Teoria da Computação

Prof. Maicon R. Zatelli

Aula 7 - Redutibilidade

Universidade Federal de Santa Catarina
Florianópolis - Brasil

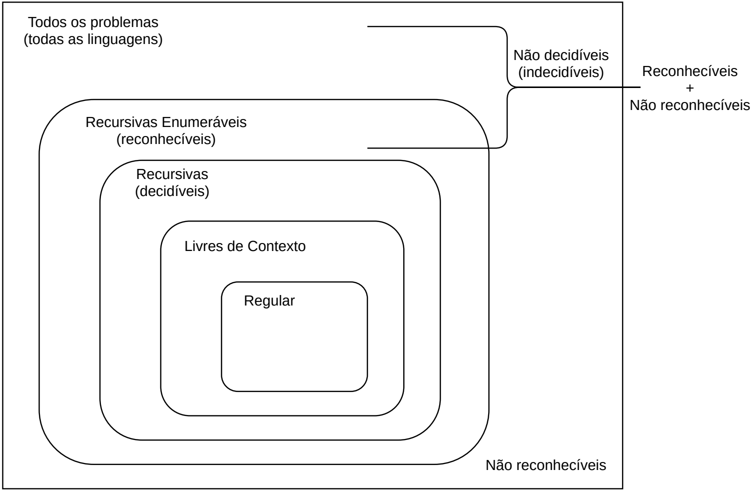
Introdução

- Linguagens Turing-reconhecíveis
- Provas através de redução

Material de apoio

- Livro Sipser, Capítulo 5
- Livro Hopcroft, Capítulo 9

Hierarquia de Chomsky



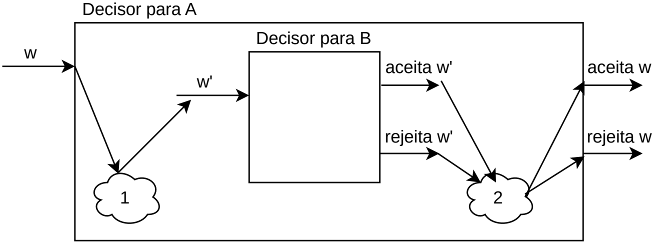
Indecidíveis - são todas as linguagens Turing-reconhecíveis mas não decidíveis e também as linguagens não Turing-reconhecíveis

Redutibilidade

Redução é uma forma de converter um problema em outro de forma que uma solução para o segundo problema pode ser usada para resolver o primeiro

Assim,

- Se A é redutível para B e B é decidível, então A é decidível
- Se A é redutível para B e A é indecidível, então B é indecidível



Na figura acima, 1 indica a transformação da entrada e 2 indica a transformação da saída

Problema da Parada: Determinar se uma MT para, aceitando ou rejeitando, com determinada entrada

Redutibilidade

Problema da Parada: Determinar se uma MT para, aceitando ou rejeitando, com determinada entrada

Linguagem: $Halt_{TM} = \{< M, w > | M$ é uma MT e M para com a entrada $w\}$

Redutibilidade

Problema da Parada: Determinar se uma MT para, aceitando ou rejeitando, com determinada entrada

Linguagem: $Halt_{TM} = \{< M, w > | M$ é uma MT e M para com a entrada $w\}$

Teorema: $Halt_{TM}$ é indecidível

Redutibilidade

Problema da Parada: Determinar se uma MT para, aceitando ou rejeitando, com determinada entrada

Linguagem: $Halt_{TM} = \{< M, w > | M$ é uma MT e M para com a entrada $w\}$

Teorema: $Halt_{TM}$ é indecidível

Prova: por contradição, usando redução

Redutibilidade

Problema da Parada: Determinar se uma MT para, aceitando ou rejeitando, com determinada entrada

Linguagem: $Halt_{TM} = \{< M, w > | M$ é uma MT e M para com a entrada $w\}$

Teorema: $Halt_{TM}$ é indecidível

Prova: por contradição, usando redução

- Sabemos que A_{TM} é indecidível

Problema da Parada: Determinar se uma MT para, aceitando ou rejeitando, com determinada entrada

Linguagem: $Halt_{TM} = \{< M, w > | M$ é uma MT e M para com a entrada $w\}$

Teorema: $Halt_{TM}$ é indecidível

Prova: por contradição, usando redução

- Sabemos que A_{TM} é indecidível
- Podemos reduzir A_{TM} para $Halt_{TM}$

Redutibilidade

Problema da Parada: Determinar se uma MT para, aceitando ou rejeitando, com determinada entrada

Linguagem: $Halt_{TM} = \{< M, w > | M$ é uma MT e M para com a entrada $w\}$

Teorema: $Halt_{TM}$ é indecidível

Prova: por contradição, usando redução

- Sabemos que A_{TM} é indecidível
- Podemos reduzir A_{TM} para $Halt_{TM}$
- Construimos um decisor para A_{TM} usando um suposto decisor para $Halt_{TM}$

Problema da Parada: Determinar se uma MT para, aceitando ou rejeitando, com determinada entrada

Linguagem: $Halt_{TM} = \{< M, w > | M$ é uma MT e M para com a entrada $w\}$

Teorema: $Halt_{TM}$ é indecidível

Prova: por contradição, usando redução

- Sabemos que A_{TM} é indecidível
- Podemos reduzir A_{TM} para $Halt_{TM}$
- Construimos um decisor para A_{TM} usando um suposto decisor para $Halt_{TM}$
- Para isso, transformamos a entrada de A_{TM} em uma entrada para $Halt_{TM}$ e a saída de $Halt_{TM}$ para a saída de A_{TM}

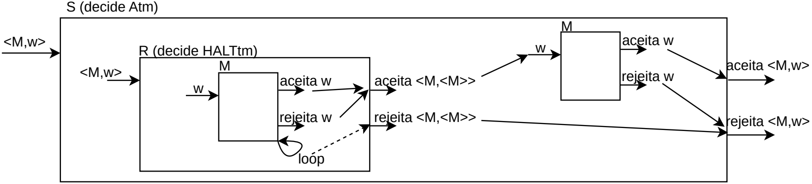
Problema da Parada: Determinar se uma MT para, aceitando ou rejeitando, com determinada entrada

Linguagem: $Halt_{TM} = \{< M, w > | M$ é uma MT e M para com a entrada $w\}$

Teorema: $Halt_{TM}$ é indecidível

Prova: por contradição, usando redução

- Suponha que $Halt_{TM}$ é decidível, então existe um decisor R para $Halt_{TM}$
- Podemos construir um decisor S para A_{TM} que usa R como subrotina

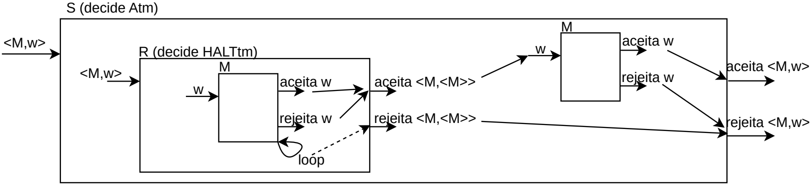


Problema da Parada: Determinar se uma MT para, aceitando ou rejeitando, com determinada entrada

Linguagem: $Halt_{TM} = \{< M, w > | M$ é uma MT e M para com a entrada $w\}$

Teorema: $Halt_{TM}$ é indecidível

Prova: por contradição, usando redução



- Note que se M aceita/rejeita então M para com a entrada w. Assim, é só rodar M com w e copiar para a saída
- Se M entra em loop, então R rejeita e S também deve rejeitar

Redutibilidade

Problema da Parada: Determinar se uma MT para, aceitando ou rejeitando, com determinada entrada

