NP-completitud y el teorema de Cook–Levin



UNIVERSIDAD DE CHILE

Arturo Merino

Plan

Recuerdo y plan

- **Recuerdo:**

Recuerdo y plan

- **Recuerdo:**
  - P: decidir en tiempo polinomial

Recuerdo y plan

- **Recuerdo:**
  - P: decidir en tiempo polinomial
  - NP: verificar en tiempo polinomial

Recuerdo y plan

- **Recuerdo:**
  - P: decidir en tiempo polinomial
  - NP: verificar en tiempo polinomial

- **Plan:**

Recuerdo y plan

- **Recuerdo:**
  - P: decidir en tiempo polinomial
  - NP: verificar en tiempo polinomial

- **Plan:**
  - Reducciones (polinomiales)

Recuerdo y plan

- **Recuerdo:**
  - P: decidir en tiempo polinomial
  - NP: verificar en tiempo polinomial

- **Plan:**
  - Reducciones (polinomiales)
  - NP-completitud

Recuerdo y plan

- **Recuerdo:**
  - P: decidir en tiempo polinomial
  - NP: verificar en tiempo polinomial
- **Plan:**
  - Reducciones (polinomiales)
  - NP-completitud
  - El teorema de Cook–Levin

Reducciones polinomiales

Comparar dificultades

- Nos gustaría usar un algoritmo para B para resolver A

Comparar dificultades

- Nos gustaría usar un algoritmo para B para resolver A
- Nos gustaría que la reducción preservara "eficiencia"

Comparar dificultades

- Nos gustaría usar un algoritmo para B para resolver A
- Nos gustaría que la reducción preservara "eficiencia"

Def.

Sean $A, B \subseteq \Sigma^*$. Una *reducción polinomial* de A a B es una función $f : \Sigma^* \to \Sigma^*$ computable en tiempo polinomial tal que

$$\forall w \in \Sigma^*, \qquad w \in A \iff f(w) \in B.$$

Si existe tal función, escribimos $A \leq_{\mathsf{P}} B$.

Comparar dificultades

- Nos gustaría usar un algoritmo para B para resolver A
- Nos gustaría que la reducción preservara "eficiencia"

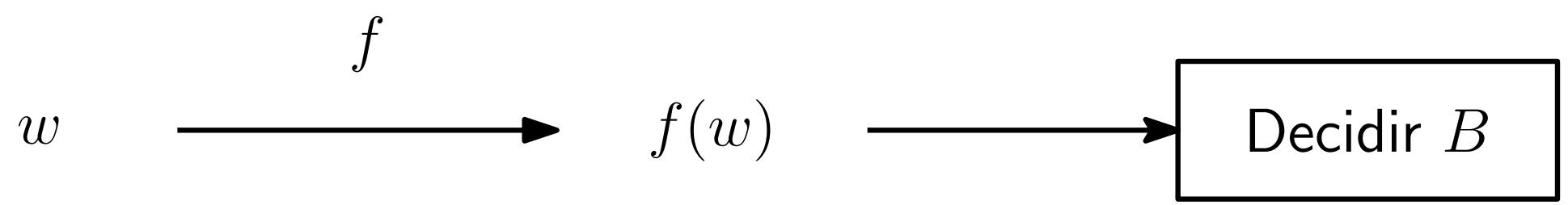
Def.

Sean $A, B \subseteq \Sigma^*$. Una *reducción polinomial* de A a B es una función $f : \Sigma^* \to \Sigma^*$ computable en tiempo polinomial tal que

$$\forall w \in \Sigma^*, \qquad w \in A \iff f(w) \in B.$$

Si existe tal función, escribimos $A \leq_{\mathsf{P}} B$.

Diremos en ese caso que A *se reduce* a B.



Comparar dificultades

- Nos gustaría usar un algoritmo para B para resolver A
- Nos gustaría que la reducción preservara "eficiencia"

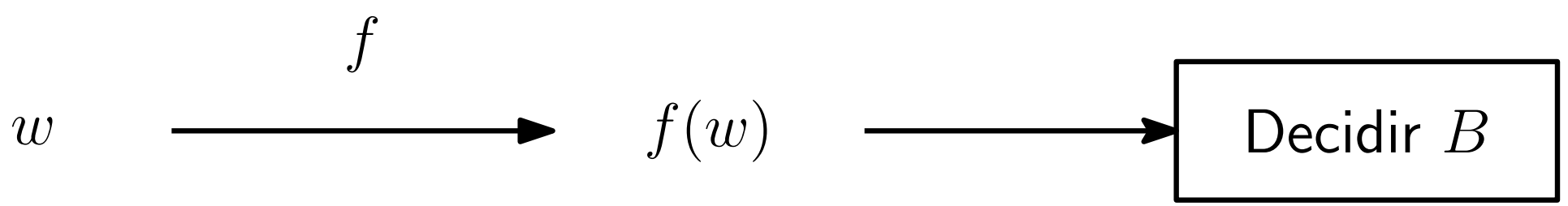
Def.

Sean $A, B \subseteq \Sigma^*$. Una *reducción polinomial* de A a B es una función $f : \Sigma^* \to \Sigma^*$ computable en tiempo polinomial tal que

$$\forall w \in \Sigma^*, \qquad w \in A \iff f(w) \in B.$$

Si existe tal función, escribimos $A \leq_{\mathsf{P}} B$.

Diremos en ese caso que A *se reduce* a B.



