Linguagens Formais e Autômatos

Aula 24 - O problema da correspondência de Post

Prof. Dr. Daniel Lucrédio
Departamento de Computação / UFSCar
Última revisão: ago/2015

Referências bibliográficas

- **Introdução à teoria dos autômatos, linguagens e computação / John E. Hopcroft, Rajeev Motwani, Jeffrey D. Ullman** ; tradução da 2.ed. original de Vandenberg D. de Souza. - Rio de Janeiro : Elsevier, 2002 (Tradução de: Introduction to automata theory, languages, and computation - ISBN 85-352-1072-5)
  - Capítulo 9 - Seção 9.4
- **Introdução à teoria da computação / Michael Sipser** ; tradução técnica Ruy José Guerra Barretto de Queiroz ; revisão técnica Newton José Vieira. -- São Paulo : Thomson Learning, 2007 (Título original : Introduction to the theory of computation. "Tradução da segunda edição norte-americana" - ISBN 978-85-221-0499-4)
  - Capítulo 5 - Seção 5.2

Indecidibilidade

- É o estudo sobre o que os computadores podem e o que não podem fazer
- Trata-se de uma limitação conceitual
- Existem problemas que não são “computáveis”
  - Ou: existem problemas para os quais não existe um algoritmo
  - Ou: existem linguagens para as quais não existem decisores (Máquinas de Turing que sempre param)

Indecidibilidade

- Para que estudar indecidibilidade?
  - 1. Se você se depara com um problema insolúvel, não há alternativa
    - Precisa ser simplificado ou alterado
  - 2. Ajuda a ganhar perspectiva sobre a computação
    - Sabendo o que é insolúvel, você conhece os limites do que pode e não pode fazer
    - Ajuda no projeto de soluções algorítmicas

Um problema insolúvel

- Problema da Correspondência de Post (PCP)
- Uma instância do PCP é:
  - Duas listas de strings, com o mesmo tamanho k:
    - A = w1, w2, ... , wk
    - B = x1, x2, ... , xk
  - Para cada i, o par (wi,xi) é:
    - Um par correspondente
    - Uma correspondência
- Uma solução para essa instância do PCP é:
  - Uma sequência de um ou mais inteiros
    - S = i1, i2, ... , im
  - Que quando interpretados como índices nas listas A e B
    - a concatenação das strings apontadas por S em A é igual à concatenação das strings apontadas por S em B

PCP

- Exemplo:
- Considere a instância do PCP à direita:
- Essa instância tem solução:
  - S = (2,1,1,3)
    - i1 = 2, i2 = 1, i3 = 1, i4 = 3
  - Pois:
    - w2w1w1w3 = 10111 1 1 10 = 101111110
    - x2x1x1x3 = 10 111 111 0 = 101111110
- Outra solução:
  - S = (2,1,1,3,2,1,1,3)

	Lista A	Lista B
i	wi	xi
1	1	111
2	10111	10
3	10	0

PCP

- Outro exemplo:
- Considere a instância do PCP à direita:
- Essa instância NÃO tem solução!
- É simples demonstrar:
- Uma solução S = (i1,i2,i3,...), certo?
- i1 = 2 e i1 = 3 é impossível, portanto i1 = 1!
- Então S = (1,i2,i3,...), certo?
- Então
  - A = 10....
  - B = 101...
  - Certo?

