Complejidad de la lógica de primer orden

IIC3263

Complejidad de la LPO

Queremos estudiar la complejidad de evaluar una consulta en LPO.

Vamos a estudiar dos nociones: Complejidad de los datos y Complejidad combinada

Complejidad de la LPO

Queremos estudiar la complejidad de evaluar una consulta en LPO.

Vamos a estudiar dos nociones: Complejidad de los datos y Complejidad combinada

Pero antes vamos a repasar algunas nociones de complejidad.

- Necesitamos más clases que PTIME y NP

Para recordar: Máquinas de Turing

Definición

Máquina de Turing (determinista): $(Q, \Sigma, q_0, \delta, F)$

- Q *es un conjunto finito de estados*
- Σ *es un alfabeto finito tal que* $\vdash, \mathtt{B} \notin \Sigma$
- $q_0 \in Q$ *es el estado inicial*
- $F \subseteq Q$ *es un conjunto de estados finales*
- δ *es una función parcial:*

$$\delta \quad : \quad Q \times (\Sigma \cup \{\vdash, \mathtt{B}\}) \to Q \times (\Sigma \cup \{\vdash, \mathtt{B}\}) \times \{\leftarrow, \square, \to\}$$

δ *es llamada función de transición*

Máquinas de Turing: Funcionamiento

La cinta de la máquina de Turing es infinita hacia la derecha.

- El símbolo $\vdash$ es usado para demarcar la posición 0 de la cinta

Máquinas de Turing: Funcionamiento

La cinta de la máquina de Turing es infinita hacia la derecha.

- El símbolo $\vdash$ es usado para demarcar la posición 0 de la cinta

Supuesto

- Si $\delta(q, \vdash)$ está definido: $\delta(q, \vdash) = (q', \vdash, X)$, con $X \in \{\square, \rightarrow\}$
- Si $a \in (\Sigma \cup \{\mathtt{B}\})$ y $\delta(q, a)$ está definido: $\delta(q, a) = (q', b, X)$, con $b \in (\Sigma \cup \{\mathtt{B}\})$

Máquinas de Turing: Funcionamiento

Σ es el alfabeto de entrada y $(\Sigma \cup \{\vdash, \mathtt{B}\})$ es el alfabeto de la cinta.

- Una palabra $w \in \Sigma^*$ de entrada de largo n es colocada en las posiciones 1, ..., n de la cinta
- Las posiciones siguientes ($n + 1$, $n + 2$, ...) contienen el símbolo B

Máquinas de Turing: Funcionamiento

Al comenzar a funcionar, la máquina se encuentra en el estado q_0 y su cabeza lectora está en la posición 1 de la cinta.

En cada instante la máquina se encuentra en un estado q y su cabeza lectora está en una posición p

- Si el símbolo en la posición p es a y $\delta(q, a) = (q', b, X)$, entonces:

Máquinas de Turing: Funcionamiento

Al comenzar a funcionar, la máquina se encuentra en el estado q_0 y su cabeza lectora está en la posición 1 de la cinta.

En cada instante la máquina se encuentra en un estado q y su cabeza lectora está en una posición p

- Si el símbolo en la posición p es a y $\delta(q, a) = (q', b, X)$, entonces:
  - La máquina escribe el símbolo b en la posición p de la cinta

Máquinas de Turing: Funcionamiento

Al comenzar a funcionar, la máquina se encuentra en el estado q_0 y su cabeza lectora está en la posición 1 de la cinta.

En cada instante la máquina se encuentra en un estado q y su cabeza lectora está en una posición p

- Si el símbolo en la posición p es a y $\delta(q, a) = (q', b, X)$, entonces:
  - La máquina escribe el símbolo b en la posición p de la cinta
  - Cambia de estado desde q a q'

Máquinas de Turing: Funcionamiento

Al comenzar a funcionar, la máquina se encuentra en el estado q_0 y su cabeza lectora está en la posición 1 de la cinta.

En cada instante la máquina se encuentra en un estado q y su cabeza lectora está en una posición p

- Si el símbolo en la posición p es a y $\delta(q, a) = (q', b, X)$, entonces:
  - La máquina escribe el símbolo b en la posición p de la cinta
  - Cambia de estado desde q a q'
  - Mueve la cabeza lectora a la posición $p - 1$ si X es $\leftarrow$, y a la posición $p + 1$ si X es $\rightarrow$. Si X es $\square$, entonces la cabeza lectora permanece en la posición p

Máquinas de Turing: Funcionamiento

Al comenzar a funcionar, la máquina se encuentra en el estado q_0 y su cabeza lectora está en la posición 1 de la cinta.