- Decido A en base a esa respuesta.

Reducciones polinomiales

- ¿Qué podemos concluir de $A \leq_{\mathsf{P}} B$?

Reducciones polinomiales

- ¿Qué podemos concluir de $A \leq_{\mathsf{P}} B$?

Prop.

Sean $A, B, C \subseteq \Sigma^*$.

- Si $A \leq_{\mathsf{P}} B$ y $B \in \mathsf{P}$, entonces $A \in \mathsf{P}$.
- Si $A \leq_{\mathsf{P}} B$ y $B \leq_{\mathsf{P}} C$, entonces $A \leq_{\mathsf{P}} C$.

Reducciones polinomiales

- ¿Qué podemos concluir de $A \leq_{\mathsf{P}} B$?

Prop.

Sean $A, B, C \subseteq \Sigma^*$.
- Si $A \leq_{\mathsf{P}} B$ y $B \in \mathsf{P}$, entonces $A \in \mathsf{P}$.
- Si $A \leq_{\mathsf{P}} B$ y $B \leq_{\mathsf{P}} C$, entonces $A \leq_{\mathsf{P}} C$.

- **Dem:**

Un primer ejemplo

- **Ej:** Grafos simples no dirigidos. Recordemos CLIQUE

Un primer ejemplo

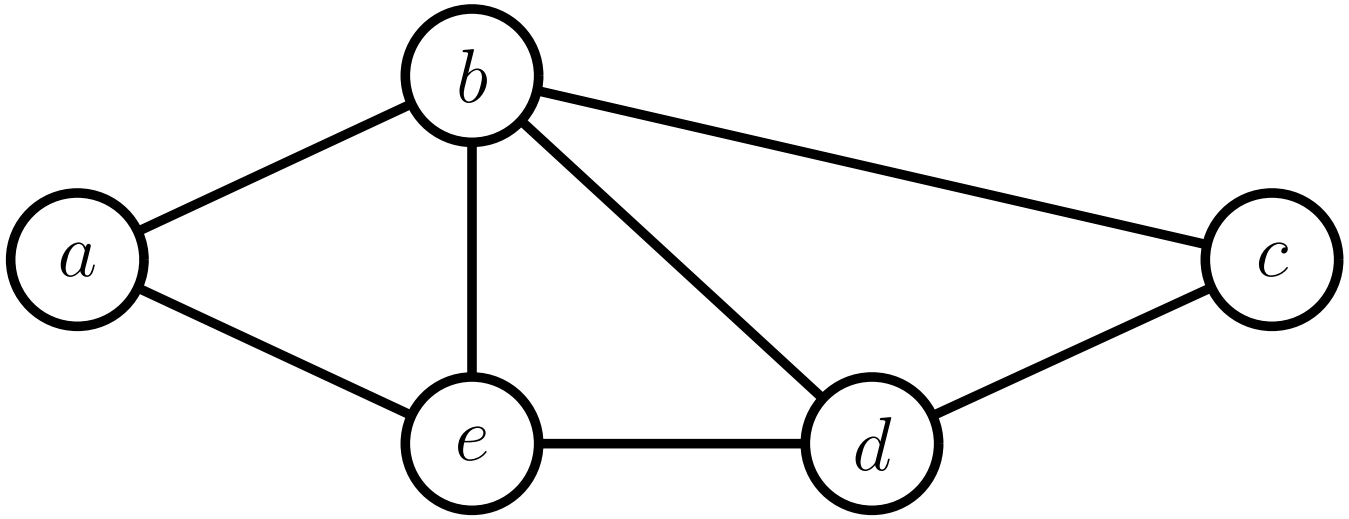
- **Ej:** Grafos simples no dirigidos. Recordemos CLIQUE
  - CLIQUE$= \{\langle G, k\rangle : G$ tiene un clique de tamaño $k\}$

Un primer ejemplo

- **Ej:** Grafos simples no dirigidos. Recordemos CLIQUE
  - CLIQUE$= \{\langle G, k\rangle : G$ tiene un clique de tamaño $k\}$
  - Un *conjunto independiente* no tiene dos vértices adyacentes.