Linguagem: $Halt_{TM} = \{< M, w > | M$ é uma MT e M para com a entrada $w\}$

Teorema: $Halt_{TM}$ é indecidível

Prova: por contradição, usando redução

```
S com a entrada <M,w>, onde M é uma MT e w é uma palavra, faz:
    1: Rode R com a entrada <M,w>
    2: Se R rejeita <M,w>, rejeite (S rejeita <M,w>)
    3: Se R aceita <M,w>, rode M com a entrada w
        3.1: Se M aceita w, aceite (S aceita <M,w>)
        3.2: Se M rejeita w, rejeite (S rejeita <M,w>)
```

Redutibilidade

Problema da Parada: Determinar se uma MT para, aceitando ou rejeitando, com determinada entrada

Linguagem: $Halt_{TM} = \{< M, w > | M$ é uma MT e M para com a entrada $w\}$

Teorema: $Halt_{TM}$ é indecidível

Prova: por contradição, usando redução

```
S com a entrada <M,w>, onde M é uma MT e w é uma palavra, faz:
    1: Rode R com a entrada <M,w>
    2: Se R rejeita <M,w>, rejeite (S rejeita <M,w>)
    3: Se R aceita <M,w>, rode M com a entrada w
        3.1: Se M aceita w, aceite (S aceita <M,w>)
        3.2: Se M rejeita w, rejeite (S rejeita <M,w>)
```

- Portanto, se R decide $Halt_{TM}$, então S decide A_{TM}

Redutibilidade

Problema da Parada: Determinar se uma MT para, aceitando ou rejeitando, com determinada entrada

Linguagem: $Halt_{TM} = \{< M, w > | M$ é uma MT e M para com a entrada $w\}$

Teorema: $Halt_{TM}$ é indecidível

Prova: por contradição, usando redução

```
S com a entrada <M,w>, onde M é uma MT e w é uma palavra, faz:
    1: Rode R com a entrada <M,w>
    2: Se R rejeita <M,w>, rejeite (S rejeita <M,w>)
    3: Se R aceita <M,w>, rode M com a entrada w
        3.1: Se M aceita w, aceite (S aceita <M,w>)
        3.2: Se M rejeita w, rejeite (S rejeita <M,w>)
```

- Portanto, se R decide $Halt_{TM}$, então S decide A_{TM}
- Sabemos que A_{TM} é indecidível e então há uma contradição

Redutibilidade

Problema da Parada: Determinar se uma MT para, aceitando ou rejeitando, com determinada entrada

Linguagem: $Halt_{TM} = \{< M, w > | M$ é uma MT e M para com a entrada $w\}$

Teorema: $Halt_{TM}$ é indecidível

Prova: por contradição, usando redução

```
S com a entrada <M,w>, onde M é uma MT e w é uma palavra, faz:
    1: Rode R com a entrada <M,w>
    2: Se R rejeita <M,w>, rejeite (S rejeita <M,w>)
    3: Se R aceita <M,w>, rode M com a entrada w
        3.1: Se M aceita w, aceite (S aceita <M,w>)
        3.2: Se M rejeita w, rejeite (S rejeita <M,w>)
```

- Portanto, se R decide $Halt_{TM}$, então S decide A_{TM}
- Sabemos que A_{TM} é indecidível e então há uma contradição
- S não pode existir e assim R também não pode existir

Redutibilidade

Problema da Parada: Determinar se uma MT para, aceitando ou rejeitando, com determinada entrada

Linguagem: $Halt_{TM} = \{< M, w > | M$ é uma MT e M para com a entrada $w\}$

Teorema: $Halt_{TM}$ é indecidível

Note que provando $Halt_{TM}$ ser indecidível também mostramos que não há um algoritmo que receba como entrada outro algoritmo e certa entrada e determine se o algoritmo para ou não com a entrada fornecida

Redutibilidade

Problema da Parada: Determinar se uma MT para, aceitando ou rejeitando, com determinada entrada

Linguagem: $Halt_{TM} = \{< M, w > |M$ é uma MT e M para com a entrada $w\}$

Teorema: $Halt_{TM}$ é indecidível

Note que provando $Halt_{TM}$ ser indecidível também mostramos que não há um algoritmo que receba como entrada outro algoritmo e certa entrada e determine se o algoritmo para ou não com a entrada fornecida

- Além disso, se o problema da parada fosse decidível, toda linguagem recursivamente enumerável seria uma linguagem recursiva

Problema da Parada: Determinar se uma MT para, aceitando ou rejeitando, com determinada entrada

Linguagem: $Halt_{TM} = \{< M, w > |M$ é uma MT e M para com a entrada $w\}$

Teorema: $Halt_{TM}$ é indecidível

Note que provando $Halt_{TM}$ ser indecidível também mostramos que não há um algoritmo que receba como entrada outro algoritmo e certa entrada e determine se o algoritmo para ou não com a entrada fornecida

- Além disso, se o problema da parada fosse decidível, toda linguagem recursivamente enumerável seria uma linguagem recursiva
- Ou seja, todos os problemas seriam decidíveis

Redutibilidade

Problema: Determinar se uma MT aceita a linguagem vazia

Redutibilidade

Problema: Determinar se uma MT aceita a linguagem vazia

Linguagem: $E_{TM} = \{< M > | M \text{ é uma MT e } L(M) = \emptyset\}$

Redutibilidade

Problema: Determinar se uma MT aceita a linguagem vazia

Linguagem: $E_{TM} = \{< M > | M$ é uma MT e $L(M) = \emptyset\}$

Teorema: E_{TM} é indecidível

Redutibilidade

Problema: Determinar se uma MT aceita a linguagem vazia

Linguagem: $E_{TM} = \{< M > | M$ é uma MT e $L(M) = \emptyset\}$

Teorema: E_{TM} é indecidível

Prova: por contradição, usando redução

Redutibilidade

Problema: Determinar se uma MT aceita a linguagem vazia

Linguagem: $E_{TM} = \{< M > | M$ é uma MT e $L(M) = \emptyset\}$

Teorema: E_{TM} é indecidível

Prova: por contradição, usando redução

- Vamos reduzir A_{TM} para E_{TM}, sabendo que A_{TM} é indecidível

Redutibilidade

Problema: Determinar se uma MT aceita a linguagem vazia

Linguagem: $E_{TM} = \{< M > | M \text{ é uma MT e } L(M) = \emptyset\}$

Teorema: E_{TM} é indecidível

Prova: por contradição, usando redução

- Vamos reduzir A_{TM} para E_{TM}, sabendo que A_{TM} é indecidível
- Suponha que E_{TM} é decidível, então existe um decisor R para E_{TM}

Problema: Determinar se uma MT aceita a linguagem vazia

Linguagem: $E_{TM} = \{< M > | M$ é uma MT e $L(M) = \emptyset\}$

Teorema: E_{TM} é indecidível

Prova: por contradição, usando redução

- Vamos reduzir A_{TM} para E_{TM}, sabendo que A_{TM} é indecidível
- Suponha que E_{TM} é decidível, então existe um decisor R para E_{TM}
- Podemos construir um decisor S para A_{TM} utilizando R como subrotina

Redutibilidade

Problema: Determinar se uma MT aceita a linguagem vazia

Linguagem: $E_{TM} = \{< M > | M$ é uma MT e $L(M) = \emptyset\}$

Teorema: E_{TM} é indecidível

Prova: por contradição, usando redução

- Vamos reduzir A_{TM} para E_{TM}, sabendo que A_{TM} é indecidível
- Suponha que E_{TM} é decidível, então existe um decisor R para E_{TM}
- Podemos construir um decisor S para A_{TM} utilizando R como subrotina
- A ideia é quando uma MT aceita $\emptyset$ significa que M não aceita w