En cada instante la máquina se encuentra en un estado q y su cabeza lectora está en una posición p

- Si el símbolo en la posición p es a y $\delta(q, a) = (q', b, X)$, entonces:
  - La máquina escribe el símbolo b en la posición p de la cinta
  - Cambia de estado desde q a q'
  - Mueve la cabeza lectora a la posición $p - 1$ si X es $\leftarrow$, y a la posición $p + 1$ si X es $\rightarrow$. Si X es $\square$, entonces la cabeza lectora permanece en la posición p

La máquina acepta w si se detiene en un estado final

El lenguaje aceptado por una máquina de Turing

Definición

Dada una máquina de Turing $M = (Q, \Sigma, q_0, \delta, F)$*:*

$$L(M) \quad = \quad \{w \in \Sigma^* \mid M \textit{ acepta } w\}$$

$L(M)$ es el lenguaje aceptado por M

Para recordar: Máquinas de Turing no determinista

Definición

Máquina de Turing no determinista: $(Q, \Sigma, q_0, \delta, F)$

- Q *es un conjunto finito de estados*
- Σ *es un alfabeto finito tal que* $\vdash, \mathtt{B} \notin \Sigma$
- $q_0 \in Q$ *es el estado inicial*
- $F \subseteq Q$ *es un conjunto de estados finales*
- δ *es una relación de transición:*

$$\delta \quad \subseteq \quad Q \times (\Sigma \cup \{\vdash, \mathtt{B}\}) \times Q \times (\Sigma \cup \{\vdash, \mathtt{B}\}) \times \{\leftarrow, \square, \rightarrow\}$$

Máquinas de Turing no determinista: Funcionamiento

Hacemos los mismos supuestos que para el caso determinista

- En particular sobre el uso del símbolo $\vdash$

Máquinas de Turing no determinista: Funcionamiento

Hacemos los mismos supuestos que para el caso determinista

- En particular sobre el uso del símbolo $\vdash$

La inicialización es igual que para el caso determinista

- Al comenzar a funcionar, la máquina se encuentra en el estado q_0 y su cabeza lectora está en la posición 1 de la cinta

Máquinas de Turing no determinista: Funcionamiento

En cada instante la máquina se encuentra en un estado q y su cabeza lectora está en una posición p que contiene un símbolo a.

- Sea $T = \{(q', b, X) \mid (q, a, q', b, X) \in \delta\}$. Si $T \neq \emptyset$, entonces la máquina elije $(q', b, X) \in T$ y:

Máquinas de Turing no determinista: Funcionamiento

En cada instante la máquina se encuentra en un estado q y su cabeza lectora está en una posición p que contiene un símbolo a.

- Sea $T = \{(q', b, X) \mid (q, a, q', b, X) \in \delta\}$. Si $T \neq \emptyset$, entonces la máquina elije $(q', b, X) \in T$ y:
  - escribe el símbolo b en la posición p de la cinta

Máquinas de Turing no determinista: Funcionamiento

En cada instante la máquina se encuentra en un estado q y su cabeza lectora está en una posición p que contiene un símbolo a.

- Sea $T = \{(q', b, X) \mid (q, a, q', b, X) \in \delta\}$. Si $T \neq \emptyset$, entonces la máquina elije $(q', b, X) \in T$ y:
  - escribe el símbolo b en la posición p de la cinta
  - cambia de estado desde q a q'

Máquinas de Turing no determinista: Funcionamiento

En cada instante la máquina se encuentra en un estado q y su cabeza lectora está en una posición p que contiene un símbolo a.

- Sea $T = \{(q', b, X) \mid (q, a, q', b, X) \in \delta\}$. Si $T \neq \emptyset$, entonces la máquina elije $(q', b, X) \in T$ y:
  - escribe el símbolo b en la posición p de la cinta
  - cambia de estado desde q a q'
  - mueve la cabeza lectora a la posición $p - 1$ si X es $\leftarrow$, y a la posición $p + 1$ si X es $\rightarrow$. Si X es $\square$, entonces la cabeza lectora permanece en la posición p

Máquinas de Turing no deterministas: Lenguaje aceptado

Definición

Dada una máquina de Turing M no determinista con alfabeto Σ:

$$
\begin{aligned}
L(M) \;=\; \{w \in \Sigma^* \mid & \textit{ existe alguna ejecución de } M \\
& \textit{con entrada } w \textit{ que termina en un estado final}\}
\end{aligned}
$$

Máquinas de Turing no deterministas: Lenguaje aceptado

Definición

Dada una máquina de Turing M no determinista con alfabeto Σ:

$$L(M) = \{w \in \Sigma^* \mid \text{existe alguna ejecución de } M \text{ con entrada } w \text{ que termina en un estado final}\}$$

Teorema

Para cada MT no determinista M, existe una MT determinista M' tal que $L(M) = L(M')$.

Complejidad de un algoritmo

Dado: MT determinista M con alfabeto Σ que para en todas las entradas

Complejidad de un algoritmo

Dado: MT determinista M con alfabeto Σ que para en todas las entradas

Definición

- *Paso de M:* *Ejecutar una instrucción de la función de transición*

Complejidad de un algoritmo

Dado: MT determinista M con alfabeto Σ que para en todas las entradas

Definición

- *Paso de M: Ejecutar una instrucción de la función de transición*
- *$tiempo_M(w)$: Número de pasos ejecutados por M con entrada w*

Complejidad de un algoritmo

Dado: MT determinista M con alfabeto Σ que para en todas las entradas

Definición

- *Paso de M: Ejecutar una instrucción de la función de transición*
- *$tiempo_M(w)$: Número de pasos ejecutados por M con entrada w*
- $t_M(n) = \text{máx}\{\ tiempo_M(w) \mid w \in \Sigma^* \text{ y } |w| = n\}$, *para cada* $n \geq 0$

