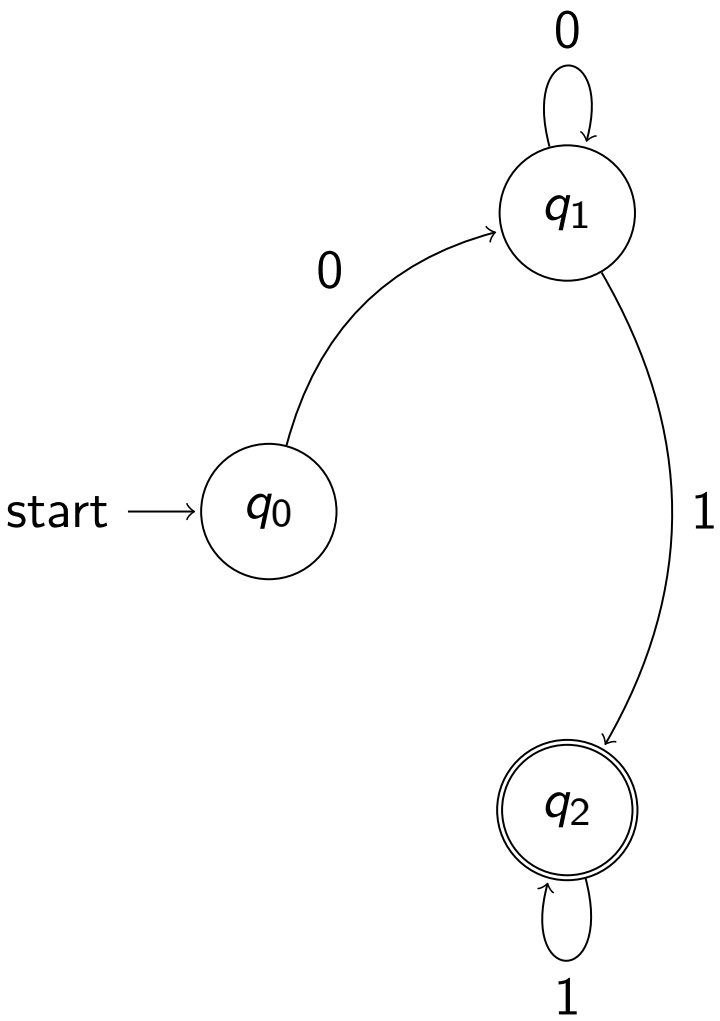
Transductor finito

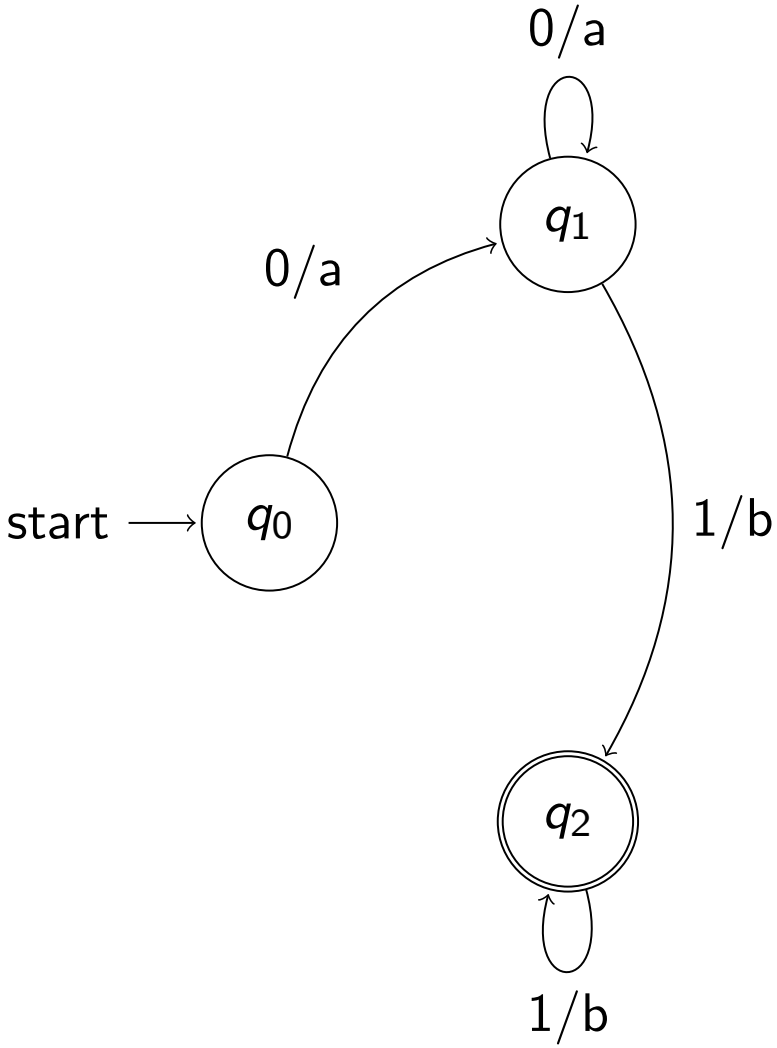
- Transductor finito
- En cada transición se lee y se escribe un símbolo

