Teoría de la Computación

Clase 30: Reducibilidad

Mauro Artigiani

12 Noviembre 2021

Universidad del Rosario, Bogotá

Reducibilidad

Reducibilidad

Para mostrar que varios lenguajes son indecidibiles, utilizaremos la reducibilidad.

Reducibilidad

Para mostrar que varios lenguajes son indecidibiles, utilizaremos la reducibilidad.

Supongamos de querer resolver un problema A. *Reducir* el problema A a otro problema B significa que si solucionamos B podemos solucionar A.

Reducibilidad

Para mostrar que varios lenguajes son indecidibiles, utilizaremos la reducibilidad.

Supongamos de querer resolver un problema A. *Reducir* el problema A a otro problema B significa que si solucionamos B podemos solucionar A.

Por ejemplo, si tenemos hambre podemos solucionar ese problema si solucionamos el problema de obtener comida, de una u otra manera.

Reducibilidad

Para mostrar que varios lenguajes son indecidibiles, utilizaremos la reducibilidad.

Supongamos de querer resolver un problema A. *Reducir* el problema A a otro problema B significa que si solucionamos B podemos solucionar A.

Por ejemplo, si tenemos hambre podemos solucionar ese problema si solucionamos el problema de obtener comida, de una u otra manera.

Otro ejemplo, más matemático, es solucionar un sistema de ecuaciones lineales. Este problema se reduce al problema de invertir la matriz de los coeficientes de cada ecuación.

Reducibilidad

Si A se reduce al problema B sabemos que encontrar una solución al problema A no es más difícil que resolver el problema B.

Reducibilidad

Si A se reduce al problema B sabemos que encontrar una solución al problema A no es más difícil que resolver el problema B.

Nosotros utilizaremos la reducibilidad para obtener más ejemplos de lenguajes indecidibiles:

Lema

Si A puede reducirse a B y B es decidible, entonces A es decidible.

Ejemplos de lenguajes indecidibiles

$\text{Halt}_{\texttt{TM}}$

En la clase pasada hemos visto que

$$A_{\texttt{TM}} = \{\langle M, w\rangle, M \text{ es una } \texttt{TM} \text{ y } M \text{ acepta la cadena } w\}$$

es indecidibile.

$\mathrm{Halt}_{\mathtt{TM}}$

En la clase pasada hemos visto que

$$A_{\mathtt{TM}} = \{\langle M, w\rangle, M \text{ es una } \mathtt{TM} \text{ y } M \text{ acepta la cadena } w\}$$

es indecidibile.

Queremos demostrar que el problema de la parada también es indecidibile.

$\text{Halt}_{\texttt{TM}}$

En la clase pasada hemos visto que

$$A_{\texttt{TM}} = \{\langle M, w\rangle, M \text{ es una } \texttt{TM} \text{ y } M \text{ acepta la cadena } w\}$$

es indecidibile.

Queremos demostrar que el problema de la parada también es indecidibile.

Teorema

Sea

$$\text{Halt}_{\texttt{TM}} = \{\langle M, w\rangle, M \text{ es una } \texttt{TM} \text{ y } M \text{ se detiene con entrada } w\}.$$

Luego $\text{Halt}_{\texttt{TM}}$ es indecidibile.

$\mathrm{Halt}_{\mathtt{TM}}$

La idea de la demostración es reducir $A_{\mathtt{TM}}$ a $\mathrm{Halt}_{\mathtt{TM}}$.

$\text{Halt}_{\texttt{TM}}$

La idea de la demostración es reducir $A_{\texttt{TM}}$ a $\text{Halt}_{\texttt{TM}}$.

Lo que hace el lenguaje $A_{\texttt{TM}}$ indecidibile es que una máquina de Turing M podría entrar en un *loop* infinito al calcular w.

$\text{Halt}_{\texttt{TM}}$

La idea de la demostración es reducir $A_{\texttt{TM}}$ a $\text{Halt}_{\texttt{TM}}$.

Lo que hace el lenguaje $A_{\texttt{TM}}$ indecidibile es que una máquina de Turing M podría entrar en un *loop* infinito al calcular w.

Si tenemos una máquina de Turing R que decide $\text{Halt}_{\texttt{TM}}$ podemos aprovechar de ella para excluir la posibilidad de un *loop*.