Redutibilidade

Problema: Determinar se uma MT aceita a linguagem vazia

Linguagem: $E_{TM} = \{< M > | M \text{ é uma MT e } L(M) = \emptyset\}$

Teorema: E_{TM} é indecidível

Prova: por contradição, usando redução

- Vamos reduzir A_{TM} para E_{TM}, sabendo que A_{TM} é indecidível
- Suponha que E_{TM} é decidível, então existe um decisor R para E_{TM}
- Podemos construir um decisor S para A_{TM} utilizando R como subrotina
- A ideia é quando uma MT aceita $\emptyset$ significa que M não aceita w
- Quando uma MT não aceita $\emptyset$ significa que ela aceita apenas w (e M aceita w)

Redutibilidade

Problema: Determinar se uma MT aceita a linguagem vazia

Linguagem: $E_{TM} = \{< M > | M$ é uma MT e $L(M) = \emptyset\}$

Teorema: E_{TM} é indecidível

Prova: por contradição, usando redução

- Vamos inicialmente precisar criar uma MT Mw

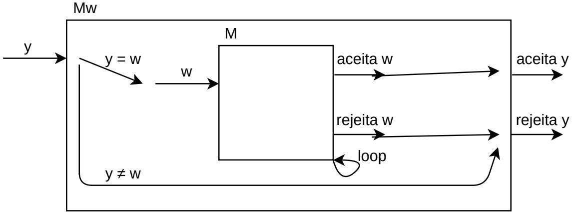
```
Mw com a entrada y, onde y é uma palavra, faz:
    1: Se y = w, rode M com a entrada w
        1.1: Se M aceita w, aceite y
        1.2: Se M rejeita w, rejeite y
    2: Senão (se y ≠ w), rejeite y
```

Problema: Determinar se uma MT aceita a linguagem vazia

Linguagem: $E_{TM} = \{< M > | M$ é uma MT e $L(M) = \emptyset\}$

Teorema: E_{TM} é indecidível

Prova: por contradição, usando redução



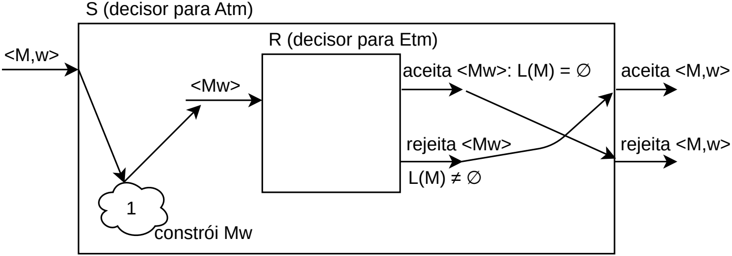
- Mw é uma MT que reconhece ou $\{w\}$ ou $\emptyset$, visto que rejeita tudo, exceto w e aceita w somente se M também aceita w
- Assim,
  - Mw aceita $\emptyset$ se e somente se M não aceita w
  - Mw não aceita $\emptyset$ se e somente se M aceita w

Problema: Determinar se uma MT aceita a linguagem vazia

Linguagem: $E_{TM} = \{< M > | M \text{ é uma MT e } L(M) = \emptyset\}$

Teorema: E_{TM} é indecidível

Prova: por contradição, usando redução



Redutibilidade

Problema: Determinar se uma MT aceita a linguagem vazia

Linguagem: $E_{TM} = \{< M > | M \text{ é uma MT e } L(M) = \emptyset\}$

Teorema: E_{TM} é indecidível

Prova: por contradição, usando redução

```
S com a entrada <M,w>, onde M é uma MT e w é uma palavra, faz:
    1: Construa a MT Mw
    2: Rode R com a entrada <Mw>
        2.1: Se R aceita <Mw>, rejeite (S rejeita <M,w>)
        2.2: Se R rejeita <Mw>, aceite (S aceita <M,w>)
```

Redutibilidade

Problema: Determinar se uma MT aceita a linguagem vazia

Linguagem: $E_{TM} = \{< M > | M \text{ é uma MT e } L(M) = \emptyset\}$

Teorema: E_{TM} é indecidível

Prova: por contradição, usando redução

```
S com a entrada <M,w>, onde M é uma MT e w é uma palavra, faz:
    1: Construa a MT Mw
    2: Rode R com a entrada <Mw>
        2.1: Se R aceita <Mw>, rejeite (S rejeita <M,w>)
        2.2: Se R rejeita <Mw>, aceite (S aceita <M,w>)
```

- Portanto, se R decide E_{TM}, então S decide A_{TM}

Problema: Determinar se uma MT aceita a linguagem vazia

Linguagem: $E_{TM} = \{< M > | M \text{ é uma MT e } L(M) = \emptyset\}$

Teorema: E_{TM} é indecidível

Prova: por contradição, usando redução

```
S com a entrada <M,w>, onde M é uma MT e w é uma palavra, faz:
    1: Construa a MT Mw
    2: Rode R com a entrada <Mw>
        2.1: Se R aceita <Mw>, rejeite (S rejeita <M,w>)
        2.2: Se R rejeita <Mw>, aceite (S aceita <M,w>)
```

- Portanto, se R decide E_{TM}, então S decide A_{TM}
- Sabemos que A_{TM} é indecidível e então há uma contradição

Problema: Determinar se uma MT aceita a linguagem vazia

Linguagem: $E_{TM} = \{< M > | M \text{ é uma MT e } L(M) = \emptyset\}$

Teorema: E_{TM} é indecidível

Prova: por contradição, usando redução

```
S com a entrada <M,w>, onde M é uma MT e w é uma palavra, faz:
    1: Construa a MT Mw
    2: Rode R com a entrada <Mw>
        2.1: Se R aceita <Mw>, rejeite (S rejeita <M,w>)
        2.2: Se R rejeita <Mw>, aceite (S aceita <M,w>)
```

- Portanto, se R decide E_{TM}, então S decide A_{TM}
- Sabemos que A_{TM} é indecidível e então há uma contradição
- S não pode existir e assim R também não pode existir

Problema: Determinar se uma MT aceita uma linguagem regular

Problema: Determinar se uma MT aceita uma linguagem regular

Linguagem: $REGULAR_{TM} = \{< M > | M$ é uma MT e $L(M)$ é regular $\}$

Redutibilidade

Problema: Determinar se uma MT aceita uma linguagem regular

Linguagem: $REGULAR_{TM} = \{< M > | M$ é uma MT e $L(M)$ é regular $\}$

Teorema: $REGULAR_{TM}$ é indecidível

Redutibilidade

Problema: Determinar se uma MT aceita uma linguagem regular

Linguagem: $REGULAR_{TM} = \{< M > | M$ é uma MT e $L(M)$ é regular $\}$

Teorema: $REGULAR_{TM}$ é indecidível

Prova: por contradição, usando redução

Redutibilidade

Problema: Determinar se uma MT aceita uma linguagem regular

Linguagem: $REGULAR_{TM} = \{< M > | M$ é uma MT e $L(M)$ é regular $\}$

Teorema: $REGULAR_{TM}$ é indecidível

Prova: por contradição, usando redução

- Vamos reduzir A_{TM} para $REGULAR_{TM}$, sabendo que A_{TM} é indecidível

Redutibilidade

Problema: Determinar se uma MT aceita uma linguagem regular

Linguagem: $REGULAR_{TM} = \{< M > | M$ é uma MT e $L(M)$ é regular $\}$

Teorema: $REGULAR_{TM}$ é indecidível

Prova: por contradição, usando redução

- Vamos reduzir A_{TM} para $REGULAR_{TM}$, sabendo que A_{TM} é indecidível
- Suponha que $REGULAR_{TM}$ é decidível, então existe um decisor R para $REGULAR_{TM}$

