Teoria da Computação

Reduções e Problemas Indecidíveis

Thiago Alves

Introdução

- IND = {“G”#“k” | G tem um conjunto independente de tamanho k}
- Um conjunto independente em um grafo é um conjunto de vértices que não tem ligações entre si

Introdução

- CLIQUE = {“G”#“k” | G tem uma clique de tamanho k}
- Uma clique de um grafo é um subgrafo em que todos os vértices são ligados entre si

Introdução

- Imagine que já temos um algoritmo para decidir CLIQUE
- Será que podemos usar esse algoritmo para decidir o IND?

Introdução

- Vamos construir uma função tal que “G”#“k” ∈ IND sse f(“G”#“k”) ∈ CLIQUE
  - Transforma uma entrada do IND em uma entrada do CLIQUE
  - Se “G”#“k” ∈ IND então f(“G”#“k”) ∈ CLIQUE
  - Se “G”#“k” ∉ IND então f(“G”#“k”) ∉ CLIQUE

Introdução

- Nossa função f vai ser a função que retorna o complemento de um grafo de entrada
  - Se G tem um conjunto independente de tamanho k então o complemento de G tem clique de tamanho k
  - Se G não tem conjunto independente de tamanho k então o complemento de G não tem clique de tamanho k

Introdução

- Como usar a função que retorna o complemento de um grafo e um algoritmo que decide o CLIQUE para decidir o IND?

```
temIndSet(g, k)
   g2,k2 = complement(g,k)
   return temClique(g2,k2)
```

Definição

- Uma redução de L_1 para L_2 é uma função computável f tal que
  - $x \in L_1$ se e somente se $f(x) \in L_2$
- Com a redução f e um algoritmo que decide L_2 conseguimos decidir L_1
- Dizemos que L_1 se reduz a L_2
- $L_1 \leq_m L_2$

Redução

- Se $L_1 \leq_m L_2$ e L_2 é decidível
  - O que podemos dizer de L_1?

Redução

- Se $L_1 \leq_m L_2$ e L_2 é decidível
  - O que podemos dizer de L_1?
  - L_1 também é decidível!

Reduções

- Teorema: Se L_2 é decidível e $L_1 \leq_m L_2$ então L_1 é decidível
- Prova:
  - Seja M_2 a máquina que decide L_2 e M_r a máquina da redução
  - Podemos construir M_1 que executa M_r com a entrada e depois executa M_2
  - M_1 decide L_1

Reduções

- Vamos utilizar reduções para mostrar outros problemas indecidíveis
- Forma mais fácil de mostrar que um problema não é decidível

Problemas Indecidíveis

- L_ε = {“M” | M pára com ε} não é decidível

Problemas Indecidíveis

- Prova:
  - Suponha L_ε decidível
  - Vamos mostrar que $H \leq_m L_\varepsilon$
  - Seja “M”#“w” uma entrada de H
  - Podemos construir M' que escreve w na fita se na configuração inicial tem B
  - Nesse caso, depois M' simula M

Problemas Indecidíveis

- Entendendo a redução $H \leq_m L_\varepsilon$:

```
reduction(M, w)
  return M'(x)
            if x == ε
              return M(w)
            return False
```

Problemas Indecidíveis

- Entendendo a redução $H \leq_m L_\varepsilon$:
  - Se M' pára com ε então M pára com w
  - Se M' não pára com ε então M não pára com w
- “M”#“w” $\in$ H sse “M'” $\in L_\varepsilon$
- “M”#“w” $\in$ H sse f(“M”#“w”) $\in L_\varepsilon$
- Logo, $H \leq_m L_\varepsilon$

Problemas Indecidíveis

- Prova:
  - Fizemos a suposição de que L_ε é decidível. Logo, H também é decidível.
  - Absurdo! Logo, L_ε não é decidível.

Problemas Indecidíveis

- S = {“M” | L(M) ≠ ∅} não é decidível

Problemas Indecidíveis

- Prova:
  - Suponha S decidível
  - Vamos mostrar que $A \leq_m S$
  - Seja “M”#“w” uma entrada de A
  - Vamos construir M' que verifica se sua entrada x é igual a w
  - Se não for, rejeita
  - Se for igual, simular M com w e aceitar se e somente se M aceitar w

Problemas Indecidíveis

- Entendendo a redução $A \leq_m S$:

```
reduction(M, w):
  return M'(x):
           if x == w:
             return M(w)
           return False
```

- Se M aceita w então M' só aceita w
- Se M não aceita w então M' não aceita nenhuma string

Problemas Indecidíveis

- Prova:
  - M aceita w sse M' aceita alguma entrada
  - “M”#“w” $\in$ A sse L(M') $\neq \emptyset$
  - “M”#“w” $\in$ A sse “M'” $\in$ S
  - “M”#“w” $\in$ A sse f(“M”#“w”) $\in$ S
  - Logo, A $\leq_m$ S

Problemas Indecidíveis

- Prova:
  - Fizemos a suposição de que S é decidível. Logo, A também é decidível.
  - Absurdo! Logo, S não é decidível.

Teorema

- Seja duas linguagens L_1 e L_2.
  Se $L_1 \leq_m L_2$ e L_1 não é decidível então L_2 não é decidível.

Teorema

- Seja duas linguagens L_1 e L_2.
  Se $L_1 \leq_m L_2$ e L_1 não é decidível então L_2 não é decidível.
- Prova:
  - Suponha L_2 decidível.
  - Logo, L_1 é decidível.
  - Absurdo! Logo, L_2 não é decidível.

Teorema

- Seja duas linguagens L_1 e L_2.
  Se $L_1 \leq_m L_2$ e L_1 não é r.e. então L_2 não é r.e.

Teorema

- Prova:
  - Suponha L_2 r.e.
  - Seja M_2 a máquina que reconhece L_2 e M_r a máquina da redução
  - Podemos construir M_1 que executa M_r com a entrada e depois executa M_2
  - M_1 reconhece L_1
  - Logo, L_1 é r.e. Absurdo! Portanto, L_2 não é r.e.