Complejidad de un algoritmo

Dado: MT determinista M con alfabeto Σ que para en todas las entradas

Definición

- *Paso de M: Ejecutar una instrucción de la función de transición*
- *$tiempo_M(w)$: Número de pasos ejecutados por M con entrada w*
- $t_M(n) = \text{máx}\{\ tiempo_M(w) \mid w \in \Sigma^* \text{ y } |w| = n\}$, *para cada* $n \geq 0$

t_M: Tiempo de ejecución de M en el peor caso

Complejidad de un problema

Definición

Un lenguaje L puede ser aceptado en tiempo t si es que existe una MT determinista M tal que:

- *M para en todas las entradas*
- *$L = L(M)$*
- *t_M es $O(t)$, vale decir, existe $c \in \mathbb{R}^+$ y $n_0 \in \mathbb{N}$ tal que $t_M(n) \leq c \cdot t(n)$ para todo $n \geq n_0$*

Complejidad de un problema

Definición

Un lenguaje L puede ser aceptado en tiempo t si es que existe una MT determinista M tal que:

- *M para en todas las entradas*
- $L = L(M)$
- *t_M es $O(t)$, vale decir, existe $c \in \mathbb{R}^+$ y $n_0 \in \mathbb{N}$ tal que $t_M(n) \leq c \cdot t(n)$ para todo $n \geq n_0$*

El tiempo para computar una función f se define de la misma forma

- ¿Cómo se define una MT que calcula una función?

Clases de complejidad

Dado: Alfabeto Σ

Definición

DTIME(t): conjunto de todos los lenguajes $L \subseteq \Sigma^*$ *que pueden ser aceptados en tiempo* t

Clases de complejidad

Dado: Alfabeto Σ

Definición

DTIME(t): conjunto de todos los lenguajes $L \subseteq \Sigma^*$ que pueden ser aceptados en tiempo t

Dos clases fundamentales:

$$\begin{aligned} \text{PTIME} &= \bigcup_{k \in \mathbb{N}} \text{DTIME}(n^k) \\ \text{EXPTIME} &= \bigcup_{k \in \mathbb{N}} \text{DTIME}(2^{n^k}) \end{aligned}$$

PTIME: Conjunto de todos los problemas que pueden ser solucionados eficientemente

Clases de complejidad no deterministas

Dado: MT no determinista M con alfabeto Σ

Clases de complejidad no deterministas

Dado: MT no determinista M con alfabeto Σ

Definición

- *Paso de M:* *Ejecutar una instrucción de la relación de transición*

Clases de complejidad no deterministas

Dado: MT no determinista M con alfabeto Σ

Definición

- *Paso de M: Ejecutar una instrucción de la relación de transición*
- *$\mathit{tiempo}_M(w)$: Número de pasos de M con entrada w en la ejecución más corta que acepta a w*

Clases de complejidad no deterministas

Dado: MT no determinista M con alfabeto Σ

Definición

- *Paso de M: Ejecutar una instrucción de la relación de transición*
- *$tiempo_M(w)$: Número de pasos de M con entrada w en la ejecución más corta que acepta a w*
- $t_M(n) \;=\; \text{máx}(\{n\} \cup \{\; tiempo_M(w) \;\mid\; w \in \Sigma^*, |w| = n \text{ y } M \text{ acepta } w\})$, *para cada* $n \geq 0$

Clases de complejidad no deterministas

Dado: MT no determinista M con alfabeto Σ

Definición

- *Paso de M: Ejecutar una instrucción de la relación de transición*
- *$tiempo_M(w)$: Número de pasos de M con entrada w en la ejecución más corta que acepta a w*
- $t_M(n) \;=\; \text{máx}(\{n\} \cup \{\ tiempo_M(w) \ \mid\ w \in \Sigma^*, |w| = n \text{ y } M \text{ acepta } w\})$, *para cada* $n \geq 0$

¿Por que incluimos $\{n\}$ en la definición?

Clases de complejidad no deterministas

Definición

Un lenguaje L es aceptado en tiempo t por una MT M no determinista si:

- $L = L(M)$
- t_M es $O(t)$

Clases de Complejidad no deterministas

Dado: Alfabeto Σ

Definición

NTIME(t): Conjunto de todos los lenguajes que pueden ser aceptados en tiempo t por alguna MT no determinista

Clases de Complejidad no deterministas

Dado: Alfabeto Σ

Definición

$NTIME(t)$: Conjunto de todos los lenguajes que pueden ser aceptados en tiempo t por alguna MT no determinista

Dos clases fundamentales:

$$\begin{aligned} \mathrm{NP} &= \bigcup_{k\in\mathbb{N}} \mathrm{NTIME}(n^k) \\ \mathrm{NEXPTIME} &= \bigcup_{k\in\mathbb{N}} \mathrm{NTIME}(2^{n^k}) \end{aligned}$$

Relación entre clases de complejidad

¿Cuál es la relación entre las clases de complejidad que acabamos de definir?

Relación entre clases de complejidad

¿Cuál es la relación entre las clases de complejidad que acabamos de definir?

Fácil de demostrar : PTIME $\subseteq$ NP $\subseteq$ EXPTIME $\subseteq$ NEXPTIME

Relación entre clases de complejidad

¿Cuál es la relación entre las clases de complejidad que acabamos de definir?