Redutibilidade

Problema: Determinar se uma MT aceita uma linguagem regular

Linguagem: $REGULAR_{TM} = \{< M > | M$ é uma MT e $L(M)$ é regular $\}$

Teorema: $REGULAR_{TM}$ é indecidível

Prova: por contradição, usando redução

- Vamos reduzir A_{TM} para $REGULAR_{TM}$, sabendo que A_{TM} é indecidível
- Suponha que $REGULAR_{TM}$ é decidível, então existe um decisor R para $REGULAR_{TM}$
- Podemos construir um decisor S para A_{TM} utilizando R como subrotina

Redutibilidade

Problema: Determinar se uma MT aceita uma linguagem regular

Linguagem: $REGULAR_{TM} = \{< M > | M$ é uma MT e $L(M)$ é regular $\}$

Teorema: $REGULAR_{TM}$ é indecidível

Prova: por contradição, usando redução

- Vamos reduzir A_{TM} para $REGULAR_{TM}$, sabendo que A_{TM} é indecidível
- Suponha que $REGULAR_{TM}$ é decidível, então existe um decisor R para $REGULAR_{TM}$
- Podemos construir um decisor S para A_{TM} utilizando R como subrotina
- A ideia é quando uma MT aceita uma linguagem regular significa que M aceita w

Redutibilidade

Problema: Determinar se uma MT aceita uma linguagem regular

Linguagem: $REGULAR_{TM} = \{< M > | M$ é uma MT e $L(M)$ é regular $\}$

Teorema: $REGULAR_{TM}$ é indecidível

Prova: por contradição, usando redução

- Vamos reduzir A_{TM} para $REGULAR_{TM}$, sabendo que A_{TM} é indecidível
- Suponha que $REGULAR_{TM}$ é decidível, então existe um decisor R para $REGULAR_{TM}$
- Podemos construir um decisor S para A_{TM} utilizando R como subrotina
- A ideia é quando uma MT aceita uma linguagem regular significa que M aceita w
- Quando uma MT aceita uma linguagem não regular significa que M não aceita w

Redutibilidade

Problema: Determinar se uma MT aceita uma linguagem regular

Linguagem: $REGULAR_{TM} = \{< M > | M$ é uma MT e $L(M)$ é regular $\}$

Teorema: $REGULAR_{TM}$ é indecidível

Prova: por contradição, usando redução

- Vamos inicialmente precisar criar uma MT *Mw* que aceita uma LR ou uma não LR

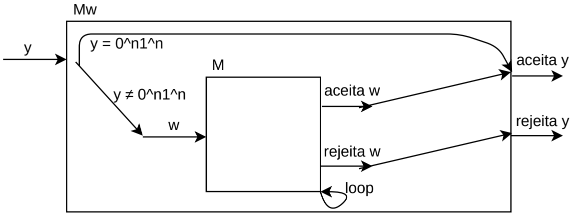
```
Mw com a entrada y, onde y é uma palavra, faz:
  1: Se y é da forma 0^n1^n, aceite y
  2: Senão (se y não é da forma 0^n1^n), rode M com a entrada w
    2.1: Se M aceita w, aceite y
    2.2: Se M rejeita w, rejeite y
```

Problema: Determinar se uma MT aceita uma linguagem regular

Linguagem: $REGULAR_{TM} = \{< M > | M$ é uma MT e $L(M)$ é regular $\}$

Teorema: $REGULAR_{TM}$ é indecidível

Prova: por contradição, usando redução



- Mw é uma MT que reconhece ou $\{0^n 1^n\}$ ou Σ^*
- Assim,
  - Mw aceita Σ^* (LR) se e somente se M aceita w
  - Mw aceita $\{0^n 1^n\}$ (não-LR) se e somente se M não aceita w

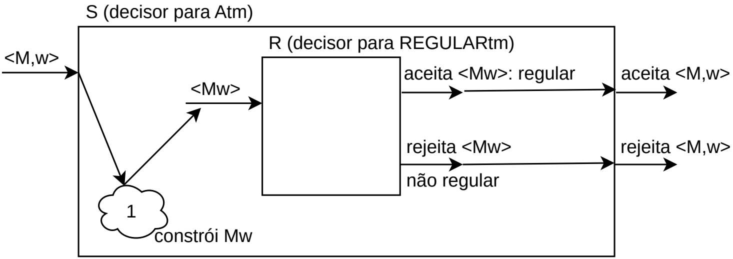
Redutibilidade

Problema: Determinar se uma MT aceita uma linguagem regular

Linguagem: $REGULAR_{TM} = \{< M > | M$ é uma MT e $L(M)$ é regular $\}$

Teorema: $REGULAR_{TM}$ é indecidível

Prova: por contradição, usando redução



Redutibilidade

Problema: Determinar se uma MT aceita uma linguagem regular

Linguagem: $REGULAR_{TM} = \{< M > | M$ é uma MT e $L(M)$ é regular $\}$

Teorema: $REGULAR_{TM}$ é indecidível

Prova: por contradição, usando redução

```
S com a entrada <M,w>, onde M é uma MT e w é uma palavra, faz:
    1: Construa a MT Mw
    2: Rode R com a entrada <Mw>
        2.1: Se R aceita <Mw>, aceite (S aceita <M,w>)
        2.2: Se R rejeita <Mw>, rejeite (S rejeita <M,w>)
```

Redutibilidade

Problema: Determinar se uma MT aceita uma linguagem regular

Linguagem: $REGULAR_{TM} = \{< M > | M$ é uma MT e $L(M)$ é regular $\}$

Teorema: $REGULAR_{TM}$ é indecidível

Prova: por contradição, usando redução

```
S com a entrada <M,w>, onde M é uma MT e w é uma palavra, faz:
    1: Construa a MT Mw
    2: Rode R com a entrada <Mw>
        2.1: Se R aceita <Mw>, aceite (S aceita <M,w>)
        2.2: Se R rejeita <Mw>, rejeite (S rejeita <M,w>)
```

- Portanto, se R decide $REGULAR_{TM}$, então S decide A_{TM}

Redutibilidade

Problema: Determinar se uma MT aceita uma linguagem regular

Linguagem: $REGULAR_{TM} = \{< M > | M$ é uma MT e $L(M)$ é regular $\}$

Teorema: $REGULAR_{TM}$ é indecidível

Prova: por contradição, usando redução

```
S com a entrada <M,w>, onde M é uma MT e w é uma palavra, faz:
    1: Construa a MT Mw
    2: Rode R com a entrada <Mw>
        2.1: Se R aceita <Mw>, aceite (S aceita <M,w>)
        2.2: Se R rejeita <Mw>, rejeite (S rejeita <M,w>)
```

- Portanto, se R decide $REGULAR_{TM}$, então S decide A_{TM}
- Sabemos que A_{TM} é indecidível e então há uma contradição

Redutibilidade

Problema: Determinar se uma MT aceita uma linguagem regular

Linguagem: $REGULAR_{TM} = \{< M > | M$ é uma MT e $L(M)$ é regular $\}$

Teorema: $REGULAR_{TM}$ é indecidível

Prova: por contradição, usando redução

```
S com a entrada <M,w>, onde M é uma MT e w é uma palavra, faz:
    1: Construa a MT Mw
    2: Rode R com a entrada <Mw>
        2.1: Se R aceita <Mw>, aceite (S aceita <M,w>)
        2.2: Se R rejeita <Mw>, rejeite (S rejeita <M,w>)
```

- Portanto, se R decide $REGULAR_{TM}$, então S decide A_{TM}
- Sabemos que A_{TM} é indecidível e então há uma contradição
- S não pode existir e assim R também não pode existir

Redutibilidade

Problema: Determinar se duas MT reconhecem a mesma linguagem (dois programas são equivalentes?)

Redutibilidade

Problema: Determinar se duas MT reconhecem a mesma linguagem (dois programas são equivalentes?)