	Lista A	Lista B
i	wi	xi
1	10	101
2	011	11
3	101	011

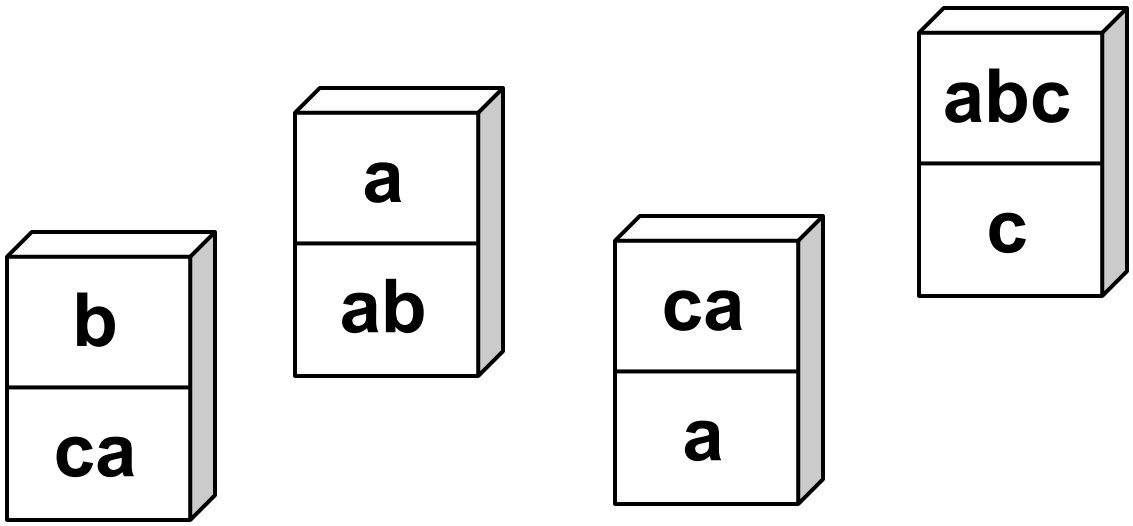
PCP

- Continuando, e i2, o que poderia ser?
- i2 = 1 e i2 = 2 é impossível
  - Portanto i2 = 3!
- Então temos:
  - S = (1,3,i3,...)
  - A = 10101...
  - B = 101011...
- Nesse ponto, i3 = 1 e i3 = 2 é impossível
  - Portanto i3 = 3
  - S = (1,3,3,...)
  - A = 10101101...
  - B = 101011011...
- Da mesma forma, i4=3,i5=3,i6=3, etc...
  - Nunca vai parar! Ou seja, nunca haverá uma correspondência!

	Lista A	Lista B
i	wi	xi
1	10	101
2	011	11
3	101	011

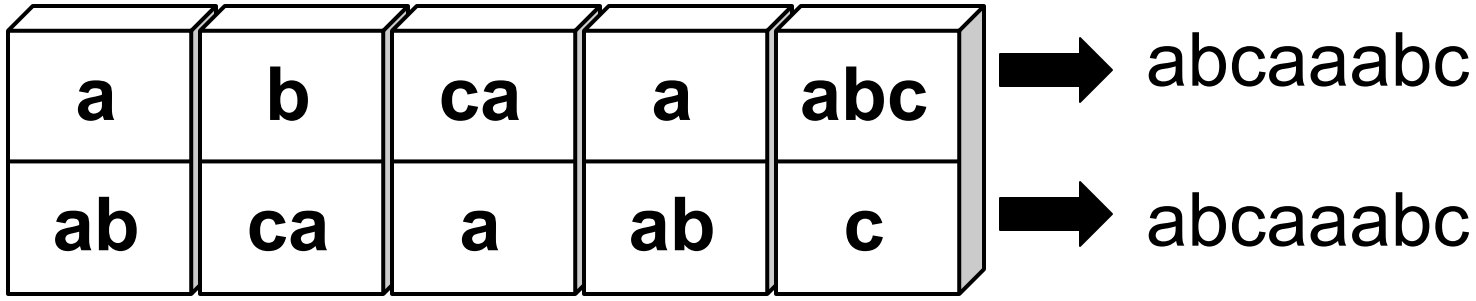
PCP

- Outra forma de visualizar o PCP é imaginando um conjunto de peças de dominó, com strings de letras ao invés de números:



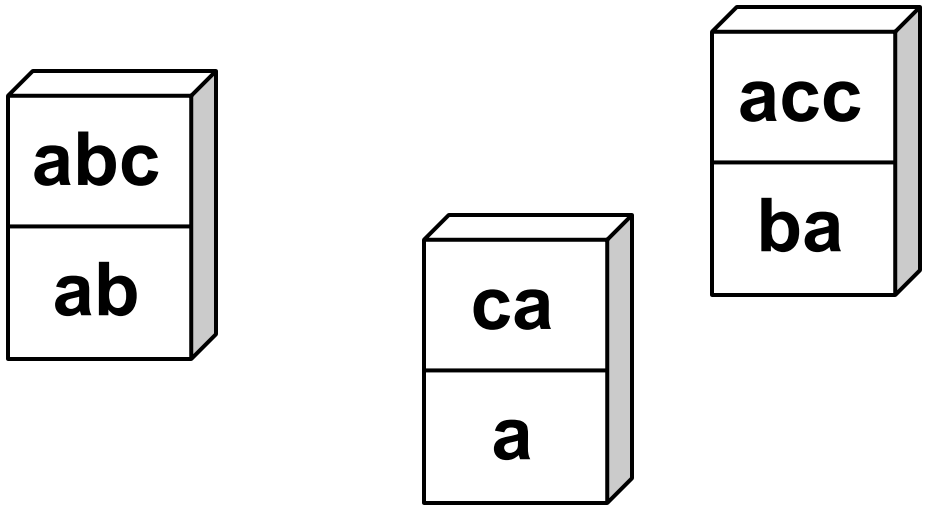
PCP

- O objetivo é fazer uma lista de peças
  - Sem girá-las
  - Com repetições permitidas
  - Não precisa usar todas
- De forma que, lendo-se a linha de cima, tem-se a mesma string que lendo-se a linha de baixo



