.UBAfiuba
FACULTAD DE INGENIERÍA

Teorema Levin-Cook

Teoría de Algoritmos I (75.29 / 95.06)
Ing. Víctor Daniel Podberezski
vpodberezski@fi.uba.ar

Teorema Levin Cook

Este teorema permite

Identificar al problema SAT como el primer lenguaje NP-Completo

En primer lugar

Debemos demostrar que SAT $\in$NP

Luego que todo lenguaje $\in$NP se puede reducir polinomialmente a SAT

Probar que SAT $\in$NP

Es sencillo, se puede construir una MT se puede relizar un verificador polinomial para hacerlo usando un certificado.

Planteo

Sea

A un lenguaje cualquier perteneciente a NP

W el input de tamaño n

N una MT no determinística que decida A en n^k -3 pasos para alguna constante k

Podemos ver que

N se va ramificando y procesando.

Cada rama se puede ver

como una sucesión de configuraciones

que parten de la configuración inicial

Transicionan a configuraciones “legales”

Tabla de transición de configuraciones

Podemos representar una rama de ejecución

En una tabla donde cada fila representa una configuración

Por conveniencia agregamos

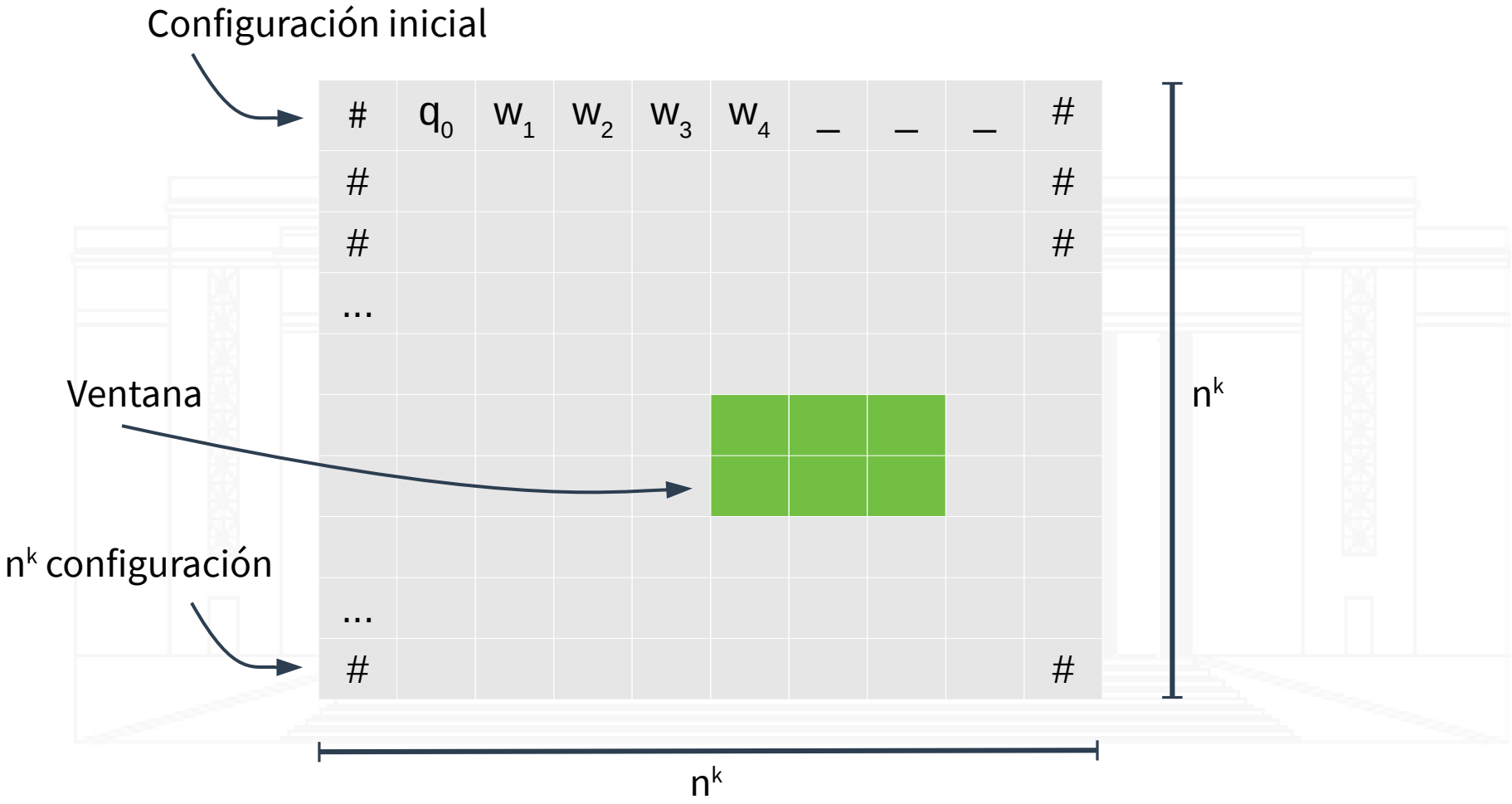
El símbolo “#” que usaremos para delimitar cada una de nuestras configuraciones