Linguagem: $EQ_{TM} = \{< M1, M2 > | M1, M2$ são MTs e $L(M1) = L(M2)\}$

Redutibilidade

Problema: Determinar se duas MT reconhecem a mesma linguagem (dois programas são equivalentes?)

Linguagem: $EQ_{TM} = \{< M1, M2 > | M1, M2$ são MTs e $L(M1) = L(M2)\}$

Teorema: EQ_{TM} é indecidível

Redutibilidade

Problema: Determinar se duas MT reconhecem a mesma linguagem (dois programas são equivalentes?)

Linguagem: $EQ_{TM} = \{< M1, M2 > | M1, M2$ são MTs e $L(M1) = L(M2)\}$

Teorema: EQ_{TM} é indecidível

Prova: por contradição, usando redução

Redutibilidade

Problema: Determinar se duas MT reconhecem a mesma linguagem (dois programas são equivalentes?)

Linguagem: $EQ_{TM} = \{< M1, M2 > | M1, M2$ são MTs e $L(M1) = L(M2)\}$

Teorema: EQ_{TM} é indecidível

Prova: por contradição, usando redução

- Vamos reduzir E_{TM} para EQ_{TM}, com E_{TM} indecidível

Problema: Determinar se duas MT reconhecem a mesma linguagem (dois programas são equivalentes?)

Linguagem: $EQ_{TM} = \{< M1, M2 > | M1, M2$ são MTs e $L(M1) = L(M2)\}$

Teorema: EQ_{TM} é indecidível

Prova: por contradição, usando redução

- Vamos reduzir E_{TM} para EQ_{TM}, com E_{TM} indecidível
- Suponha que EQ_{TM} é decidível, então existe um decisor R para EQ_{TM}

Redutibilidade

Problema: Determinar se duas MT reconhecem a mesma linguagem (dois programas são equivalentes?)

Linguagem: $EQ_{TM} = \{< M1, M2 > | M1, M2$ são MTs e $L(M1) = L(M2)\}$

Teorema: EQ_{TM} é indecidível

Prova: por contradição, usando redução

- Vamos reduzir E_{TM} para EQ_{TM}, com E_{TM} indecidível
- Suponha que EQ_{TM} é decidível, então existe um decisor R para EQ_{TM}
- Podemos construir um decisor S para E_{TM} utilizando R como subrotina

Redutibilidade

Problema: Determinar se duas MT reconhecem a mesma linguagem (dois programas são equivalentes?)

Linguagem: $EQ_{TM} = \{< M1, M2 > | M1, M2$ são MTs e $L(M1) = L(M2)\}$

Teorema: EQ_{TM} é indecidível

Prova: por contradição, usando redução

- Vamos reduzir E_{TM} para EQ_{TM}, com E_{TM} indecidível
- Suponha que EQ_{TM} é decidível, então existe um decisor R para EQ_{TM}
- Podemos construir um decisor S para E_{TM} utilizando R como subrotina
- A ideia é ter uma MT Mx que rejeita tudo e ver se M (entrada de S) é equivalente a Mx

Redutibilidade

Problema: Determinar se duas MT reconhecem a mesma linguagem (dois programas são equivalentes?)

Linguagem: $EQ_{TM} = \{< M1, M2 > | M1, M2$ são MTs e $L(M1) = L(M2)\}$

Teorema: EQ_{TM} é indecidível

Prova: por contradição, usando redução

- Vamos reduzir E_{TM} para EQ_{TM}, com E_{TM} indecidível
- Suponha que EQ_{TM} é decidível, então existe um decisor R para EQ_{TM}
- Podemos construir um decisor S para E_{TM} utilizando R como subrotina
- A ideia é ter uma MT Mx que rejeita tudo e ver se M (entrada de S) é equivalente a Mx
- Se M for equivalente a Mx sabemos que M reconhece a linguagem $\emptyset$, caso contrário, ela não reconhece a linguagem $\emptyset$

Redutibilidade

Problema: Determinar se duas MT reconhecem a mesma linguagem (dois programas são equivalentes?)

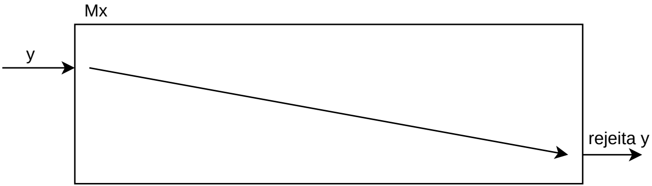
Linguagem: $EQ_{TM} = \{< M1, M2 > | M1, M2$ são MTs e $L(M1) = L(M2)\}$

Teorema: EQ_{TM} é indecidível

Prova: por contradição, usando redução

- Vamos inicialmente precisar criar uma MT Mx que rejeita tudo

```
Mx com a entrada y, onde y é uma palavra, faz:
  1: rejeite (Mx rejeita y)
```

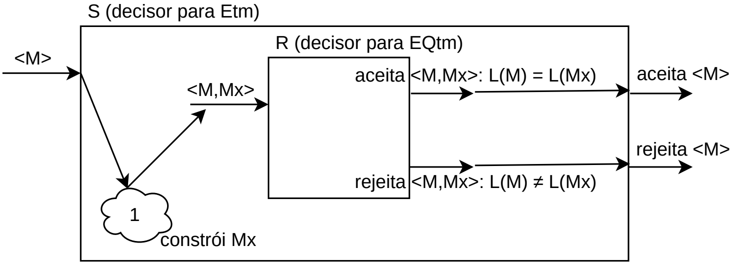


Problema: Determinar se duas MT reconhecem a mesma linguagem (dois programas são equivalentes?)

Linguagem: $EQ_{TM} = \{< M1, M2 > | M1, M2$ são MTs e $L(M1) = L(M2)\}$

Teorema: EQ_{TM} é indecidível

Prova: por contradição, usando redução



Redutibilidade

Problema: Determinar se duas MT reconhecem a mesma linguagem (dois programas são equivalentes?)

Linguagem: $EQ_{TM} = \{< M1, M2 > | M1, M2$ são MTs e $L(M1) = L(M2)\}$

Teorema: EQ_{TM} é indecidível

Prova: por contradição, usando redução

```
S com a entrada <M>, onde M é uma MT, faz:
    1: Construa a MT Mx que aceita ∅
    2: Rode R com a entrada <M,Mx>
        2.1: Se R aceita <M,Mx>, aceite (S aceita <M>)
        2.2: Se R rejeita <M,Mx>, rejeite (S rejeita <M>)
```

Problema: Determinar se duas MT reconhecem a mesma linguagem (dois programas são equivalentes?)

Linguagem: $EQ_{TM} = \{< M1, M2 > | M1, M2$ são MTs e $L(M1) = L(M2)\}$

Teorema: EQ_{TM} é indecidível

Prova: por contradição, usando redução

```
S com a entrada <M>, onde M é uma MT, faz:
    1: Construa a MT Mx que aceita ∅
    2: Rode R com a entrada <M,Mx>
        2.1: Se R aceita <M,Mx>, aceite (S aceita <M>)
        2.2: Se R rejeita <M,Mx>, rejeite (S rejeita <M>)
```

- Portanto, se R decide EQ_{TM}, então S decide E_{TM}

Redutibilidade

Problema: Determinar se duas MT reconhecem a mesma linguagem (dois programas são equivalentes?)

Linguagem: $EQ_{TM} = \{< M1, M2 > | M1, M2$ são MTs e $L(M1) = L(M2)\}$

Teorema: EQ_{TM} é indecidível

Prova: por contradição, usando redução

```
S com a entrada <M>, onde M é uma MT, faz:
    1: Construa a MT Mx que aceita ∅
    2: Rode R com a entrada <M,Mx>
        2.1: Se R aceita <M,Mx>, aceite (S aceita <M>)
        2.2: Se R rejeita <M,Mx>, rejeite (S rejeita <M>)
```

- Portanto, se R decide EQ_{TM}, então S decide E_{TM}
- Sabemos que E_{TM} é indecidível e então há uma contradição

Redutibilidade

Problema: Determinar se duas MT reconhecem a mesma linguagem (dois programas são equivalentes?)