Fácil de demostrar	:	PTIME $\subseteq$ NP $\subseteq$ EXPTIME $\subseteq$ NEXPTIME
No tan fácil de demostrar	:	PTIME $\subsetneq$ EXPTIME

Relación entre clases de complejidad

¿Cuál es la relación entre las clases de complejidad que acabamos de definir?

Fácil de demostrar	:	PTIME $\subseteq$ NP $\subseteq$ EXPTIME $\subseteq$ NEXPTIME
No tan fácil de demostrar	:	PTIME $\subsetneq$ EXPTIME
Aún sin resolver	:	¿PTIME $\subsetneq$ NP?

Relación entre clases de complejidad

¿Cuál es la relación entre las clases de complejidad que acabamos de definir?

Fácil de demostrar	:	PTIME $\subseteq$ NP $\subseteq$ EXPTIME $\subseteq$ NEXPTIME
No tan fácil de demostrar	:	PTIME $\subsetneq$ EXPTIME
Aún sin resolver	:	¿PTIME $\subsetneq$ NP?

También necesitamos clases definidas en término del espacio utilizado.

Espacio utilizado en una Máquina de Turing

Para introducir la noción de espacio utilizado en una MT tenemos que distinguir entre:

- el espacio ocupado por la entrada, y
- el espacio necesario para procesar la entrada

Para hacer esta distinción utilizamos MT con dos cintas:

- Cinta para la entrada: Sólo de lectura
- Cinta de trabajo: Como en una MT usual

El espacio utilizado se mide en términos de las celdas visitadas en la cinta de trabajo

Máquinas de Turing con cinta de trabajo

Definición

Máquina de Turing con cinta de trabajo (determinista):

$(Q, \Sigma, q_0, \delta, F)$

- Q *es un conjunto finito de estados*
- Σ *es un alfabeto finito tal que* $\vdash, \mathtt{B} \notin \Sigma$
- $q_0 \in Q$ *es el estado inicial*
- $F \subseteq Q$ *es un conjunto de estados finales*
- δ *es una función parcial:*

$$\begin{aligned}\delta \ : \ Q \times (\Sigma \cup \{\vdash, \mathtt{B}\}) \times (\Sigma \cup \{\vdash, \mathtt{B}\}) \ &\to \\ Q \times \{\leftarrow, \square, \to\} \times (\Sigma \cup \{\vdash, \mathtt{B}\}) \times \{\leftarrow, \square, \to\}\end{aligned}$$

Espacio utilizado en una MT

Dado: MT determinista M con alfabeto Σ que para en todas las entradas

Espacio utilizado en una MT

Dado: MT determinista M con alfabeto Σ que para en todas las entradas

Definición

- $espacio_M(w)$: *número de celdas visitadas en la cinta de trabajo de M al procesar w*

Espacio utilizado en una MT

Dado: MT determinista M con alfabeto Σ que para en todas las entradas

Definición

- $\textit{espacio}_M(w)$: *número de celdas visitadas en la cinta de trabajo de M al procesar w*
- $s_M(n) \ = \ \max\{\ \textit{espacio}_M(w) \ \mid \ w \in \Sigma^* \text{ y } |w| = n\}$, *para cada* $n \geq 0$

Espacio utilizado en una MT

Dado: MT determinista M con alfabeto Σ que para en todas las entradas

Definición

- $espacio_M(w)$: *número de celdas visitadas en la cinta de trabajo de* M *al procesar* w
- $s_M(n) \;=\; \text{máx}\{\; espacio_M(w) \;\mid\; w \in \Sigma^* \text{ y } |w| = n\}$, *para cada* $n \geq 0$

s_M: Espacio utilizado en la ejecución de M en el peor caso

Espacio utilizado para resolver un problema

Definición

Un lenguaje L puede ser aceptado en espacio s si es que existe una MT determinista M tal que:

- *M para en todas las entradas*
- $L = L(M)$
- s_M *es* $O(s)$

Espacio utilizado para resolver un problema

Definición

Un lenguaje L puede ser aceptado en espacio s si es que existe una MT determinista M tal que:

- *M para en todas las entradas*
- $L = L(M)$
- s_M *es* $O(s)$

El espacio requerido para computar una función f se define de la misma forma

Clases de complejidad: Espacio

Dado: Alfabeto Σ

Definición

DSPACE(s): Conjunto de todos los lenguajes $L \subseteq \Sigma^*$ que pueden ser aceptados en espacio s

Clases de complejidad: Espacio

Dado: Alfabeto Σ

Definición

DSPACE(s): Conjunto de todos los lenguajes $L \subseteq \Sigma^*$ que pueden ser aceptados en espacio s

Tres clases fundamentales:

$$\begin{aligned}
\text{LOGSPACE} &= \text{DSPACE}(\log n) \\
\text{PSPACE} &= \bigcup_{k \in \mathbb{N}} \text{DSPACE}(n^k) \\
\text{EXPSPACE} &= \bigcup_{k \in \mathbb{N}} \text{DSPACE}(2^{n^k})
\end{aligned}$$

Relación entre clases de complejidad

¿Cuál es la relación entre las clases de complejidad que acabamos de definir?