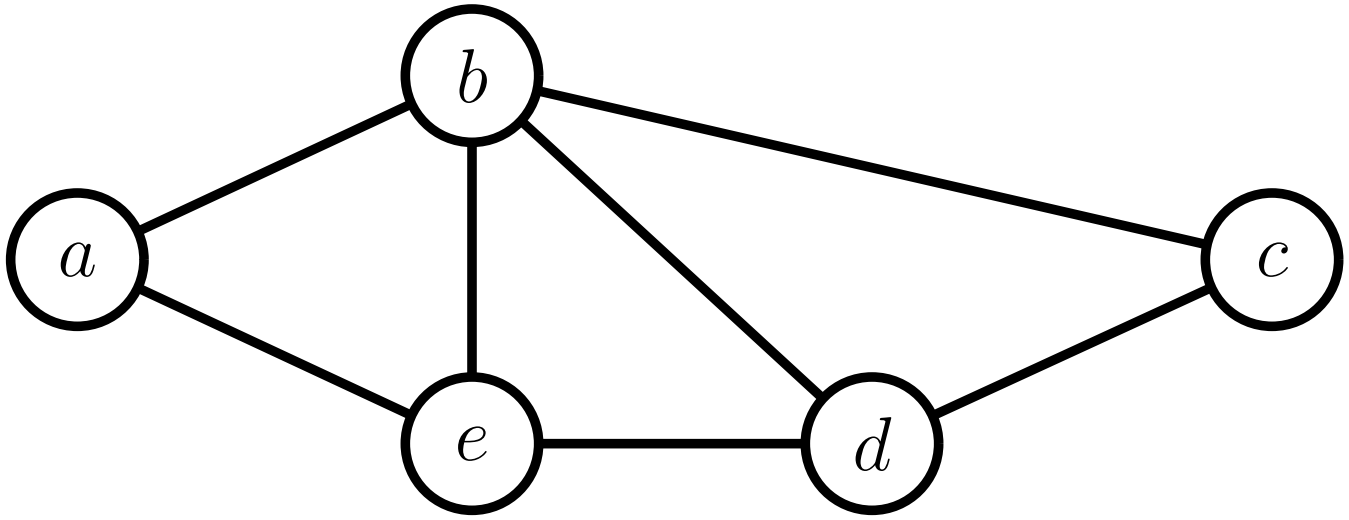
Un primer ejemplo

- **Ej:** Grafos simples no dirigidos. Recordemos CLIQUE
  - CLIQUE$= \{\langle G, k\rangle : G$ tiene un clique de tamaño $k\}$
  - Un *conjunto independiente* no tiene dos vértices adyacentes.
  - $\mathrm{IS} = \{\langle G, k\rangle : G$ tiene un conjunto independiente de tamaño $k\}$.



Un primer ejemplo

- **Ej:** Grafos simples no dirigidos. Recordemos CLIQUE
  - CLIQUE$= \{\langle G, k\rangle : G$ tiene un clique de tamaño $k\}$
  - Un *conjunto independiente* no tiene dos vértices adyacentes.
  - $\mathrm{IS} = \{\langle G, k\rangle : G$ tiene un conjunto independiente de tamaño $k\}$.



- ¿Cómo reducir CLIQUE a IS?

NP-completitud

Más difícil que NP

- ¿Hay un problema que sea tan difícil como todos los de NP?

Más difícil que NP

- ¿Hay un problema que sea tan difícil como todos los de NP?

Def.

Sea $B \subseteq \Sigma^*$. Decimos que B es *NP-difícil* si

$$\forall A \in \mathsf{NP}, \qquad A \leq_{\mathsf{P}} B.$$

Decimos que B es *NP-completo* si es NP-difícil y $B \in \mathsf{NP}$.

Más difícil que NP

- ¿Hay un problema que sea tan difícil como todos los de NP?

Def.

Sea $B \subseteq \Sigma^*$. Decimos que B es *NP-difícil* si

$$\forall A \in \mathsf{NP}, \qquad A \leq_{\mathsf{P}} B.$$

Decimos que B es *NP-completo* si es NP-difícil y $B \in \mathsf{NP}$.

- Para resolver cualquier $A \in \mathsf{NP}$, bastaría con resolver B

Más difícil que NP

- ¿Hay un problema que sea tan difícil como todos los de NP?

Def.

Sea $B \subseteq \Sigma^*$. Decimos que B es *NP-difícil* si

$$\forall A \in \mathsf{NP}, \qquad A \leq_{\mathsf{P}} B.$$

Decimos que B es *NP-completo* si es NP-difícil y $B \in \mathsf{NP}$.

- Para resolver cualquier $A \in \mathsf{NP}$, bastaría con resolver B

- NP-difícil

Más difícil que NP

- ¿Hay un problema que sea tan difícil como todos los de NP?

Def.

Sea $B \subseteq \Sigma^*$. Decimos que B es *NP-difícil* si

$$\forall A \in \mathsf{NP}, \qquad A \leq_{\mathsf{P}} B.$$

Decimos que B es *NP-completo* si es NP-difícil y $B \in \mathsf{NP}$.

- Para resolver cualquier $A \in \mathsf{NP}$, bastaría con resolver B
- NP-difícil
  - Problemas igual o más difíciles que NP.

Más difícil que NP

- ¿Hay un problema que sea tan difícil como todos los de NP?

Def.

Sea $B \subseteq \Sigma^*$. Decimos que B es *NP-difícil* si

$$\forall A \in \mathsf{NP}, \qquad A \leq_{\mathsf{P}} B.$$

Decimos que B es *NP-completo* si es NP-difícil y $B \in \mathsf{NP}$.

- Para resolver cualquier $A \in \mathsf{NP}$, bastaría con resolver B
- NP-difícil
  - Problemas igual o más difíciles que NP.
- NP-completo

Más difícil que NP

- ¿Hay un problema que sea tan difícil como todos los de NP?

Def.

Sea $B \subseteq \Sigma^*$. Decimos que B es *NP-difícil* si

$$\forall A \in \mathsf{NP}, \qquad A \leq_{\mathsf{P}} B.$$

Decimos que B es *NP-completo* si es NP-difícil y $B \in \mathsf{NP}$.

- Para resolver cualquier $A \in \mathsf{NP}$, bastaría con resolver B
- NP-difícil
  - Problemas igual o más difíciles que NP.
- NP-completo
  - Los problemas "más difíciles de NP"

Consecuencias

Prop.

Si B es NP-completo, entonces

$$B \in \mathsf{P} \iff \mathsf{P} = \mathsf{NP}.$$

Consecuencias

Prop.