Linguagem: $EQ_{TM} = \{< M1, M2 > | M1, M2$ são MTs e $L(M1) = L(M2)\}$

Teorema: EQ_{TM} é indecidível

Prova: por contradição, usando redução

```
S com a entrada <M>, onde M é uma MT, faz:
    1: Construa a MT Mx que aceita ∅
    2: Rode R com a entrada <M,Mx>
        2.1: Se R aceita <M,Mx>, aceite (S aceita <M>)
        2.2: Se R rejeita <M,Mx>, rejeite (S rejeita <M>)
```

- Portanto, se R decide EQ_{TM}, então S decide E_{TM}
- Sabemos que E_{TM} é indecidível e então há uma contradição
- S não pode existir e assim R também não pode existir

Redutibilidade por Mapeamento

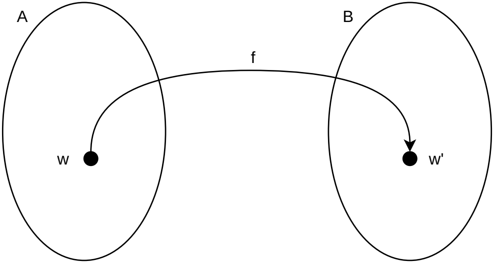
Ser capaz de reduzir o problema A para o problema B usando uma redução por mapeamento significa que existe uma função computável que converte instâncias do problema A para instâncias do problema B

Se tivermos tal função de conversão, denominada redução, podemos resolver o problema A com um solucionador para o problema B

Redutibilidade por Mapeamento

A linguagem A é dita redutível por mapeamento para a linguagem B, escrito $A \leq_m B$, se existe uma função computável $f : \Sigma^* \rightarrow \Sigma^*$, onde para toda palavra/entrada w tem-se que $w \in A \Leftrightarrow f(w) \in B$

A função f é denominada a redução de A para B



Redutibilidade

Teorema 23: Se $A \leq_m B$ e B é decidível, então A é decidível

Redutibilidade

Teorema 23: Se $A \leq_m B$

Teorema 23: Se $A \leq_m B$ e B é decidível, então A é decidível

Prova: por construção

- Se B é decidível, então seja M um decisor para B

Teorema 23: Se $A \leq_m B$ e B é decidível, então A é decidível

Prova: por construção

- Se B é decidível, então seja M um decisor para B
- Como A é redutível para B, então seja f a redução de A para B

Redutibilidade

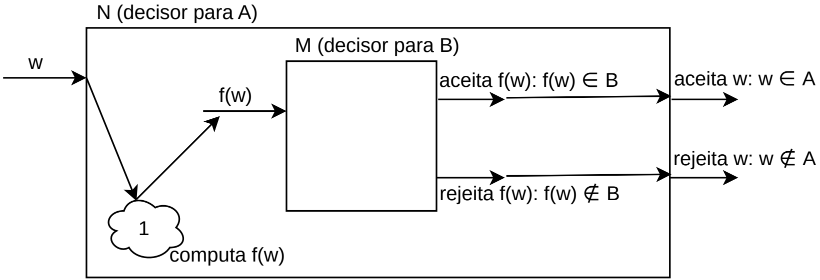
Teorema 23: Se $A \leq_m B$

Redutibilidade

Teorema 23: Se $A \leq_m B$

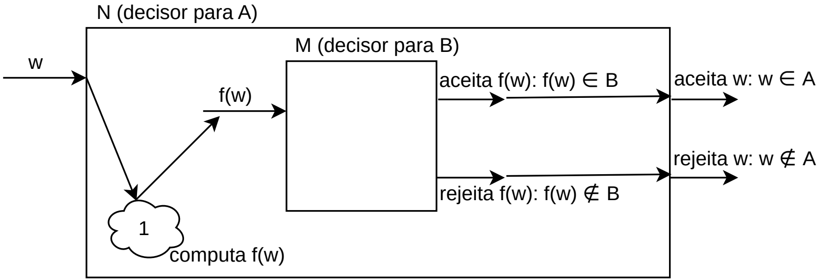
Teorema 22: Se $A \leq_m B$ e B é decidível, então A é decidível

Prova: por construção



Teorema 22: Se $A \leq_m B$ e B é decidível, então A é decidível

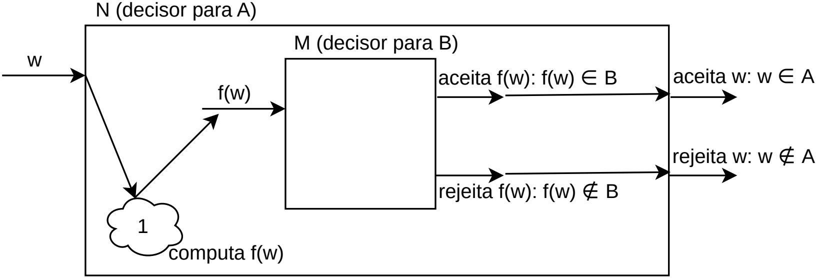
Prova: por construção



- Se $w \in A$, então $f(w) \in B$ pois f é uma redução de A para B

Teorema 22: Se $A \leq_m B$ e B é decidível, então A é decidível

Prova: por construção



- Se $w \in A$, então $f(w) \in B$ pois f é uma redução de A para B
- Assim, M aceita $f(w)$ sempre que $w \in A$

Redutibilidade

Corolário 23: Se $A \leq_m B$ e A é indecidível, então B é indecidível

Corolário 23: Se $A \leq_m B$ e A é indecidível, então B é indecidível

Prova: por contradição

Corolário 23: Se $A \leq_m B$ e A é indecidível, então B é indecidível

Prova: por contradição

- Suponha que $A \leq_m B$ e A é indecidível

Redutibilidade

Corolário 23: Se $A \leq_m B$ e A é indecidível, então B é indecidível

Prova: por contradição

- Suponha que $A \leq_m B$ e A é indecidível
- Suponha também que B ainda pudesse ser decidível

Corolário 23: Se $A \leq_m B$ e A é indecidível, então B é indecidível

Prova: por contradição

- Suponha que $A \leq_m B$ e A é indecidível
- Suponha também que B ainda pudesse ser decidível
- Neste caso, pelo teorema 22, se $A \leq_m B$ e B decidível, então A certamente também seria decidível e isso contradiz a suposição de que A fosse indecidível

Redutibilidade

Problema da Parada: Determinar se uma MT para, aceitando ou rejeitando, com determinada entrada

Linguagem: $Halt_{TM} = \{< M, w > | M$ é uma MT e M para com a entrada $w\}$

Teorema: $Halt_{TM}$ é indecidível

Redutibilidade

Problema da Parada: Determinar se uma MT para, aceitando ou rejeitando, com determinada entrada

Linguagem: $Halt_{TM} = \{< M, w > | M$ é uma MT e M para com a entrada $w\}$

Teorema: $Halt_{TM}$ é indecidível

Prova: usando redução por mapeamento

Problema da Parada: Determinar se uma MT para, aceitando ou rejeitando, com determinada entrada

Linguagem: $Halt_{TM} = \{< M, w > | M$ é uma MT e M para com a entrada $w\}$

Teorema: $Halt_{TM}$ é indecidível

Prova: usando redução por mapeamento

- Vamos mostrar que $A_{TM} \leq_m Halt_{TM}$

Redutibilidade

Problema da Parada: Determinar se uma MT para, aceitando ou rejeitando, com determinada entrada