Relación entre clases de complejidad

¿Cuál es la relación entre las clases de complejidad que acabamos de definir?

La relación es más compleja:

$$\begin{aligned}\text{LOGSPACE} \subseteq \text{PTIME} \subseteq \text{NP} \subseteq \text{PSPACE} \subseteq \\ \text{EXPTIME} \subseteq \text{NEXPTIME} \subseteq \text{EXPSPACE}\end{aligned}$$

Relación entre clases de complejidad

¿Cuál es la relación entre las clases de complejidad que acabamos de definir?

La relación es más compleja:

$$\text{LOGSPACE} \subseteq \text{PTIME} \subseteq \text{NP} \subseteq \text{PSPACE} \subseteq \\ \text{EXPTIME} \subseteq \text{NEXPTIME} \subseteq \text{EXPSPACE}$$

También se sabe que: $\text{LOGSPACE} \subsetneq \text{PSPACE} \subsetneq \text{EXPSPACE}$

Relación entre clases de complejidad

¿Cuál es la relación entre las clases de complejidad que acabamos de definir?

La relación es más compleja:

$$\text{LOGSPACE} \subseteq \text{PTIME} \subseteq \text{NP} \subseteq \text{PSPACE} \subseteq \text{EXPTIME} \subseteq \text{NEXPTIME} \subseteq \text{EXPSPACE}$$

También se sabe que: $\text{LOGSPACE} \subsetneq \text{PSPACE} \subsetneq \text{EXPSPACE}$

Y todavía nos faltan las clases no deterministas ...

Clases de complejidad no deterministas: Espacio

Dado: MT no determinista M con cinta de trabajo y alfabeto Σ

Clases de complejidad no deterministas: Espacio

Dado: MT no determinista M con cinta de trabajo y alfabeto Σ

Definición

- *$\text{espacio}_M(w)$: Número mínimo de celdas visitadas en la cinta de trabajo de M, entre todas las ejecuciones de M que aceptan w*

Clases de complejidad no deterministas: Espacio

Dado: MT no determinista M con cinta de trabajo y alfabeto Σ

Definición

- *$espacio_M(w)$: Número mínimo de celdas visitadas en la cinta de trabajo de M, entre todas las ejecuciones de M que aceptan w*
- *$s_M(n) = \max(\{1\} \cup \{ espacio_M(w) \mid w \in \Sigma^*, |w| = n \text{ y } M \text{ acepta } w\})$, para cada $n \geq 0$*

Clases de complejidad no deterministas: Espacio

Dado: MT no determinista M con cinta de trabajo y alfabeto Σ

Definición

- *$\mathit{espacio}_M(w)$: Número mínimo de celdas visitadas en la cinta de trabajo de M, entre todas las ejecuciones de M que aceptan w*
- $s_M(n) \; = \; \text{máx}(\{1\} \cup \{ \; \mathit{espacio}_M(w) \; \mid \; w \in \Sigma^*, |w| = n \text{ y } M \text{ acepta } w\})$, *para cada* $n \geq 0$

¿Por que incluimos $\{1\}$ en la definición?

Clases de complejidad no deterministas: Espacio

Definición

Un lenguaje L es aceptado en espacio s por una MT M no determinista si:

- $L = L(M)$
- s_M *es* $O(s)$

Clases de Complejidad no deterministas: Espacio

Dado: Alfabeto Σ

Definición

NSPACE(s): Conjunto de todos los lenguajes que pueden ser aceptados en espacio s por alguna MT no determinista

Clases de Complejidad no deterministas: Espacio

Dado: Alfabeto Σ

Definición

$NSPACE(s)$: Conjunto de todos los lenguajes que pueden ser aceptados en espacio s por alguna MT no determinista

Tres clases fundamentales:

$$\begin{aligned}
\text{NLOGSPACE} &= \text{NSPACE}(\log n) \\
\text{NPSPACE} &= \bigcup_{k \in \mathbb{N}} \text{NSPACE}(n^k) \\
\text{NEXPSPACE} &= \bigcup_{k \in \mathbb{N}} \text{NSPACE}(2^{n^k})
\end{aligned}$$

Relación entre clases de complejidad

¿Cuál es la relación entre las clases de complejidad que definimos?

Se sabe que: PSPACE = NPSPACE y EXPSPACE = NEXPSPACE

Relación entre clases de complejidad

¿Cuál es la relación entre las clases de complejidad que definimos?

Se sabe que: PSPACE $=$ NPSPACE y EXPSPACE $=$ NEXPSPACE

Relación final:

$$\text{LOGSPACE} \subseteq \text{NLOGSPACE} \subseteq \text{PTIME} \subseteq \text{NP} \subseteq \text{PSPACE} \subseteq \text{EXPTIME} \subseteq \text{NEXPTIME} \subseteq \text{EXPSPACE}$$

La noción de completitud

Dado: Clase de complejidad $\mathcal{C}$ que contiene a LOGSPACE

Definición

- *Decimos que L es hard para $\mathcal{C}$ si para todo $L' \in \mathcal{C}$ existe una reducción de L' a L que puede ser calculada en espacio logarítmico*
- *Decimos que L es completo para $\mathcal{C}$ (o $\mathcal{C}$-completo) si $L \in \mathcal{C}$ y L es hard para $\mathcal{C}$*

Problemas completos

Problema completo para NP: SAT

Problemas completos

Problema completo para NP: SAT

- ¿Puede nombrar otros?