La cantidad de columnas de la tabla

la definimos en n^k

Puesto que N decide en n^k -3 pasos, el cabezal no puede pasar de n^k - 3 posiciones como mucho

Tabla – representación



Problema equivalente

Una tabla representa una computación de una rama que acepta

Si cualquiera de sus celdas contiene un estado de q_{accept}

Por lo tanto, el problema de determinar si N acepta w

Es equivalente a saber si existe una tabla que acepta w para N

Reducción polinomial a SAT

Definiremos la reducción polinomial f de A a SAT

Para un input w produciremos una instancia φ de SAT

Sea

Q el set de estados de N

El Γ set de símbolos del alfabeto de la cinta de N

Creamos

$C = Q \cup \Gamma \cup \{\#\}$

Cada celda de nuestra tabla

Contiene un símbolo de C

Variables y función booleana a satisfacer

Definiremos la variables booleanas

$x_{i,j,s}$ que vale 1 si en la celda[i,j] se encuentra el símbolo s, sino vale 0

Donde

i y j van de 1 a n^k

S son los símbolos de C

Con esto, construiremos la formula φ

Para que la asignación de las variables corresponda a una tabla de aceptación de N con w

Detalle de la formula

La formula φ estará conformada por la conjunción de 4 partes

$\varphi_{c\,ell}$ esta formula asegura que unicamente se asigne un símbolo a cada celda

φ_{start} esta formula asegura que la primera fila de la tabla sea la configuración inicial de N con w

φ_{move} esta formula asegura que cada fila de la tabla corresponde a una configuración que legalmente proceda a la configuración de la fila anterior

φ_{accept} esta formula asegura que en alguna de las celdas se encuentre el estado de aceptación

Entonces

$$\Phi = \varphi_{c\,ell} \wedge \varphi_{start} \wedge \varphi_{move} \wedge \varphi_{accept}$$

Φ_{start} y φ_{accept}

Con Φ_{start} buscamos que la primera fila contenga la configuración inicial

$$\varphi_{start} = X_{1,1,\#} \wedge X_{1,2,q0} \wedge X_{1,3,w1} \wedge X_{1,4,w2} \wedge \dots \wedge X_{1,n+2,wn} \wedge X_{1,n+3,\_} \wedge \dots \wedge X_{1,n^k-1,\_} \wedge X_{1,n^k,\#}$$

Con φ_{accept} buscamos que en alguna celda tenga el estado de aceptación

$$\varphi_{accept} = \bigvee_{1 \le i,j \le n^k} x_{i,j,q_{accept}}$$

Con φ_{cell} buscamos que unicamente se asigne un símbolo a cada celda

Se realiza para cada celda, verificamos que al menos 1 símbolo este en la celda y que no haya mas de 1 en ella

$$\varphi_{cell} = \wedge_{1 \leq i,j \leq n^k} \left[\left(\vee_{s \in C} x_{i,j,s} \right) \wedge \left(\vee_{s,t \in C\, s \neq t} \left(\bar{x}_{i,j,s} \vee \bar{x}_{i,j,t} \right) \right) \right]$$

φ_{move}

Con φ_{move} buscamos que cada transición entre configuraciones, es decir que cada fila, sea "legal" de acuerdo a la función de transición de N

Para hacer eso, se analiza de a bloques de la tabla.