$\text{Halt}_{\texttt{TM}}$

La idea de la demostración es reducir $A_{\texttt{TM}}$ a $\text{Halt}_{\texttt{TM}}$.

Lo que hace el lenguaje $A_{\texttt{TM}}$ indecidibile es que una máquina de Turing M podría entrar en un *loop* infinito al calcular w.

Si tenemos una máquina de Turing R que decide $\text{Halt}_{\texttt{TM}}$ podemos aprovechar de ella para excluir la posibilidad de un *loop*. Dicho de otra manera, primero pasamos $\langle M, w\rangle$ a R. Si R acepta, significa que M se detiene calculando w y por eso podemos ver si M acepta o rechaza w.

$\text{Halt}_{\texttt{TM}}$

La idea de la demostración es reducir $A_{\texttt{TM}}$ a $\text{Halt}_{\texttt{TM}}$.

Lo que hace el lenguaje $A_{\texttt{TM}}$ indecidibile es que una máquina de Turing M podría entrar en un *loop* infinito al calcular w.

Si tenemos una máquina de Turing R que decide $\text{Halt}_{\texttt{TM}}$ podemos aprovechar de ella para excluir la posibilidad de un *loop*. Dicho de otra manera, primero pasamos $\langle M, w\rangle$ a R. Si R acepta, significa que M se detiene calculando w y por eso podemos ver si M acepta o rechaza w. Si R rechaza, significa que M no se detiene calculando w y entonces $w \notin L(M)$, lo que nos permitiría decidir $A_{\texttt{TM}}$.

$\text{Halt}_{\texttt{TM}}$

En resumen, si existe una $\texttt{TM}$ R que decide $\text{Halt}_{\texttt{TM}}$, podemos construir una $\texttt{TM}$ S que decida $A_{\texttt{TM}}$, la cual realiza este algoritmo:

$\text{Halt}_{\texttt{TM}}$

En resumen, si existe una `TM` R que decide $\text{Halt}_{\texttt{TM}}$, podemos construir una `TM` S que decida $A_{\texttt{TM}}$, la cual realiza este algoritmo:

```
procedure S(⟨M, w⟩)
    Correr R(⟨M, w⟩)
    if se obtiene Aceptar then
        Correr M(w)
        if se obtiene Aceptar then
            Aceptar
        else
            Rechazar
    else
        Rechazar
```

$E_{\texttt{TM}}$

Teorema

Sea

$$E_{\texttt{TM}} = \{\langle M \rangle, M \text{ es una } \texttt{TM} \text{ y } L(M) = \emptyset\}.$$

Luego $E_{\texttt{TM}}$ es indecidibile.

Teorema

Sea

$$E_{\texttt{TM}} = \{\langle M \rangle, M \text{ es una } \texttt{TM} \text{ y } L(M) = \emptyset\}.$$

Luego $E_{\texttt{TM}}$ es indecidibile.

Queremos reducir $A_{\texttt{TM}}$ a $E_{\texttt{TM}}$.

$E_{\mathtt{TM}}$

Teorema

Sea

$$E_{\mathtt{TM}} = \{\langle M \rangle, M \text{ es una } \mathtt{TM} \text{ y } L(M) = \emptyset\}.$$

Luego $E_{\mathtt{TM}}$ es indecidibile.

Queremos reducir $A_{\mathtt{TM}}$ a $E_{\mathtt{TM}}$. Supongamos que exista una máquina de Turing R que decida $E_{\mathtt{TM}}$. Cómo podemos decidir si una máquina de Turing M acepta la palabra w?

Teorema

Sea

$$E_{\mathtt{TM}} = \{\langle M \rangle, M \text{ es una } \mathtt{TM} \text{ y } L(M) = \emptyset\}.$$

Luego $E_{\mathtt{TM}}$ es indecidibile.

Queremos reducir $A_{\mathtt{TM}}$ a $E_{\mathtt{TM}}$. Supongamos que exista una máquina de Turing R que decida $E_{\mathtt{TM}}$. Cómo podemos decidir si una máquina de Turing M acepta la palabra w?

La idea es primero construir *otra* TM, M_1, que acepte w si y solo si M la acepta y rechace cualquiera otra palabra. En este caso, $L(M_1) = \emptyset$ si y solo si M rechaza w, entonces si podemos decir el primer hecho podemos decidir el segundo.