Transductor finito

- Transductor finito
- En cada transición se lee y se escribe un símbolo
- Se reconoce y se escribe una cadena

Transductor finito





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
- Si se lee un 1 y se escribe una a (positiva)

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

- Cadena binaria w generar todas las cláusulas satisfacibles por w
- Si se lee un 1 y se escribe una a (positiva)
- Si se lee un 0 y se escribe una a (negativa)

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

- Cadena binaria w generar todas las cláusulas satisfacibles por w
- Si se lee un 1 y se escribe una a (positiva)
- Si se lee un 0 y se escribe una a (negativa)
- Si se lee un 1 y se escribe una b (negativa)

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

- Cadena binaria w generar todas las cláusulas satisfacibles por w
- Si se lee un 1 y se escribe una a (positiva)
- Si se lee un 0 y se escribe una a (negativa)
- Si se lee un 1 y se escribe una b (negativa)
- Si se lee un 0 y se escribe una b (positiva)

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

- Cadena binaria w generar todas las cláusulas satisfacibles por w
- Si se lee un 1 y se escribe una a (positiva)
- Si se lee un 0 y se escribe una a (negativa)
- Si se lee un 1 y se escribe una b (negativa)
- Si se lee un 0 y se escribe una b (positiva)
- Si se lee cualquier cosa y se escribe una c (negativa)

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

- Cadena binaria w generar todas las cláusulas satisfacibles por w
- Si se lee un 1 y se escribe una a (positiva)
- Si se lee un 0 y se escribe una a (negativa)
- Si se lee un 1 y se escribe una b (negativa)
- Si se lee un 0 y se escribe una b (positiva)
- Si se lee cualquier cosa y se escribe una c (negativa)
- Transductor T_{CLAUSE}