Problemas completos

Problema completo para NP: SAT

- ¿Puede nombrar otros?

Problema completo para NLOGSPACE: Graph Reachability

Problemas completos

Problema completo para NP: SAT

- ¿Puede nombrar otros?

Problema completo para NLOGSPACE: Graph Reachability

Problema completo para PSPACE: QBF

Complejidad de la LPO

Dado: Vocabulario $\mathcal{L}$

Definición

La complejidad de LPO se define como la complejidad del siguiente lenguaje:

$$L = \{(\mathfrak{A}, \varphi) \mid \mathfrak{A} \in \textsc{Struct}[\mathcal{L}],\ \varphi \textit{ es una } \mathcal{L}\textit{-oración y } \mathfrak{A} \models \varphi\}$$

Complejidad de la LPO

Dado: Vocabulario $\mathcal{L}$

Definición

La complejidad de LPO se define como la complejidad del siguiente lenguaje:

$$L = \{(\mathfrak{A}, \varphi) \mid \mathfrak{A} \in \textsc{Struct}[\mathcal{L}],\ \varphi \textit{ es una } \mathcal{L}\textit{-oración y } \mathfrak{A} \models \varphi\}$$

Notación

Llamamos a esto complejidad combinada

Complejidad combinada de la LPO

¿Cuál es la complejidad combinada de la lógica de primer orden?

Complejidad combinada de la LPO

¿Cuál es la complejidad combinada de la lógica de primer orden?

Teorema

L es PSPACE-completo

Complejidad combinada de la LPO

¿Cuál es la complejidad combinada de la lógica de primer orden?

Teorema

L es PSPACE-completo

Ejercicio

Demuestre que la pertenencia es válida para cualquier vocabulario

Complejidad combinada de la LPO

El resultado anterior nos dice que es muy difícil evaluar una consulta en LPO

Complejidad combinada de la LPO

El resultado anterior nos dice que es muy difícil evaluar una consulta en LPO

- ¿Cómo puede entonces funcionar un sistema de bases de datos?

Complejidad combinada de la LPO

El resultado anterior nos dice que es muy difícil evaluar una consulta en LPO

- ¿Cómo puede entonces funcionar un sistema de bases de datos?
- En general: Consultas son pequeñas comparadas con la base de datos

Complejidad de los datos de la LPO

Vamos a tomar en cuenta la diferencia entre el tamaño de las consultas y las bases de datos.

Complejidad de los datos de la LPO

Vamos a tomar en cuenta la diferencia entre el tamaño de las consultas y las bases de datos.

Dado: Vocabulario $\mathcal{L}$

Definición

La complejidad de evaluar una fórmula fija φ en LPO se define como la complejidad del siguiente lenguaje:

$$L_\varphi = \{\mathfrak{A} \mid \mathfrak{A} \in \textsc{Struct}[\mathcal{L}] \text{ y } \mathfrak{A} \models \varphi\}$$

Complejidad de los datos de la LPO

Vamos a tomar en cuenta la diferencia entre el tamaño de las consultas y las bases de datos.

Dado: Vocabulario $\mathcal{L}$

Definición

La complejidad de evaluar una fórmula fija φ en LPO se define como la complejidad del siguiente lenguaje:

$$L_\varphi \quad = \quad \{\mathfrak{A} \mid \mathfrak{A} \in \textsc{Struct}[\mathcal{L}] \text{ y } \mathfrak{A} \models \varphi\}$$

Notación

Llamamos a esto complejidad de los datos

Complejidad de los datos de la LPO

Teorema

Para cada $\mathcal{L}$-oración φ se tiene que L_φ está en LOGSPACE

Complejidad de los datos de la LPO

Teorema

Para cada $\mathcal{L}$-oración φ se tiene que L_φ está en LOGSPACE

Ejercicio

Demuestre el teorema

Complejidad de los datos de la LPO

Teorema

Para cada $\mathcal{L}$-oración φ se tiene que L_φ está en LOGSPACE

Ejercicio

Demuestre el teorema

Conclusión

Cada consulta puede ser evaluada eficientemente

La gran pregunta (y otra más)

- ¿Existe algún lenguaje que sea completo para PTIME en términos de complejidad de los datos? ¿Puede ser este lenguaje LPO?

La gran pregunta (y otra más)

- ¿Existe algún lenguaje que sea completo para PTIME en términos de complejidad de los datos? ¿Puede ser este lenguaje LPO?
- ¿Hay consultas en algún lenguaje que no pueden ser evaluadas eficientemente?

La gran pregunta (y otra más)

- ¿Existe algún lenguaje que sea completo para PTIME en términos de complejidad de los datos? ¿Puede ser este lenguaje LPO?
- ¿Hay consultas en algún lenguaje que no pueden ser evaluadas eficientemente?
- Partimos por la segunda, pero antes...

Intermezzo: falla de completitud en el caso finito

No se puede tener un sistema deductivo completo en el caso finito.

Intermezzo: falla de completitud en el caso finito

No se puede tener un sistema deductivo completo en el caso finito.