Construcción de la TM M_1

Dada la máquina de Turing M, construimos M_1:

procedure $M_1(x)$
    if $x \neq w$ **then**
        Rechazar
    else
        Correr $M(w)$
        if se obtiene Aceptar **then**
            Aceptar
        else
            Rechazar

Construcción de la `TM` M_1

Dada la máquina de Turing M, construimos M_1:

```
procedure M₁(x)
    if x ≠ w then
        Rechazar
    else
        Correr M(w)
        if se obtiene Aceptar then
            Aceptar
        else
            Rechazar
```

Nótese que $L(M_1) = w$ si y solo si M acepta w y es vacío en los otros casos.

Construcción de la TM M_1

Dada la máquina de Turing M, construimos M_1:

procedure $M_1(x)$
    if $x \neq w$ **then**
        Rechazar
    else
        Correr $M(w)$
        if se obtiene Aceptar **then**
            Aceptar
        else
            Rechazar

Nótese que $L(M_1) = w$ si y solo si M acepta w y es vacío en los otros casos.

Cuidado: M_1 está construida solamente para después correr $R(\langle M_1 \rangle)$, nunca vamos a correr de verdad M_1.

Reducción de $A_{\texttt{TM}}$ a $E_{\texttt{TM}}$

Supongamos que R sea una `TM` que decida $E_{\texttt{TM}}$. Construimos la siguiente máquina:

procedure $S(\langle M, w \rangle)$
    Construir la máquina M_1
    Correr $R(M_1)$
    if se obtiene Aceptar **then**
        Rechazar
    else
        Aceptar

Reducción de $A_{\mathtt{TM}}$ a $E_{\mathtt{TM}}$

Supongamos que R sea una `TM` que decida $E_{\mathtt{TM}}$. Construimos la siguiente máquina:

procedure $S(\langle M, w \rangle)$
    Construir la máquina M_1
    Correr $R(M_1)$
    if se obtiene Aceptar **then**
        Rechazar
    else
        Aceptar

S acepta $\langle M, w \rangle$ sii $L(M_1) \neq \emptyset$. Además, $L(M_1) \neq \emptyset$ sii M acepta w.

Reducción de $A_{\mathtt{TM}}$ a $E_{\mathtt{TM}}$

Supongamos que R sea una `TM` que decida $E_{\mathtt{TM}}$. Construimos la siguiente máquina:

procedure $S(\langle M, w \rangle)$
    Construir la máquina M_1
    Correr $R(M_1)$
    if se obtiene Aceptar **then**
        Rechazar
    else
        Aceptar

S acepta $\langle M, w \rangle$ sii $L(M_1) \neq \emptyset$. Además, $L(M_1) \neq \emptyset$ sii M acepta w. Por lo tanto, S decide $A_{\mathtt{TM}}$.

$\text{Regular}_{\texttt{TM}}$

Veamos un último ejemplo.

$\text{Regular}_{\texttt{TM}}$

Veamos un último ejemplo.

Teorema

Sea

$$\text{Regular}_{\texttt{TM}} = \{\langle M\rangle, M \text{ es una } \texttt{TM} \text{ y } \mathrm{L(M)} \text{ es regular}\}.$$

Luego $\text{Regular}_{\texttt{TM}}$ es indecidibile.

$\text{Regular}_{\texttt{TM}}$

Veamos un último ejemplo.

Teorema

Sea

$$\text{Regular}_{\texttt{TM}} = \{\langle M \rangle, M \text{ es una } \texttt{TM} \text{ y } \mathrm{L(M)} \text{ es regular}\}.$$

Luego $\text{Regular}_{\texttt{TM}}$ es indecidibile.

La idea es otra vez reducir $A_{\texttt{TM}}$ a $\text{Regular}_{\texttt{TM}}$.

$\text{Regular}_{\texttt{TM}}$

Veamos un último ejemplo.

Teorema

Sea

$$\text{Regular}_{\texttt{TM}} = \{\langle M \rangle, M \text{ es una } \texttt{TM} \text{ y L(M) es regular}\}.$$

Luego $\text{Regular}_{\texttt{TM}}$ es indecidibile.