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

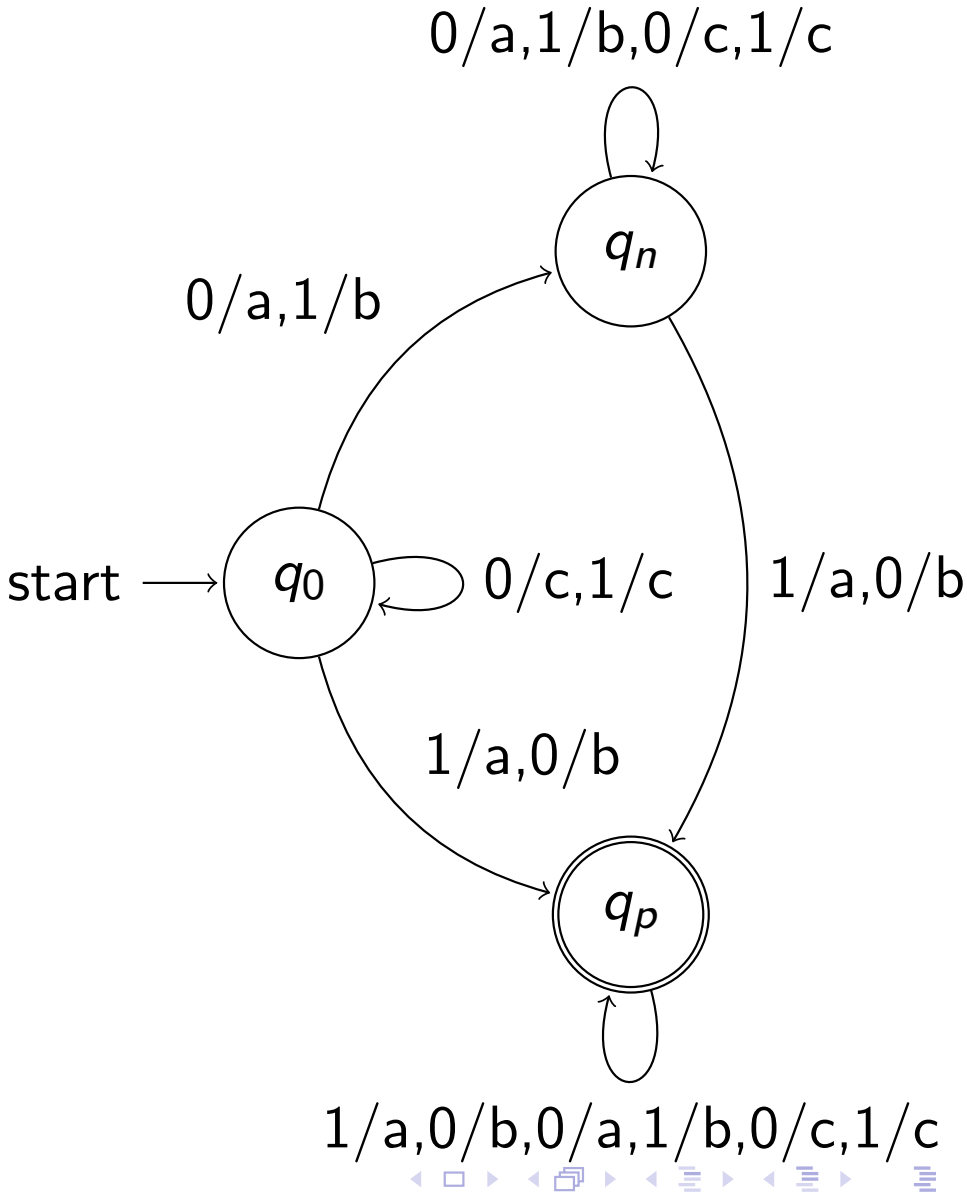
Transductor T_{CLAUSE}

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



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

- 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 $r \in L_{0,1,d}$ 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 $r \in L_{0,1,d}$ generar todas las cadenas $e \in L_{FULL-SAT}$
- e representa una fórmula booleana satisfacible por r
- Transductor T_{SAT}