Si B es NP-completo, entonces

$$B \in \mathsf{P} \iff \mathsf{P} = \mathsf{NP}.$$

- **Dem:**

Consecuencias

Prop.

Si B es NP-completo, entonces

$$B \in \mathsf{P} \iff \mathsf{P} = \mathsf{NP}.$$

- **Dem:**

- Para resolver P vs NP

Consecuencias

Prop.

Si B es NP-completo, entonces

$$B \in \mathsf{P} \iff \mathsf{P} = \mathsf{NP}.$$

- **Dem:**

- Para resolver P vs NP
  - Basta con mostrar que un problema NP-completo está en P o no.

Consecuencias

Prop.

Si B es NP-completo, entonces

$$B \in \mathsf{P} \iff \mathsf{P} = \mathsf{NP}.$$

- **Dem:**

- Para resolver P vs NP
  - Basta con mostrar que un problema NP-completo está en P o no.

- Los problemas NP-completos permiten encontrar otros problemas NP-completos.

Consecuencias

Prop.

Si B es NP-completo, entonces

$$B \in \mathsf{P} \iff \mathsf{P} = \mathsf{NP}.$$

- **Dem:**

- Para resolver P vs NP
  - Basta con mostrar que un problema NP-completo está en P o no.

- Los problemas NP-completos permiten encontrar otros problemas NP-completos.

Teo.

Si A es NP-completo, $B \in \mathsf{NP}$ y $A \leq_{\mathsf{P}} B$, entonces B es NP-completo.

Consecuencias

Prop.

Si B es NP-completo, entonces

$$B \in \mathsf{P} \iff \mathsf{P} = \mathsf{NP}.$$

- **Dem:**

- Para resolver P vs NP
  - Basta con mostrar que un problema NP-completo está en P o no.

- Los problemas NP-completos permiten encontrar otros problemas NP-completos.

Teo.

Si A es NP-completo, $B \in \mathsf{NP}$ y $A \leq_{\mathsf{P}} B$, entonces B es NP-completo.

- **Dem:**

El teorema de Cook–Levin

NP-completitud

- ¿Existen problemas NP-completos?

NP-completitud

- ¿Existen problemas NP-completos?
  - Recordemos $\mathrm{SAT} = \{\langle\varphi\rangle : \varphi \text{ es satisfacible}\}$.

NP-completitud

- ¿Existen problemas NP-completos?
  - Recordemos $\mathrm{SAT} = \{\langle\varphi\rangle : \varphi \text{ es satisfacible}\}$.

Teo.

Cook–Levin. El lenguaje SAT es NP-completo.

NP-completitud

- ¿Existen problemas NP-completos?
  - Recordemos $\mathrm{SAT} = \{\langle\varphi\rangle : \varphi \text{ es satisfacible}\}$.

Teo.

Cook–Levin. El lenguaje SAT es NP-completo.

- **Consecuencias:**

NP-completitud

- ¿Existen problemas NP-completos?
  - Recordemos $\mathrm{SAT} = \{\langle\varphi\rangle : \varphi \text{ es satisfacible}\}$.

Teo.

Cook–Levin. El lenguaje SAT es NP-completo.

- **Consecuencias:**
  - Si uno cree P$\neq$NP, entonces uno cree que SAT$\notin P$.

NP-completitud

- ¿Existen problemas NP-completos?
  - Recordemos $\mathrm{SAT} = \{\langle\varphi\rangle : \varphi \text{ es satisfacible}\}$.

Teo.

Cook–Levin. El lenguaje SAT es NP-completo.

- **Consecuencias:**
  - Si uno cree P$\neq$NP, entonces uno cree que SAT$\notin P$.
  - Díficultad *condicional*.

NP-completitud

- ¿Existen problemas NP-completos?
  - Recordemos $\mathrm{SAT} = \{\langle\varphi\rangle : \varphi \text{ es satisfacible}\}$.

Teo.

Cook–Levin. El lenguaje SAT es NP-completo.

- **Consecuencias:**
  - Si uno cree P$\neq$NP, entonces uno cree que SAT$\notin P$.
  - Díficultad *condicional*.

- **Dem:**

NP-completitud

- ¿Existen problemas NP-completos?
  - Recordemos $\mathrm{SAT} = \{\langle\varphi\rangle : \varphi \text{ es satisfacible}\}$.

Teo.

Cook–Levin. El lenguaje SAT es NP-completo.

- **Consecuencias:**
  - Si uno cree P$\neq$NP, entonces uno cree que SAT$\notin P$.
  - Díficultad *condicional*.

- **Dem:**

- Ya sabemos que $\mathrm{SAT} \in$ NP. ¿Y la NP-dificultad?