Linguagem: $Halt_{TM} = \{< M, w > | M$ é uma MT e M para com a entrada $w\}$

Teorema: $Halt_{TM}$ é indecidível

Prova: usando redução por mapeamento

- Vamos mostrar que $A_{TM} \leq_m Halt_{TM}$
- Precisamos agora converter uma entrada para A_{TM}, dada por $< M, w >$, para uma entrada para $Halt_{TM}$, dada por $< M', w' >$, tal que se $< M, w > \in A_{TM}$, então $< M', w' > \in Halt_{TM}$

Redutibilidade

Problema da Parada: Determinar se uma MT para, aceitando ou rejeitando, com determinada entrada

Linguagem: $Halt_{TM} = \{< M, w > | M$ é uma MT e M para com a entrada $w\}$

Teorema: $Halt_{TM}$ é indecidível

Prova: usando redução por mapeamento

- Vamos mostrar que $A_{TM} \leq_m Halt_{TM}$
- Precisamos agora converter uma entrada para A_{TM}, dada por $< M, w >$, para uma entrada para $Halt_{TM}$, dada por $< M', w' >$, tal que se $< M, w > \in A_{TM}$, então $< M', w' > \in Halt_{TM}$
- Precisamos mostrar uma função computável f que tem como entrada $< M, w >$ e retorna $< M', w' >$

Redutibilidade

Problema da Parada: Determinar se uma MT para, aceitando ou rejeitando, com determinada entrada

Linguagem: $Halt_{TM} = \{< M, w > | M$ é uma MT e M para com a entrada $w\}$

Teorema: $Halt_{TM}$ é indecidível

Prova: usando redução por mapeamento

Problema da Parada: Determinar se uma MT para, aceitando ou rejeitando, com determinada entrada

Linguagem: $Halt_{TM} = \{< M, w > | M$ é uma MT e M para com a entrada $w\}$

Teorema: $Halt_{TM}$ é indecidível

Prova: usando redução por mapeamento

- A máquina Mf computa f da seguinte forma

Redutibilidade

Problema da Parada: Determinar se uma MT para, aceitando ou rejeitando, com determinada entrada

Linguagem: $Halt_{TM} = \{< M, w > | M$ é uma MT e M para com a entrada $w\}$

Teorema: $Halt_{TM}$ é indecidível

Prova: usando redução por mapeamento

- A máquina Mf computa f da seguinte forma

```
Mf com a entrada <M,w>, onde M é uma MT e w é uma palavra, faz:
  1: Construa a MT M′
    "M′ = com a entrada x, faz:
      1: Rode M com a entrada x
      2: Se M aceita x, aceite (M′ aceita x)
      3: Se M rejeita x, loop (M′ fica em loop)"
  2: Dê como saída <M′,w′>, onde w′ = w
```

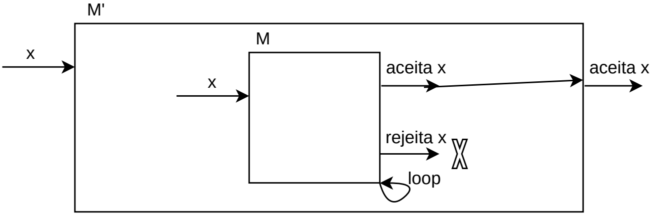
Redutibilidade

Problema da Parada: Determinar se uma MT para, aceitando ou rejeitando, com determinada entrada

Linguagem: $Halt_{TM} = \{< M, w > | M$ é uma MT e M para com a entrada $w\}$

Teorema: $Halt_{TM}$ é indecidível

Prova: usando redução por mapeamento



- M' ou aceita a entrada ou entra em loop, assim se fornecer $< M', w' >$ como entrada para $Halt_{TM}$, onde $w' = w$, ou $< M', w' > \in Halt_{TM}$ ou $< M', w' > \notin Halt_{TM}$

Redutibilidade

Teorema 24: Se $A \leq_m B$ e B é Turing-reconhecível, então A é Turing-reconhecível

Prova: idem prova do Teorema 22 (mas com reconhecedores)

Redutibilidade

Teorema 24: Se $A \leq_m B$ e B é Turing-reconhecível, então A é Turing-reconhecível

Prova: idem prova do Teorema 22 (mas com reconhecedores)

Corolário 25: Se $A \leq_m B$ e A não é Turing-reconhecível, então B não é Turing-reconhecível

Prova: idem prova do Corolário 23 (mas com reconhecedores)

Redutibilidade

- Note que a definição de redutibilidade por mapeamento implica que $A \leq_m B$ é o mesmo que $\overline{A} \leq_m \overline{B}$

- Note que a definição de redutibilidade por mapeamento implica que $A \leq_m B$ é o mesmo que $\overline{A} \leq_m \overline{B}$
- Assim, para provar que B não é Turing-reconhecível podemos mostrar que $A \leq_m \overline{B}$, uma vez que sabemos que A é indecidível e $\overline{A}$ não é Turing-reconhecível

Redutibilidade

- Note que a definição de redutibilidade por mapeamento implica que $A \leq_m B$ é o mesmo que $\overline{A} \leq_m \overline{B}$
- Assim, para provar que B não é Turing-reconhecível podemos mostrar que $A \leq_m \overline{B}$, uma vez que sabemos que A é indecidível e $\overline{A}$ não é Turing-reconhecível
- Como mostramos que $\overline{B}$ é indecidível, podemos concluir que B não é Turing-reconhecível, visto que se $A \leq_m \overline{B}$ então, por definição de redutibilidade por mapeamento, temos que $\overline{A} \leq_m B$

Teorema 30: EQ_{TM} não é nem Turing-reconhecível nem co-Turing-reconhecível

Teorema 30: EQ_{TM} não é nem Turing-reconhecível nem co-Turing-reconhecível

Prova: por construção usando redutibilidade por mapeamento

Teorema 30: EQ_{TM} não é nem Turing-reconhecível nem co-Turing-reconhecível

Prova: por construção usando redutibilidade por mapeamento

- Primeiramente, mostramos que EQ_{TM} não é Turing-reconhecível

Redutibilidade

Teorema 30: EQ_{TM} não é nem Turing-reconhecível nem co-Turing-reconhecível

Prova: por construção usando redutibilidade por mapeamento

- Primeiramente, mostramos que EQ_{TM} não é Turing-reconhecível
- Mostramos isso por meio de redução de A_{TM} para $\overline{EQ_{TM}}$

Teorema 30: EQ_{TM} não é nem Turing-reconhecível nem co-Turing-reconhecível

Prova: por construção usando redutibilidade por mapeamento

- Primeiramente, mostramos que EQ_{TM} não é Turing-reconhecível
- Mostramos isso por meio de redução de A_{TM} para $\overline{EQ_{TM}}$
- Precisamos de uma função de mapeamento que converta a entrada de A_{TM}, $< M, w >$, para a entrada de $\overline{EQ_{TM}}$, $< M1, M2 >$

Teorema 30: EQ_{TM} não é nem Turing-reconhecível nem co-Turing-reconhecível

Prova: por construção usando redutibilidade por mapeamento

- A MT Mf computa a função de mapeamento

Redutibilidade

Teorema 30: EQ_{TM} não é nem Turing-reconhecível nem co-Turing-reconhecível

Prova: por construção usando redutibilidade por mapeamento

- A MT *Mf* computa a função de mapeamento

```
Mf com a entrada <M,w>, onde M é uma MT e w é uma palavra, faz:
  1: Construa as MT M1 e M2
    "M1 = com a entrada x, faz:
       1: Rejeite (M1 rejeita x)"
    "M2 = com a entrada x, faz:
       1: Rode M com a entrada w
       2: Se M aceita w, aceite (M2 aceita x)"
  2: Dê como saída <M1,M2>
```