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

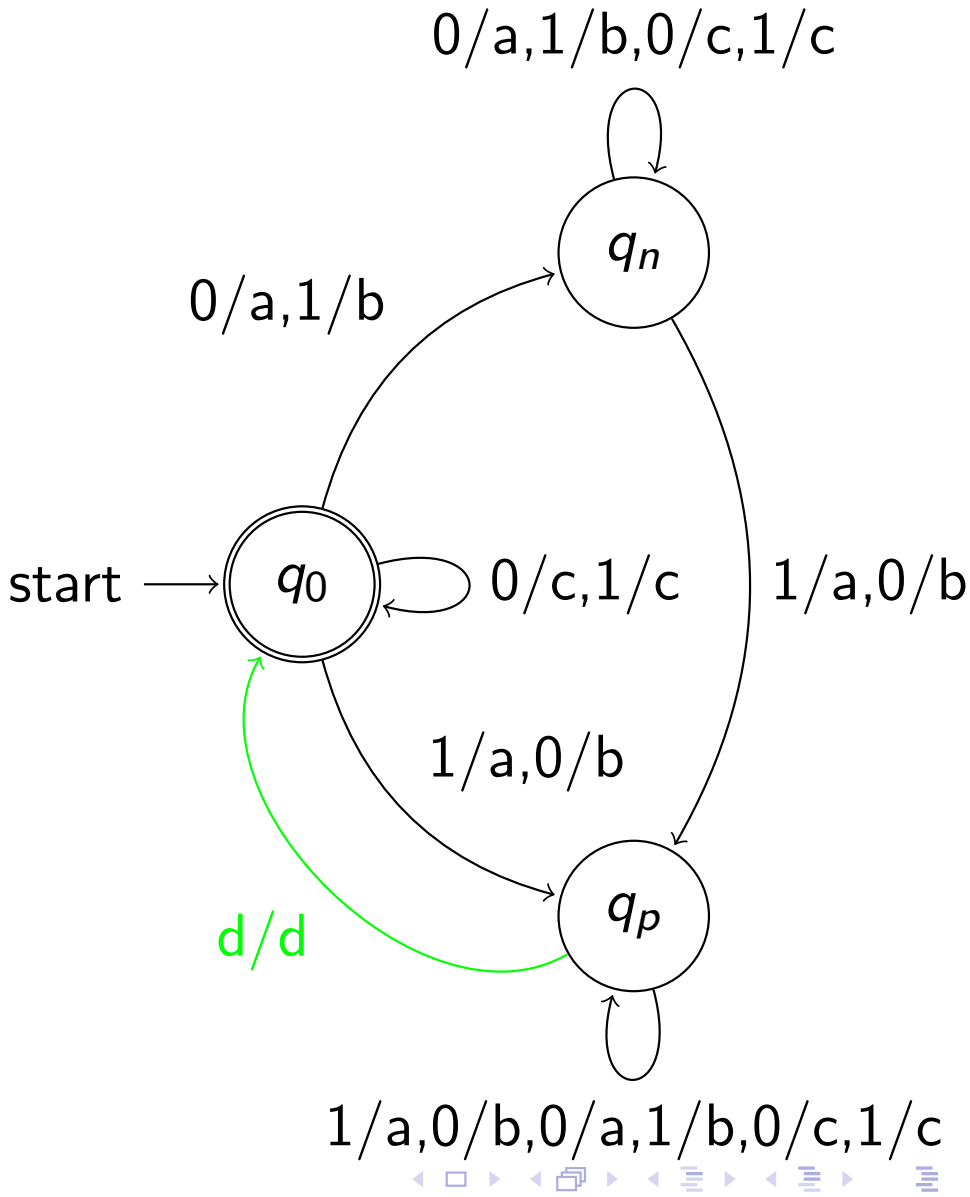
Transductor T_{SAT}

Entrada y Salida

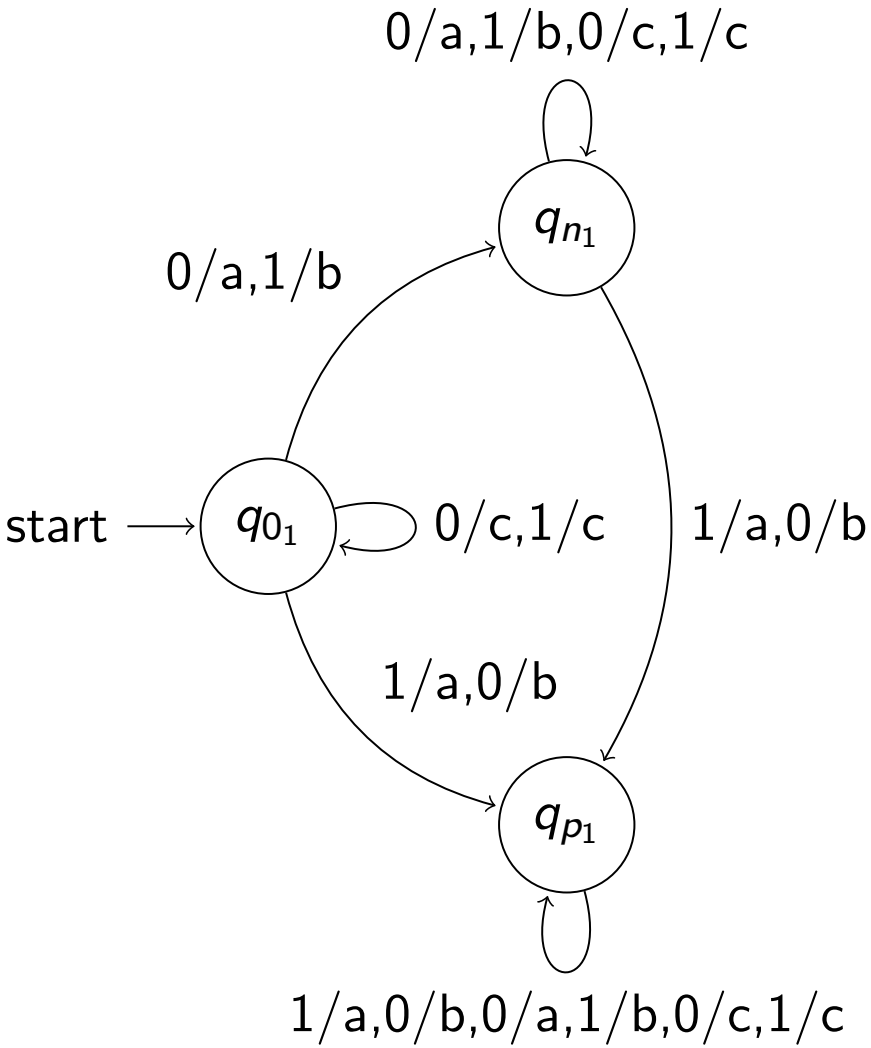
- Cadena $r \in L_{0,1,d}$
- Todas las fórmulas satisfacibles por r