Historial de computación

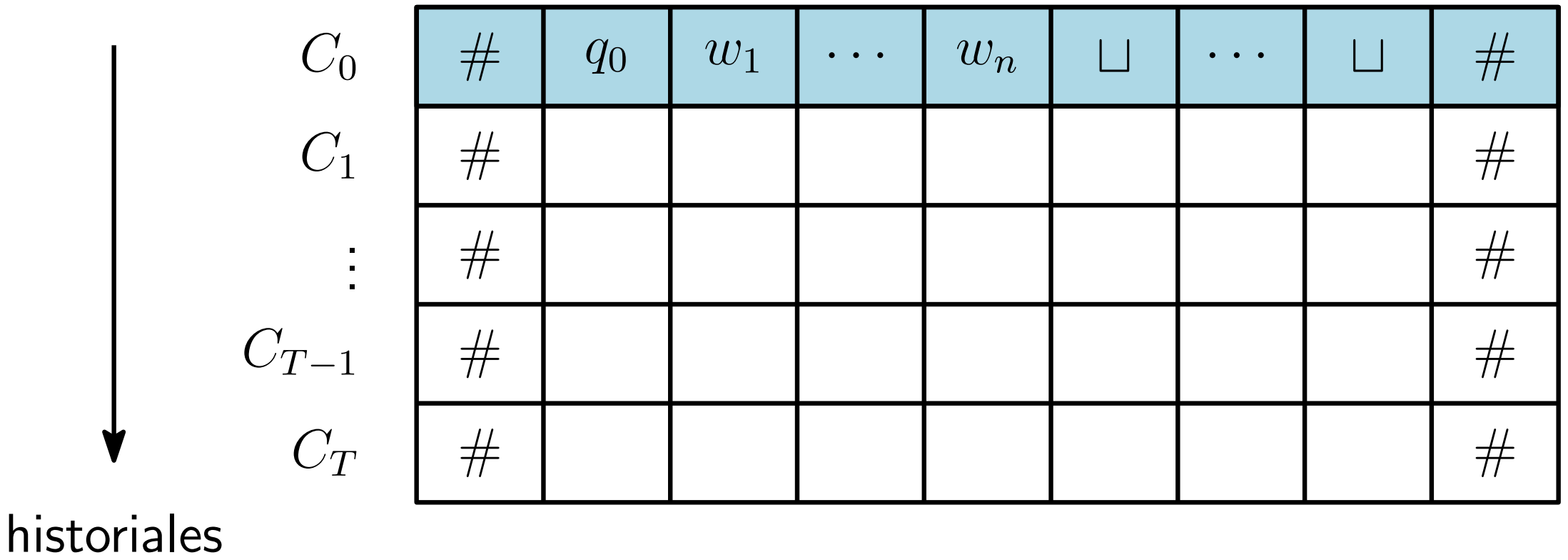
- Sea $A \in$ NP y M una MTND de una cinta que decide A.

Historial de computación

- Sea $A \in$ NP y M una MTND de una cinta que decide A.
  - Toma a lo más $T = p(|w|)$ pasos, con $T \geq |w| + 1$.

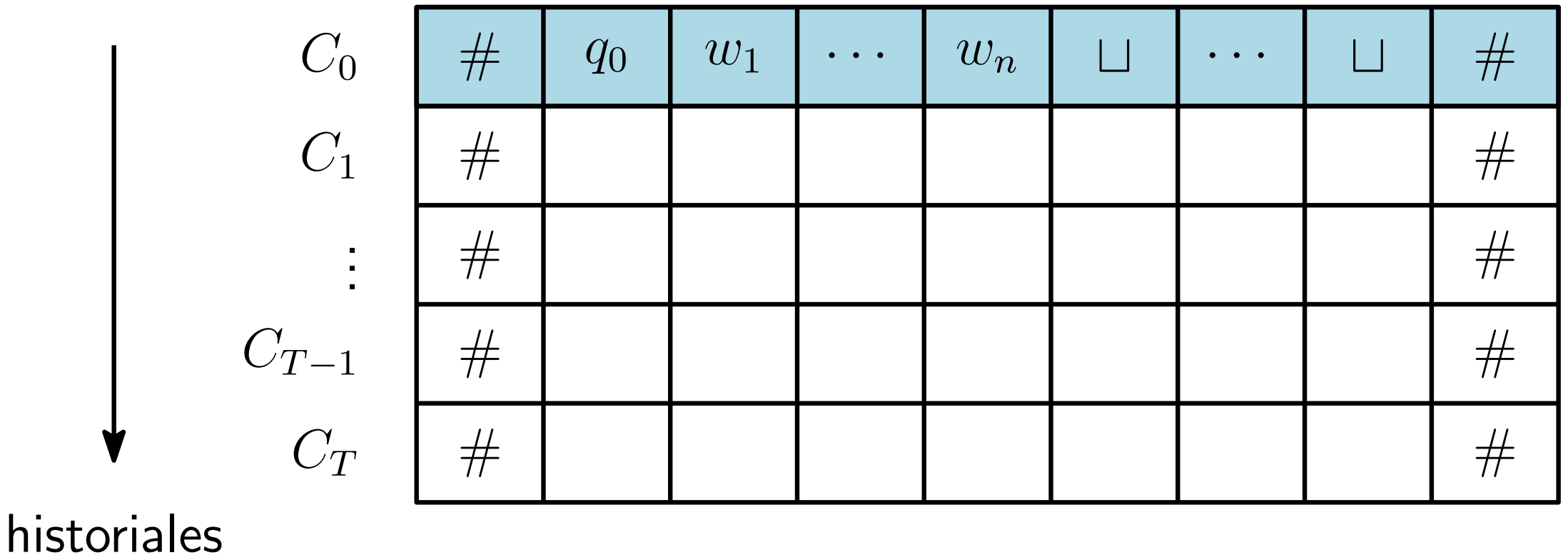
Historial de computación

- Sea $A \in$ NP y M una MTND de una cinta que decide A.
  - Toma a lo más $T = p(|w|)$ pasos, con $T \geq |w| + 1$.
- Una rama: $T + 1$ filas y $O(T)$ columnas