Esto es una consecuencia de la indecidibilidad del siguiente problema:

$$\text{FIN-SAT} \quad = \quad \{\varphi \mid \varphi \text{ tiene un modelo finito}\}$$

Intermezzo: falla de completitud en el caso finito

No se puede tener un sistema deductivo completo en el caso finito.

Esto es una consecuencia de la indecidibilidad del siguiente problema:

$$\text{FIN-SAT} \quad = \quad \{\varphi \mid \varphi \text{ tiene un modelo finito}\}$$

Teorema (Trakhtenbrot)

FIN-SAT es indecidible

Demostración: Reducimos desde el problema de verificar si una MT determinista acepta la palabra vacía.

Teorema de Trakhtenbrot: Demostración

Sea $\Sigma = \{0,1\}$ y $M = (Q, \Sigma, q_0, \delta, F)$ una MT tal que:

- $F = \{q_m\}$
- $\delta : (Q \smallsetminus \{q_m\}) \times (\Sigma \cup \{\mathtt{B}, \vdash\}) \to Q \times (\Sigma \cup \{\mathtt{B}, \vdash\}) \times \{\leftarrow, \Box, \rightarrow\}$ es una función total

Teorema de Trakhtenbrot: Demostración

Sea $\Sigma = \{0, 1\}$ y $M = (Q, \Sigma, q_0, \delta, F)$ una MT tal que:

- $F = \{q_m\}$
- $\delta : (Q \smallsetminus \{q_m\}) \times (\Sigma \cup \{\texttt{B}, \vdash\}) \to Q \times (\Sigma \cup \{\texttt{B}, \vdash\}) \times \{\leftarrow, \square, \rightarrow\}$ es una función total

Sea $\mathcal{L}$ es siguiente vocabulario:

$$\{L(\cdot,\cdot), S(\cdot,\cdot), P(\cdot), H(\cdot,\cdot), \\ T_0(\cdot,\cdot), T_1(\cdot,\cdot), T_{\texttt{B}}(\cdot,\cdot), T_{\vdash}(\cdot,\cdot)\} \cup \{E_q(\cdot) \mid q \in Q\}$$

Teorema de Trakhtenbrot: Demostración

El funcionamiento de la MT M sobre la palabra vacía es definido por una oración φ sobre $\mathcal{L}$.

- M acepta la palabra vacía si y sólo si $\varphi \in$ FIN-SAT

Teorema de Trakhtenbrot: Demostración

El funcionamiento de la MT M sobre la palabra vacía es definido por una oración φ sobre $\mathcal{L}$.

- M acepta la palabra vacía si y sólo si $\varphi \in$ FIN-SAT

φ es definida como $\varphi_L \wedge \varphi_S \wedge \varphi_P \wedge \varphi_I \wedge \varphi_C \wedge \varphi_A \wedge \varphi_\delta$:

Teorema de Trakhtenbrot: Demostración

El funcionamiento de la MT M sobre la palabra vacía es definido por una oración φ sobre $\mathcal{L}$.

- M acepta la palabra vacía si y sólo si $\varphi \in$ FIN-SAT

φ es definida como $\varphi_L \wedge \varphi_S \wedge \varphi_P \wedge \varphi_I \wedge \varphi_C \wedge \varphi_A \wedge \varphi_\delta$:

φ_L: L es un orden lineal

$$\forall x\,(\neg L(x,x)) \wedge \forall x \forall y\,(x = y \vee L(x,y) \vee L(y,x)) \wedge \\ \forall x \forall y \forall z\,(L(x,y) \wedge L(y,z) \rightarrow L(x,z))$$

Teorema de Trakhtenbrot: Demostración

φ_S: S es la relación de sucesor asociada a L

$$\forall x \forall y \, (S(x,y) \leftrightarrow (L(x,y) \wedge \neg \exists z \, (L(x,z) \wedge L(z,y))))$$

Teorema de Trakhtenbrot: Demostración

φ_S: S es la relación de sucesor asociada a L

$$\forall x \forall y\, (S(x,y) \leftrightarrow (L(x,y) \wedge \neg \exists z\, (L(x,z) \wedge L(z,y))))$$

φ_P: P almacena el primer elemento del orden

$$\forall x\, (P(x) \leftrightarrow \neg \exists y\, L(y,x))$$

Teorema de Trakhtenbrot: Demostración

φ_S: S es la relación de sucesor asociada a L

$$\forall x \forall y\,(S(x,y) \leftrightarrow (L(x,y) \wedge \neg \exists z\,(L(x,z) \wedge L(z,y))))$$

φ_P: P almacena el primer elemento del orden

$$\forall x\,(P(x) \leftrightarrow \neg \exists y\, L(y,x))$$

φ_I: Estado inicial

$$\forall x \forall y\,((P(x) \wedge S(x,y)) \;\rightarrow$$
$$(E_{q_0}(x) \wedge H(x,y) \wedge T_{\vdash}(x,x) \wedge \forall z\,(L(x,z) \rightarrow T_{\mathtt{B}}(x,z))))$$

Teorema de Trakhtenbrot: Demostración

φ_C: La máquina funciona correctamente.

- Se define como la conjunción de cuatro fórmulas

Teorema de Trakhtenbrot: Demostración

φ_C: La máquina funciona correctamente.