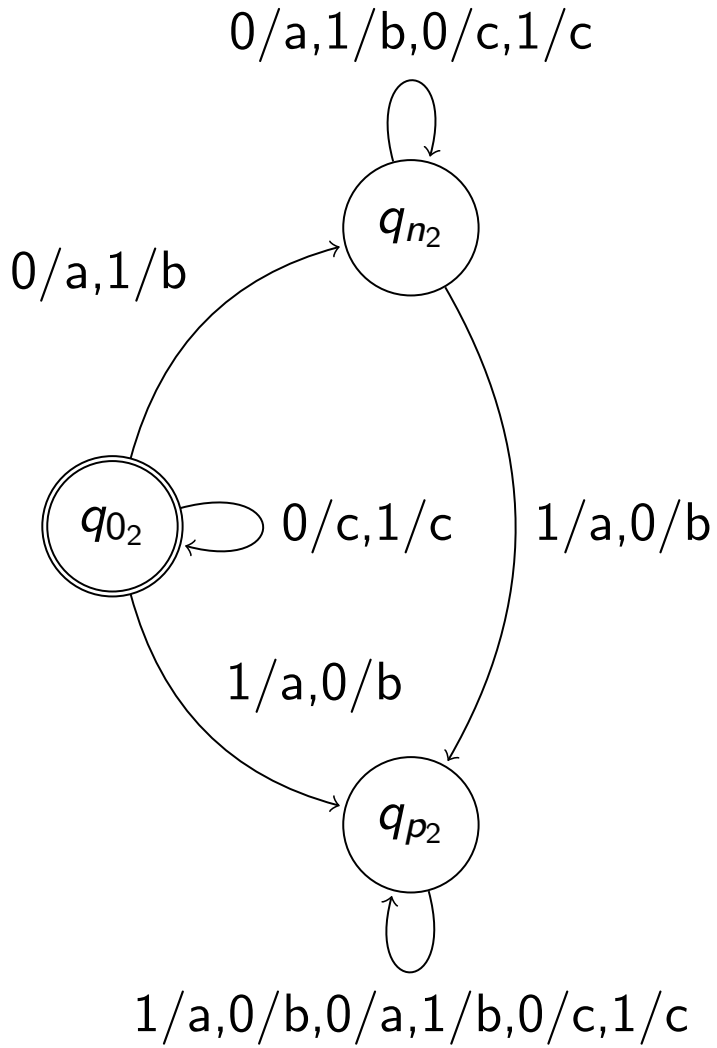
Estados

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

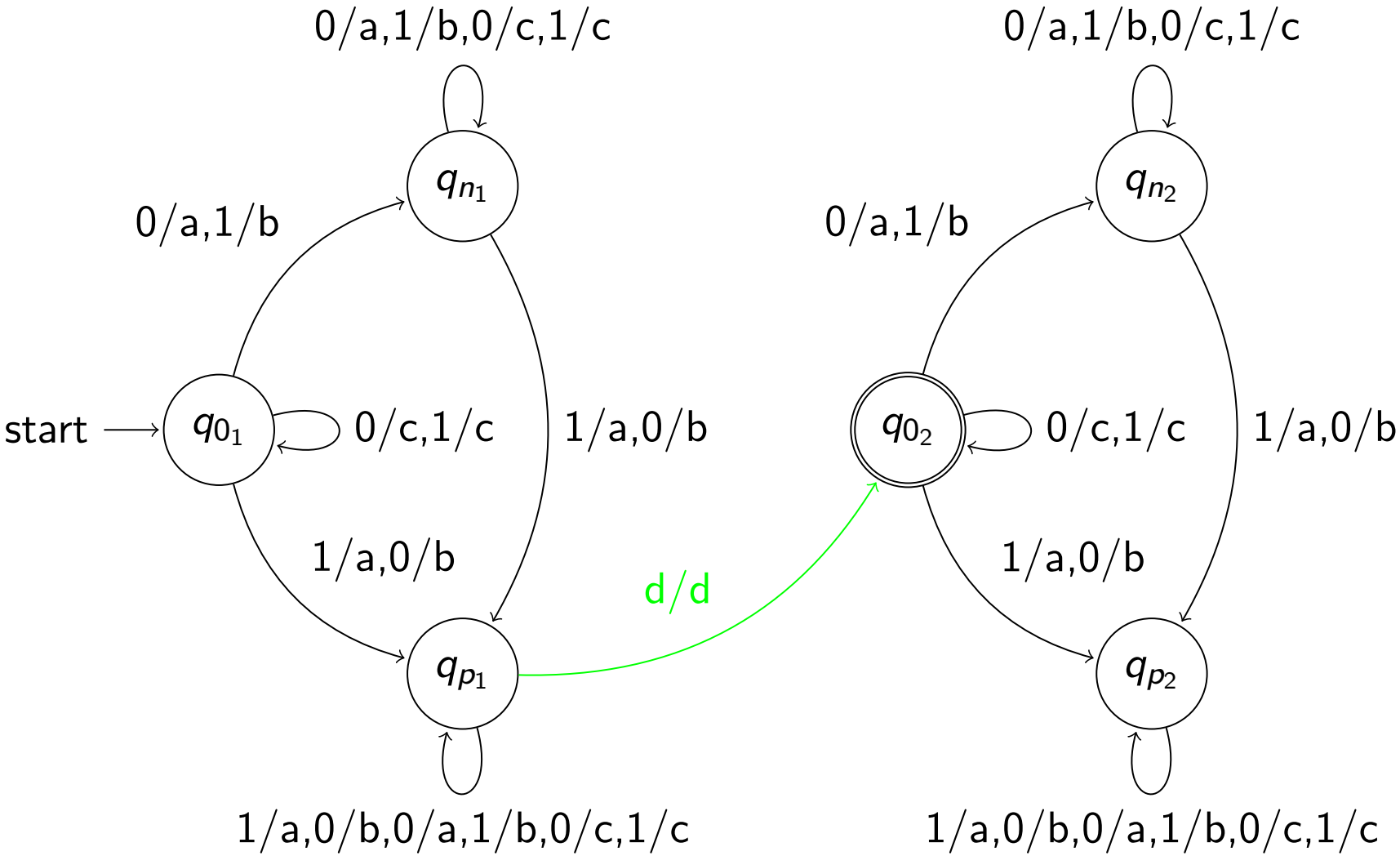


Transductor T_{SAT}

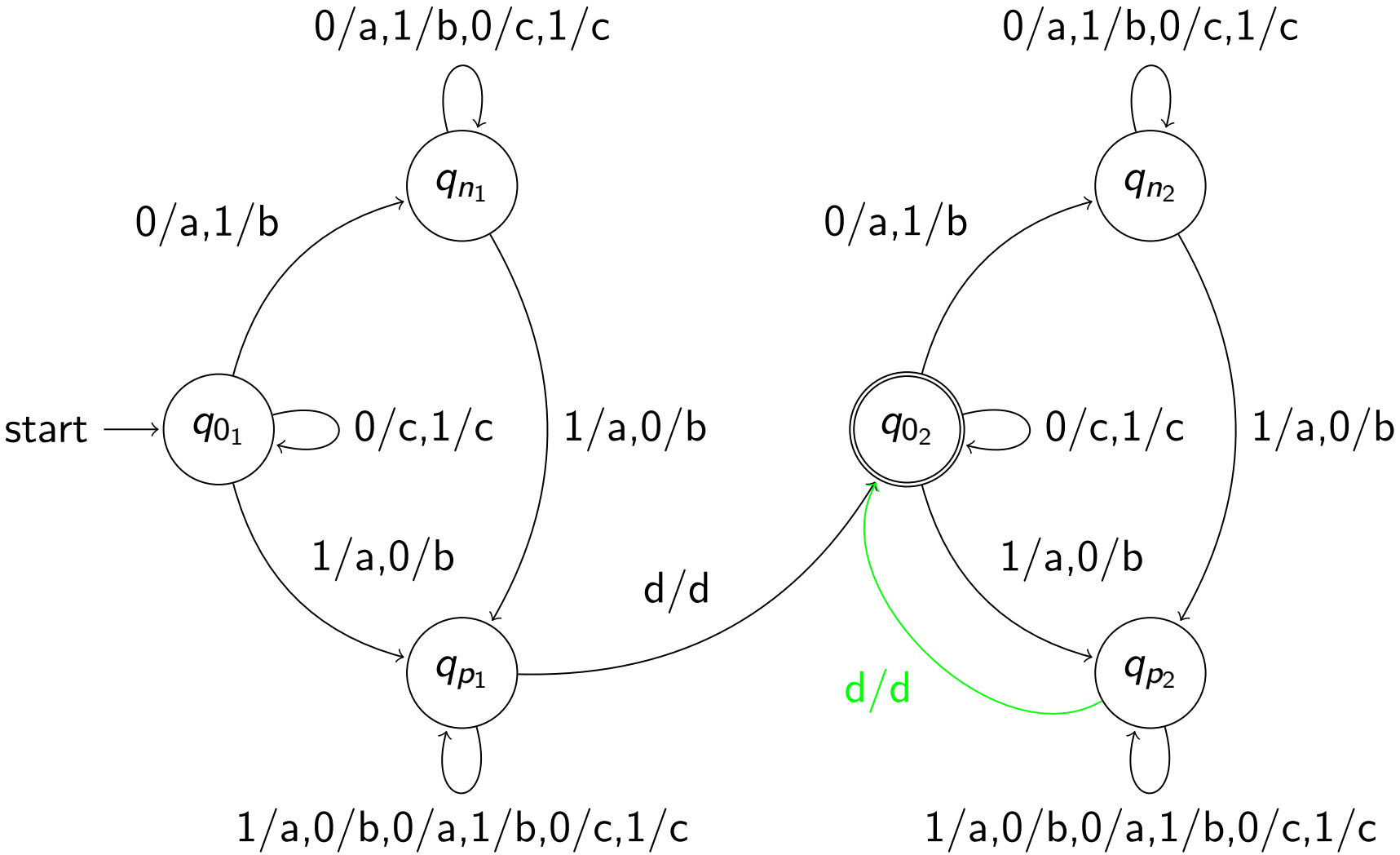




Transductor T_{SAT}



Transductor T_{SAT}



Resultados derivados de T_{SAT}

Lenguaje de todas las fórmulas booleanas satisfacibles

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

Resultados derivados de T_{SAT}

Lenguaje de todas las fórmulas booleanas satisfacibles

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

Construcción de L_{S-SAT}

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

Resultados derivados de T_{SAT}

Lenguaje de todas las fórmulas booleanas satisfacibles

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

Construcción de L_{S-SAT}

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

Resultados derivados de T_{SAT}

Lenguaje de todas las fórmulas booleanas satisfacibles

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

Construcción de L_{S-SAT}

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

Resultados

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

Resultados derivados de T_{SAT}

Lenguaje de todas las fórmulas booleanas satisfacibles

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

Construcción de L_{S-SAT}

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

Resultados

- 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}

Lenguaje de todas las fórmulas booleanas satisfacibles

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

Construcción de L_{S-SAT}

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

Resultados

- 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}

Lenguaje de todas las fórmulas booleanas satisfacibles

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

Construcción de L_{S-SAT}

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

Resultados

- 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)

Conclusiones

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

Conclusiones

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

Conclusiones

- 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

Conclusiones

- 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)

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 $O(1)$

Una aproximación al lenguaje de todas las fórmulas booleanas satisfacibles

Raudel Alejandro Gómez Molina

Facultad de Matemática y Computación
Universidad de La Habana

16 de noviembre de 2025

Tutor: MSc. Fernando Raul Rodriguez Flores