Se verifican ventanas : un bloque de celdas de 2X3

Diremos que una ventana es legal

Si la misma no viola las acciones especificadas por la función de transición de N

Ventanas legales - ejemplos

Ejemplo:

Sean

$a,b,c \in \Gamma$ de N

$q_1,q_2 \in Q$ de N

$\delta(q_1, a) = \{(q_1,b,R)\}$ y $\delta(q_1, b) = \{(q_2,c,L), (q_2,a,R)\}$ parte de la función de transición

Las siguientes ventanas son legales entre las configuraciones

a	q1	b
q2	a	c

Corresponde a la aplicación de $\delta(q_1, b) = \{(q_2,c,L)\}$

a	q1	b
a	a	q2

Corresponde a la aplicación de $\delta(q_1, b) = \{(q_2,a,R)\}$

Ventanas legales – ejemplos (cont.)

Otras ventanas legales

#	b	a
#	b	a

Corresponde unas celdas donde la transición no afectó (contenido igual)

a	a	q1
a	a	b

Como no vemos sobre que símbolo esta el cabezal, y podría ser el caso de aplicación de $\delta(q_1, a) = \{(q_1,b,R)\}$

b	b	b
c	b	b

Como cambio la primera fila, suponemos que el cabezal esta antes. Pero no sabemos en que estado. Podría ser el caso de aplicación de $\delta(q_1, b) = \{(q_1,c,L)\}$

a	b	a
a	b	q2

no vemos sobre que símbolo esta el cabezal ni su estado, podría ser el caso de aplicación de $\delta(q_1, b) = \{(q_2,c,L)\}$

Ventanas ilegales - ejemplos

Existen, por otro lado, ventanas ilegales

a	b	a
a	a	a

No puede cambiar de “b” a “a”. No esta el cabezal ubicado antes de la celda

a	q1	b
q2	a	a

No corresponde a una transición declarada por la función de transición (seria legal si hubiese cambiado de “b” a “c”)

b	q1	b
q2	b	q2

No puede aparecer en la fila inferior 2 estados

Ventanas legales - ejemplos

Para toda ventana posible

tengo que verificar que es legal

$$\varphi_{move} = \wedge_{1 \le i \le n^k, 1 \le j \le n^k} \left(\text{la ventana (i,j) es legal}\right)$$

La reglas para saber si la ventana es legal

se tienen que crear mediante el análisis de la función de transición y agregar el caso donde no hay modificación

$$\varphi_{\text{ventana legal}} = \vee_{a_{1,\ldots},a_6} \left(x_{i,j-1,a_1} \wedge x_{i,j,a_2} \wedge x_{i,j+1,a_3} \wedge x_{i+1,j-1,a_4} \wedge x_{i+1,j,a_5} \wedge x_{i+1,j+1,a}\right.$$

Φ_{move} (cont.)

Con la función φ definida

$$\Phi = \varphi_{c\,ell} \wedge \varphi_{start} \wedge \varphi_{move} \wedge \varphi_{accept}$$

Podemos usar una MT no deterministica que resuelva SAT para resolverlo,

Si φ se puede satisfacer, entonces w pertenece al lenguaje de nuestro problema original

Podemos por lo tanto

Reducir cualquier problema perteneciente a NP a SAT

Verificación que la reducción es polinomial

Resta ver que la reducción es polinomial

Tenemos n^{2k} celdas. El numero de símbolos depende de la TM (pero lo podemos tomar como constante).

Tenemos por lo tanto $O(n^{2k})$ variables

$\varphi_{c\,ell}$ se realiza por cada celda → $O(n^{2k})$

φ_{start} se realiza con la primer fila de la tabla $O(n^{k})$

φ_{move} usa un numero constante de variables por ventana. Por n^{2k} ventanas → $O(n^{2k})$

φ_{accept} se realiza por cada celda de la tabla $O(n^{2k})$

Por lo tanto

Es una reducción polinomial. Y SAT ES NP-COMPLETO

.UBAfiuba

FACULTAD DE INGENIERÍA

Presentación realizada en Julio de 2020