Redutibilidade

Teorema 30: EQ_{TM} não é nem Turing-reconhecível nem co-Turing-reconhecível

Prova: por construção usando redutibilidade por mapeamento

- A MT *Mf* computa a função de mapeamento

```
Mf com a entrada <M,w>, onde M é uma MT e w é uma palavra, faz:
  1: Construa as MT M1 e M2
    "M1 = com a entrada x, faz:
       1: Rejeite (M1 rejeita x)"
    "M2 = com a entrada x, faz:
       1: Rode M com a entrada w
       2: Se M aceita w, aceite (M2 aceita x)"
  2: Dê como saída <M1,M2>
```

- $M1$ não aceita nada e se M aceita w, $M2$ aceita tudo e então as duas MTs não são equivalentes

Redutibilidade

Teorema 30: EQ_{TM} não é nem Turing-reconhecível nem co-Turing-reconhecível

Prova: por construção usando redutibilidade por mapeamento

- A MT *Mf* computa a função de mapeamento

```
Mf com a entrada <M,w>, onde M é uma MT e w é uma palavra, faz:
  1: Construa as MT M1 e M2
    "M1 = com a entrada x, faz:
       1: Rejeite (M1 rejeita x)"
    "M2 = com a entrada x, faz:
       1: Rode M com a entrada w
       2: Se M aceita w, aceite (M2 aceita x)"
  2: Dê como saída <M1,M2>
```

- $M1$ não aceita nada e se M aceita w, $M2$ aceita tudo e então as duas MTs não são equivalentes
- Se M não aceita w, $M2$ não aceita nada e então as duas são equivalentes

Teorema 30: EQ_{TM} não é nem Turing-reconhecível nem co-Turing-reconhecível

Prova: por construção usando redutibilidade por mapeamento

Teorema 30: EQ_{TM} não é nem Turing-reconhecível nem co-Turing-reconhecível

Prova: por construção usando redutibilidade por mapeamento

- Quando as duas MTs não são equivalentes temos que $< M, w > \in A_{TM}$, pois $< M1, M2 > \in \overline{EQ_{TM}}$

Teorema 30: EQ_{TM} não é nem Turing-reconhecível nem co-Turing-reconhecível

Prova: por construção usando redutibilidade por mapeamento

- Quando as duas MTs não são equivalentes temos que $< M, w > \in A_{TM}$, pois $< M1, M2 > \in \overline{EQ_{TM}}$
- Mostramos que $\overline{EQ_{TM}}$ é indecidível, então EQ_{TM} não é Turing-reconhecível visto que se $A_{TM} \leq_m \overline{EQ_{TM}}$, por definição de redutibilidade por mapeamento, temos que $\overline{A_{TM}} \leq_m EQ_{TM}$ e $\overline{A_{TM}}$ não é Turing-reconhecível

Teorema 30: EQ_{TM} não é nem Turing-reconhecível nem co-Turing-reconhecível

Prova: por construção usando redutibilidade por mapeamento

Teorema 30: EQ_{TM} não é nem Turing-reconhecível nem co-Turing-reconhecível

Prova: por construção usando redutibilidade por mapeamento

- Agora, mostramos que $\overline{EQ_{TM}}$ não é Turing-reconhecível

Teorema 30: EQ_{TM} não é nem Turing-reconhecível nem co-Turing-reconhecível

Prova: por construção usando redutibilidade por mapeamento

- Agora, mostramos que $\overline{EQ_{TM}}$ não é Turing-reconhecível
- Mostramos isso por meio de redução de A_{TM} para EQ_{TM} (complemento de $\overline{EQ_{TM}}$)

Redutibilidade

Teorema 30: EQ_{TM} não é nem Turing-reconhecível nem co-Turing-reconhecível

Prova: por construção usando redutibilidade por mapeamento

- Agora, mostramos que $\overline{EQ_{TM}}$ não é Turing-reconhecível
- Mostramos isso por meio de redução de A_{TM} para EQ_{TM} (complemento de $\overline{EQ_{TM}}$)
- Precisamos de uma função de mapeamento que converta a entrada de A_{TM}, $< M, w >$, para a entrada de EQ_{TM}, $< M1, M2 >$

Redutibilidade

Teorema 30: EQ_{TM} não é nem Turing-reconhecível nem co-Turing-reconhecível

Prova: por construção usando redutibilidade por mapeamento

- A MT Mf computa a função de mapeamento

Redutibilidade

Teorema 30: EQ_{TM} não é nem Turing-reconhecível nem co-Turing-reconhecível

Prova: por construção usando redutibilidade por mapeamento

- A MT *Mf* computa a função de mapeamento

```
Mf com a entrada <M,w>, onde M é uma MT e w é uma palavra, faz:
  1: Construa as MT M1 e M2
    "M1 = com a entrada x, faz:
       1: Aceite (M1 aceita x)"
    "M2 = com a entrada x, faz:
       1: Rode M com a entrada w
       2: Se M aceita w, aceite (M2 aceita x)"
  2: Dê como saída <M1,M2>
```

Teorema 30: EQ_{TM} não é nem Turing-reconhecível nem co-Turing-reconhecível

Prova: por construção usando redutibilidade por mapeamento

- A MT *Mf* computa a função de mapeamento

```
Mf com a entrada <M,w>, onde M é uma MT e w é uma palavra, faz:
  1: Construa as MT M1 e M2
    "M1 = com a entrada x, faz:
       1: Aceite (M1 aceita x)"
    "M2 = com a entrada x, faz:
       1: Rode M com a entrada w
       2: Se M aceita w, aceite (M2 aceita x)"
  2: Dê como saída <M1,M2>
```

- $M1$ aceita tudo e se M aceita w, $M2$ aceita tudo e então as duas MTs são equivalentes

Redutibilidade

Teorema 30: EQ_{TM} não é nem Turing-reconhecível nem co-Turing-reconhecível

Prova: por construção usando redutibilidade por mapeamento

- A MT *Mf* computa a função de mapeamento

```
Mf com a entrada <M,w>, onde M é uma MT e w é uma palavra, faz:
  1: Construa as MT M1 e M2
    "M1 = com a entrada x, faz:
       1: Aceite (M1 aceita x)"
    "M2 = com a entrada x, faz:
       1: Rode M com a entrada w
       2: Se M aceita w, aceite (M2 aceita x)"
  2: Dê como saída <M1,M2>
```

- $M1$ aceita tudo e se M aceita w, $M2$ aceita tudo e então as duas MTs são equivalentes
- Se M não aceita w, $M2$ não aceita nada e então as duas não são equivalentes

Teorema 30: EQ_{TM} não é nem Turing-reconhecível nem co-Turing-reconhecível

Prova: por construção usando redutibilidade por mapeamento

Teorema 30: EQ_{TM} não é nem Turing-reconhecível nem co-Turing-reconhecível

Prova: por construção usando redutibilidade por mapeamento

- Quando as duas MTs são equivalentes temos que $< M, w > \in A_{TM}$, pois $< M1, M2 > \in EQ_{TM}$

Redutibilidade

Teorema 30: EQ_{TM} não é nem Turing-reconhecível nem co-Turing-reconhecível

Prova: por construção usando redutibilidade por mapeamento

- Quando as duas MTs são equivalentes temos que $< M, w >\in A_{TM}$, pois $< M1, M2 >\in EQ_{TM}$
- Mostramos que EQ_{TM} é indecidível, então $\overline{EQ_{TM}}$ não é Turing-reconhecível visto que se $A_{TM} \leq_m EQ_{TM}$, por definição de redutibilidade por mapeamento, temos que $\overline{A_{TM}} \leq_m \overline{EQ_{TM}}$ e $\overline{A_{TM}}$ não é Turing-reconhecível

Conclusão

- Linguagens Turing-reconhecíveis
- Provas através de redução

Material de apoio

- Livro Sipser, Capítulo 5
- Livro Hopcroft, Capítulo 9