Historial de computación

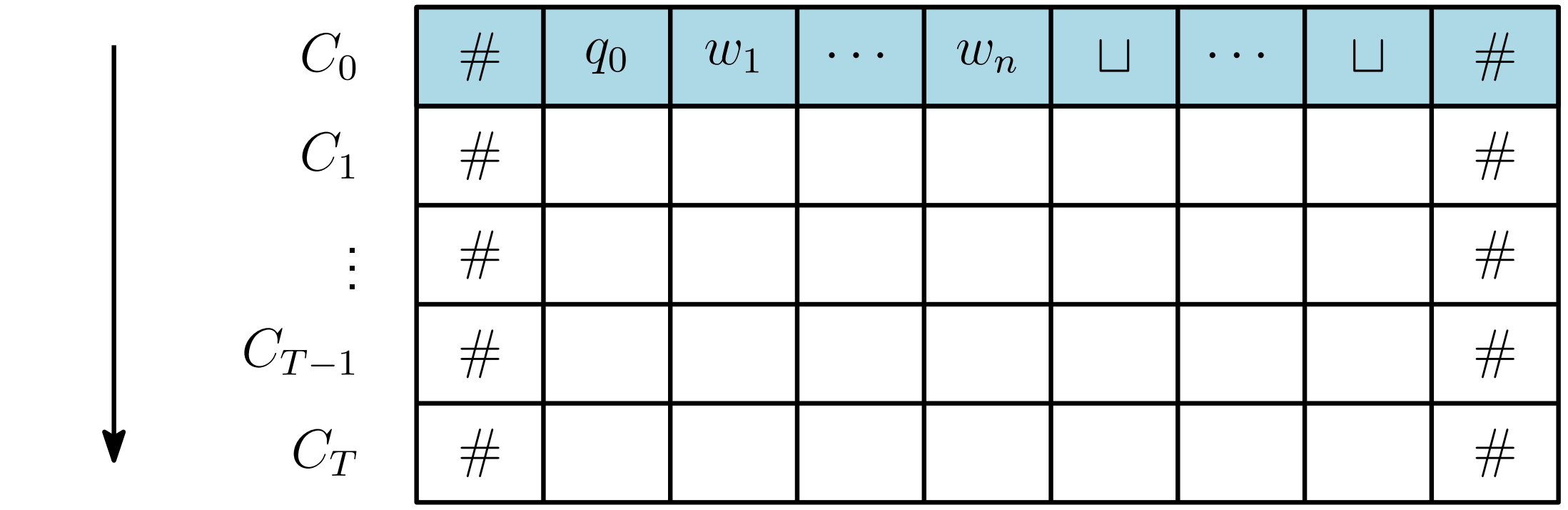
- Sea $A \in$ NP y M una MTND de una cinta que decide A.
  - Toma a lo más $T = p(|w|)$ pasos, con $T \geq |w| + 1$.
- Una rama: $T + 1$ filas y $O(T)$ columnas



  - El historial tiene $O(T^2)$ celdas.

Historial de computación

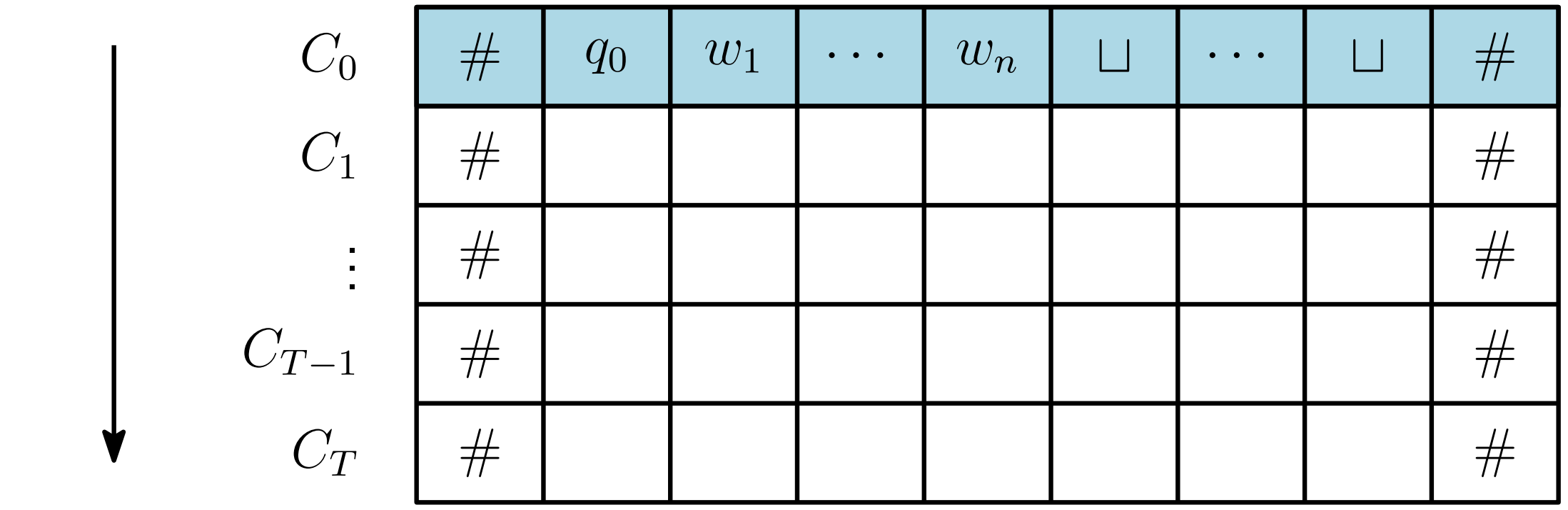
- Sea $A \in$ NP y M una MTND de una cinta que decide A.
  - Toma a lo más $T = p(|w|)$ pasos, con $T \geq |w| + 1$.
- Una rama: $T + 1$ filas y $O(T)$ columnas



  - El historial tiene $O(T^2)$ celdas.
  - Detalles: bordes con $\#$, repetir configs. si toma menos que T.