- Se define como la conjunción de cuatro fórmulas

Cada celda siempre contiene un único símbolo:

$$\begin{aligned}\forall x \forall y\, ((T_0(x,y) \vee T_1(x,y) \vee T_{\mathrm{B}}(x,y) \vee T_{\vdash}(x,y)) \wedge \\ (\neg T_0(x,y) \vee \neg T_1(x,y)) \wedge (\neg T_0(x,y) \vee \neg T_{\mathrm{B}}(x,y)) \wedge \\ (\neg T_0(x,y) \vee \neg T_{\vdash}(x,y)) \wedge (\neg T_1(x,y) \vee \neg T_{\mathrm{B}}(x,y)) \wedge \\ (\neg T_1(x,y) \vee \neg T_{\vdash}(x,y)) \wedge (\neg T_{\mathrm{B}}(x,y) \vee \neg T_{\vdash}(x,y)))\end{aligned}$$

Teorema de Trakhtenbrot: Demostración

La máquina siempre está en un único estado:

$$\forall x \left(\bigvee_{q \in Q} \left(E_q(x) \wedge \bigwedge_{q' \in (Q \setminus \{q\})} \neg E_{q'}(x) \right) \right)$$

Teorema de Trakhtenbrot: Demostración

La máquina siempre está en un único estado:

$$\forall x \left(\bigvee_{q \in Q} \left(E_q(x) \wedge \bigwedge_{q' \in (Q \setminus \{q\})} \neg E_{q'}(x) \right) \right)$$

La cabeza siempre está en una única posición:

$$\forall x \exists y \left(H(x, y) \wedge \forall z \left(H(x, z) \rightarrow y = z \right) \right)$$

Teorema de Trakhtenbrot: Demostración

Finalmente, el valor de una celda no cambia si no es apuntada por la cabeza lectora:

$$\forall x \forall y \forall z \left(\Big(S(x,y) \wedge \neg H(x,z) \Big) \;\to\; \Big((T_0(x,z) \wedge T_0(y,z)) \vee (T_1(x,z) \wedge T_1(y,z)) \vee (T_{\mathrm{B}}(x,z) \wedge T_{\mathrm{B}}(y,z)) \vee (T_{\vdash}(x,z) \wedge T_{\vdash}(y,z)) \Big) \right)$$

Teorema de Trakhtenbrot: Demostración

φ_A: La máquina acepta la palabra vacía

$$\exists x \, E_{q_m}(x)$$

Teorema de Trakhtenbrot: Demostración

φ_A: La máquina acepta la palabra vacía

$$\exists x\, E_{q_m}(x)$$

φ_δ: Representa la función de transición δ

Vamos a hacer uno de los casos: $\delta(q,a) = (q', b, \leftarrow)$

$$\forall x \forall y \forall u \forall v \left(\left(H(x,y) \wedge T_a(x,y) \wedge E_q(x) \wedge \right.\right.$$
$$\left.\left. S(x,u) \wedge S(v,y) \right) \rightarrow \left(H(u,v) \wedge T_b(u,y) \wedge E_{q'}(u) \right) \right)$$

$\square$

Teorema de Trakhtenbrot: Aplicaciones

Vamos a ver algunas aplicaciones del Teorema de Trakhtenbrot

Notación

FIN-VAL $= \{\varphi \mid \varphi$ *es satisfecha por todos los modelos finitos* $\}$

Teorema de Trakhtenbrot: Aplicaciones

Vamos a ver algunas aplicaciones del Teorema de Trakhtenbrot

Notación

FIN-VAL $= \{\varphi \mid \varphi$ *es satisfecha por todos los modelos finitos* $\}$

Corolario

FIN-VAL es indecidible

Teorema de Trakhtenbrot: Aplicaciones

Vamos a ver algunas aplicaciones del Teorema de Trakhtenbrot

Notación

$\mathit{FIN\text{-}VAL} = \{\varphi \mid \varphi \textit{ es satisfecha por todos los modelos finitos } \}$

Corolario

FIN-VAL es indecidible

Ejercicio

Demuestre el corolario

Teorema de Trakhtenbrot: Aplicaciones

Corolario

No existe un sistema de deducción decidible para FIN-VAL que sea correcto y completo

Teorema de Trakhtenbrot: Aplicaciones

Corolario

No existe un sistema de deducción decidible para FIN-VAL que sea correcto y completo

Demostración: Si existiera este sistema, entonces:

$$\overline{\text{FIN-SAT}} \quad = \quad \{\varphi \mid \varphi \textbf{ no } \text{tiene un modelo finito}\}$$

sería recursivamente enumerable (¿Por qué?)

Teorema de Trakhtenbrot: Aplicaciones

Corolario

No existe un sistema de deducción decidible para FIN-VAL que sea correcto y completo

Demostración: Si existiera este sistema, entonces:

$$\overline{\text{FIN-SAT}} \quad = \quad \{\varphi \mid \varphi \textbf{ no} \text{ tiene un modelo finito}\}$$

sería recursivamente enumerable (¿Por qué?)

Pero FIN-SAT es recursivamente enumerable

- Por lo tanto: FIN-SAT sería decidible, lo cual contradice el Teorema de Trakhtenbrot

$\square$