PCP

- Para alguns conjuntos de peças, existe uma solução (exemplo anterior)
- Para outros (veja abaixo), não existe



PCP

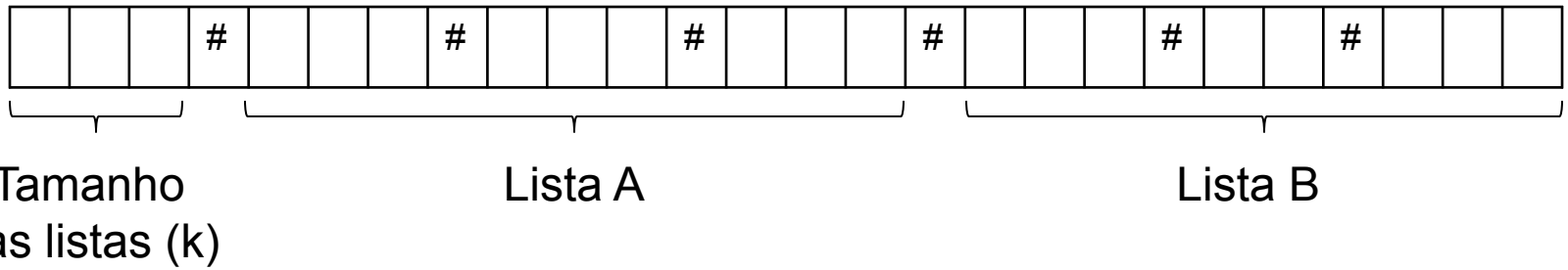
- O problema da correspondência de Post é:

Dada uma instância do PCP, diga se essa instância tem uma solução

- Esse problema é insolúvel
  - Não existe um algoritmo que consiga resolvê-lo

Outra maneira de vermos o PCP

- Suponha uma versão binária do PCP (como a que vimos anteriormente)
  - Ou seja, as strings somente possuem 0s e 1s
- Uma linguagem que descreve instâncias do PCP poderia ser:
  - Linguagem LPCP é uma linguagem sobre Σ = {0,1,#}
  - Onde as cadeias representam instâncias do PCP, no seguinte formato:



- E as cadeias representam instâncias do PCP que possuem solução

PCP como uma linguagem

- Exemplos

Cadeia c1

1	1	#	1	#	1	0	1	1	1	#	1	0	#	1	1	1	#	1	0	#	0

	Lista A	Lista B
i	wi	xi
1	1	111
2	10111	10
3	10	0

	Lista A	Lista B
i	wi	xi
1	10	101
2	011	11
3	101	011

Cadeia c2

1	1	#	1	0	#	0	1	1	#	1	0	1	#	1	0	1	#	1	1	#	0	1	1

PCP como uma linguagem

- Nos exemplos anteriores
  - c1 pertence à linguagem
  - c2 não pertence à linguagem
  - nenhuma cadeia que não está no formato correto pertence à linguagem
- Dessa forma, o PCP pode ser visto como um problema de pertinência em uma linguagem
  - Que é a nossa definição de problema!
- Ou seja:
  - Dada uma cadeia c, determinar se ela pertence ou não à linguagem LPCP

Outra forma de vermos a insolubilidade do PCP

- Veremos que não existe um algoritmo para o PCP (por enquanto, acredite que não existe)
- Na terminologia formal, isto significa que:
  - Não existe um decisor para a linguagem LPCP
  - É impossível projetar uma Máquina de Turing que sempre para (aceitando ou rejeitando) para a linguagem LPCP
  - É impossível construir um programa em C, Java, C#, Pascal, Ruby on Rails, Groovy, ou qualquer outra linguagem de programação, que resolve este problema!

Indecidibilidade

- Veremos:
  - Como provar que o PCP é insolúvel
    - Usaremos – obviamente – Máquinas de Turing e conceitos de linguagem para isso
    - Mas poderíamos fazer o mesmo com a noção de algoritmos!
- Veremos que existem duas formas de indecidibilidade
  - Existem problemas para os quais é impossível projetar uma MT
    - Linguagens não-recursivamente enumeráveis
  - Existem problemas para os quais é possível projetar uma MT, mas ela pode entrar em loop
    - Ou seja, não é um decisor
    - Linguagens não-recursivas
- Na prática, dá no mesmo, pois em ambos os casos não existe um algoritmo

Fim

Aula 24 - O problema da correspondência de Post