Historial de computación

- Sea $A \in$ NP y M una MTND de una cinta que decide A.
  - Toma a lo más $T = p(|w|)$ pasos, con $T \geq |w| + 1$.
- Una rama: $T + 1$ filas y $O(T)$ columnas



  - El historial tiene $O(T^2)$ celdas.
  - Detalles: bordes con #, repetir configs. si toma menos que T.
  - **Idea:** codificar historial válido con $x_{i,j,s} \leftarrow$ la tabla en (i, j) tiene s.

Chequear la válidez de la tabla

- Verificamos cada paso revisando todas las ventanas de 2×3

Chequear la válidez de la tabla

- Verificamos cada paso revisando todas las ventanas de 2×3
  - Como M está fija, hay un número constante de patrones posibles.

$$j-1 \qquad j \qquad j+1$$

Chequear la válidez de la tabla

- Verificamos cada paso revisando todas las ventanas de 2×3
  - Como M está fija, hay un número constante de patrones posibles.

	$j-1$	j	$j+1$
i	q_0	0	1
$i+1$	1	q_1	1

- **Ej:** $(q_1, 1, R) \in \delta(q_0, 0)$

Chequear la válidez de la tabla

- Verificamos cada paso revisando todas las ventanas de 2×3
  - Como M está fija, hay un número constante de patrones posibles.

	$j-1$	j	$j+1$
i	q_0	0	1
$i+1$	1	q_1	1

- **Ej:** $(q_1, 1, R) \in \delta(q_0, 0)$

  - $x_{i,j,s}$: la celda (i, j) contiene s.

Chequear la válidez de la tabla

- Verificamos cada paso revisando todas las ventanas de 2×3
  - Como M está fija, hay un número constante de patrones posibles.

	$j-1$	j	$j+1$
i	q_0	0	1
$i+1$	1	q_1	1

- **Ej:** $(q_1, 1, R) \in \delta(q_0, 0)$

  - $x_{i,j,s}$: la celda (i, j) contiene s.

$$\psi_{i,j} = x_{i,j-1,q_0} \wedge x_{i,j,0} \wedge x_{i,j+1,1} \wedge x_{i+1,j-1,1} \wedge x_{i+1,j,q_1} \wedge x_{i+1,j+1,1}.$$

Chequear la válidez de la tabla

- Verificamos cada paso revisando todas las ventanas de 2×3
  - Como M está fija, hay un número constante de patrones posibles.

	$j-1$	j	$j+1$
i	q_0	0	1
$i+1$	1	q_1	1

- **Ej:** $(q_1, 1, R) \in \delta(q_0, 0)$

  - $x_{i,j,s}$: la celda (i, j) contiene s.

$$\psi_{i,j} = x_{i,j-1,q_0} \wedge x_{i,j,0} \wedge x_{i,j+1,1} \wedge x_{i+1,j-1,1} \wedge x_{i+1,j,q_1} \wedge x_{i+1,j+1,1}.$$

- Disyunción de los patrones permitidos: tamaño constante.

Chequear la válidez de la tabla

- Verificamos cada paso revisando todas las ventanas de 2×3
  - Como M está fija, hay un número constante de patrones posibles.

	$j-1$	j	$j+1$
i	q_0	0	1
$i+1$	1	q_1	1

- **Ej:** $(q_1, 1, R) \in \delta(q_0, 0)$

  - $x_{i,j,s}$: la celda (i, j) contiene s.

$$\psi_{i,j} = x_{i,j-1,q_0} \wedge x_{i,j,0} \wedge x_{i,j+1,1} \wedge x_{i+1,j-1,1} \wedge x_{i+1,j,q_1} \wedge x_{i+1,j+1,1}.$$

- Disyunción de los patrones permitidos: tamaño constante.
- Sólo hay un número polinomial de ventanas que revisar.

Chequear la válidez de la tabla

- Verificamos cada paso revisando todas las ventanas de 2×3
  - Como M está fija, hay un número constante de patrones posibles.

	$j-1$	j	$j+1$
i	q_0	0	1
$i+1$	1	q_1	1

- **Ej:** $(q_1, 1, R) \in \delta(q_0, 0)$

  - $x_{i,j,s}$: la celda (i, j) contiene s.

$$\psi_{i,j} = x_{i,j-1,q_0} \wedge x_{i,j,0} \wedge x_{i,j+1,1} \wedge x_{i+1,j-1,1} \wedge x_{i+1,j,q_1} \wedge x_{i+1,j+1,1}.$$

- Disyunción de los patrones permitidos: tamaño constante.
- Sólo hay un número polinomial de ventanas que revisar.
- Hay que chequear además el inicio y que acepta.