La idea es otra vez reducir $A_{\texttt{TM}}$ a $\text{Regular}_{\texttt{TM}}$.

Asumimos que exista una TM, R, que decida $\text{Regular}_{\texttt{TM}}$. A partir de $\langle M, w \rangle$ podemos construir *otra* TM, M_2, cuyo lenguaje sea regular si y solo si M acepte w.

$\mathrm{Regular}_{\mathtt{TM}}$

Veamos un último ejemplo.

Teorema

Sea

$$\mathrm{Regular}_{\mathtt{TM}} = \{\langle M\rangle, M \text{ es una } \mathtt{TM} \text{ y } \mathrm{L(M)} \text{ es regular}\}.$$

Luego $\mathrm{Regular}_{\mathtt{TM}}$ es indecidibile.

La idea es otra vez reducir $A_{\mathtt{TM}}$ a $\mathrm{Regular}_{\mathtt{TM}}$.

Asumimos que exista una $\mathtt{TM}$, R, que decida $\mathrm{Regular}_{\mathtt{TM}}$. A partir de $\langle M, w\rangle$ podemos construir *otra* $\mathtt{TM}$, M_2, cuyo lenguaje sea regular si y solo si M acepte w. Entonces $R(\langle M_2\rangle)$ acepta si y solo si M acepta w. Además R decide $\mathrm{Regular}_{\mathtt{TM}}$, y por eso nos permite decidir $A_{\mathtt{TM}}$.

Construcción de M_2

Elegimos un lenguaje no regular: $\{0^n 1^n, n \in \mathbb{N}\}$ y uno regular: Σ^*.

Construcción de M_2

Elegimos un lenguaje no regular: $\{0^n 1^n, n \in \mathbb{N}\}$ y uno regular: Σ^*.

Con esto podemos construir M_2:

```
procedure M_2(x)
    if x = 0^n 1^n para algún n then
        Aceptar
    else
        Correr M(w)
        if se obtiene Aceptar then
            Aceptar
        else
            Rechazar
```

Construcción de M_2

Elegimos un lenguaje no regular: $\{0^n 1^n, n \in \mathbb{N}\}$ y uno regular: Σ^*.

Con esto podemos construir M_2:

procedure $M_2(x)$
    if $x = 0^n 1^n$ para algún n **then**
        Aceptar
    else
        Correr $M(w)$
        if se obtiene Aceptar **then**
            Aceptar
        else
            Rechazar

Nótese que $L(M_2) = \Sigma^*$ si M acepta w y $\{0^n 1^n, n \in \mathbb{N}\}$ si no.

Reducción de $A_{\texttt{TM}}$ a $\text{Regular}_{\texttt{TM}}$

Supongamos que R es una `TM` que decide $\text{Regular}_{\texttt{TM}}$. Construimos la siguiente máquina:

```
procedure S(⟨M, w⟩)
    Construir la máquina M_2
    Correr R(M_2)
    if se obtiene Aceptar then
        Aceptar
    else
        Rechazar
```

Reducción de $A_{\texttt{TM}}$ a Regular$_{\texttt{TM}}$

Supongamos que R es una TM que decide Regular$_{\texttt{TM}}$. Construimos la siguiente máquina:

```
procedure S(⟨M, w⟩)
    Construir la máquina M₂
    Correr R(M₂)
    if se obtiene Aceptar then
        Aceptar
    else
        Rechazar
```

S acepta $\langle M, w\rangle$ sii $L(M_2)$ es regular. Además, $L(M_2)$ es regular sii M acepta w.

Reducción de $A_{\texttt{TM}}$ a Regular$_{\texttt{TM}}$

Supongamos que R es una TM que decide Regular$_{\texttt{TM}}$. Construimos la siguiente máquina:

procedure $S(\langle M, w\rangle)$
    Construir la máquina M_2
    Correr $R(M_2)$
    if se obtiene Aceptar **then**
        Aceptar
    else
        Rechazar

S acepta $\langle M, w\rangle$ sii $L(M_2)$ es regular. Además, $L(M_2)$ es regular sii M acepta w. Por lo tanto, S decide $A_{\texttt{TM}}$.

Resumen

Resumen

Hoy aprendimos:

- Cómo reducir la decidibilidad de un lenguaje a la decidibilidad de otro.