Del historial a fórmula

- Codifiquemos cada celda con variables booleanas

Del historial a fórmula

- Codifiquemos cada celda con variables booleanas
  - Una variable $x_{i,j,s}$: la celda (i, j) contiene s.

Del historial a fórmula

- Codifiquemos cada celda con variables booleanas
  - Una variable $x_{i,j,s}$: la celda (i, j) contiene s.

$$\varphi = \varphi_{\text{celda}} \wedge \varphi_{\text{inicio}} \wedge \varphi_{\text{paso}} \wedge \varphi_{\text{acepta}}.$$

Del historial a fórmula

- Codifiquemos cada celda con variables booleanas
  - Una variable $x_{i,j,s}$: la celda (i, j) contiene s.

$$\varphi = \varphi_{\text{celda}} \wedge \varphi_{\text{inicio}} \wedge \varphi_{\text{paso}} \wedge \varphi_{\text{acepta}}.$$

  - φ_{celda}: exactamente un símbolo por celda.

Del historial a fórmula

- Codifiquemos cada celda con variables booleanas
  - Una variable $x_{i,j,s}$: la celda (i, j) contiene s.

$$\varphi = \varphi_{\text{celda}} \wedge \varphi_{\text{inicio}} \wedge \varphi_{\text{paso}} \wedge \varphi_{\text{acepta}}.$$

  - φ_{celda}: exactamente un símbolo por celda.
  - φ_{inicio}: la primera fila describe el inicio sobre w.

Del historial a fórmula

- Codifiquemos cada celda con variables booleanas
  - Una variable $x_{i,j,s}$: la celda (i, j) contiene s.

$$\varphi = \varphi_{\text{celda}} \wedge \varphi_{\text{inicio}} \wedge \varphi_{\text{paso}} \wedge \varphi_{\text{acepta}}.$$

  - φ_{celda}: exactamente un símbolo por celda.
  - φ_{inicio}: la primera fila describe el inicio sobre w.
  - φ_{paso}: cada fila es sucesora legal de la anterior.

Del historial a fórmula

- Codifiquemos cada celda con variables booleanas
  - Una variable $x_{i,j,s}$: la celda (i, j) contiene s.

$$\varphi = \varphi_{\text{celda}} \wedge \varphi_{\text{inicio}} \wedge \varphi_{\text{paso}} \wedge \varphi_{\text{acepta}}.$$

  - φ_{celda}: exactamente un símbolo por celda.
  - φ_{inicio}: la primera fila describe el inicio sobre w.
  - φ_{paso}: cada fila es sucesora legal de la anterior.
  - φ_{acepta}: la última fila es de aceptación.

Del historial a fórmula

- Codifiquemos cada celda con variables booleanas
  - Una variable $x_{i,j,s}$: la celda (i, j) contiene s.

$$\varphi = \varphi_{\text{celda}} \wedge \varphi_{\text{inicio}} \wedge \varphi_{\text{paso}} \wedge \varphi_{\text{acepta}}.$$

  - φ_{celda}: exactamente un símbolo por celda.
  - φ_{inicio}: la primera fila describe el inicio sobre w.
  - φ_{paso}: cada fila es sucesora legal de la anterior.
  - φ_{acepta}: la última fila es de aceptación.
- Construimos φ en tiempo polinomial.

Del historial a fórmula

- Codifiquemos cada celda con variables booleanas
  - Una variable $x_{i,j,s}$: la celda (i, j) contiene s.

$$\varphi = \varphi_{\text{celda}} \wedge \varphi_{\text{inicio}} \wedge \varphi_{\text{paso}} \wedge \varphi_{\text{acepta}}.$$

  - φ_{celda}: exactamente un símbolo por celda.
  - φ_{inicio}: la primera fila describe el inicio sobre w.
  - φ_{paso}: cada fila es sucesora legal de la anterior.
  - φ_{acepta}: la última fila es de aceptación.
- Construimos φ en tiempo polinomial.

$$w \in A \iff \varphi \text{ es satisfacible.}$$

Del historial a fórmula

- Codifiquemos cada celda con variables booleanas
  - Una variable $x_{i,j,s}$: la celda (i, j) contiene s.

$$\varphi = \varphi_{\text{celda}} \wedge \varphi_{\text{inicio}} \wedge \varphi_{\text{paso}} \wedge \varphi_{\text{acepta}}.$$

  - φ_{celda}: exactamente un símbolo por celda.
  - φ_{inicio}: la primera fila describe el inicio sobre w.
  - φ_{paso}: cada fila es sucesora legal de la anterior.
  - φ_{acepta}: la última fila es de aceptación.
- Construimos φ en tiempo polinomial.

$$w \in A \iff \varphi \text{ es satisfacible.}$$

- **Concluyamos.**

Dudas/Preguntas

Wrap-up

- **Hoy:**

Wrap-up

- **Hoy:**
    - Reducciones (polinomiales)

Wrap-up

- **Hoy:**
  - Reducciones (polinomiales)
  - NP-completitud

Wrap-up

- **Hoy:**
  - Reducciones (polinomiales)
  - NP-completitud
  - El teorema de Cook–Levin

Wrap-up

- **Hoy:**
  - Reducciones (polinomiales)
  - NP-completitud
  - El teorema de Cook–Levin

- **Próx. clase:**

Wrap-up

- **Hoy:**
  - Reducciones (polinomiales)
  - NP-completitud
  - El teorema de Cook–Levin

- **Próx. clase:**
  - Más problemas NP-completos