UDESC

Teoria da Computação

Prof. Diego Buchinger
diego.buchinger@outlook.com
diego.buchinger@udesc.br

SIPSER, Michael.
Introdução à Teoria da Computação

Revisitando a
Hierarquia de Chomsky

Hierarquia de Chomsky

Existe alguma linguagem que não seja Recursivamente Enumerável, isto é, esteja fora deste conjunto?

Linguagens Turing-reconhecíveis

Corolário: Algumas linguagens não são Turing-reconhecíveis. Elas são Turing-irreconhecíveis.

Prova:

1. Mostrar que o conjunto de todas as MTs é contável
2. Mostrar que o conjunto de todas as linguagens é incontável

Linguagens Turing-reconhecíveis

P1: Mostrar que o conjunto de todas as MTs é contável

- Σ^* o conjunto de todas as cadeias de <u>tamanho finito</u> sobre o alfabeto Σ e é contável:

{cadeia de comprimento 0, cadeias de comprimento 1, cadeias de comprimento 2, ... , cadeias de comprimento *n*}

Lembrete: um algoritmo ou MT possuem um número finito de passos.

Como uma MT M pode ser representada por uma codificação usando Σ^*, então a quantidade de MTs também é contável.

(ainda desconsiderando codificações inválidas de MTs – ex: MT sempre começam com 0)

Linguagens Turing-reconhecíveis

P2: Mostrar que o conjunto de todas as linguagens é incontável

- Considere o conjunto de todas as sequências binárias infinitas, que chamaremos de B;
- Provamos que B é incontável utilizando o método da diagonalização; [prove!]
- Considere também $\mathcal{L}$ como o conjunto de todas as linguagens sobre o alfabeto Σ.
- Mostramos que $\mathcal{L}$ é incontável apresentando uma correspondência com B (=os dois são do mesmo tamanho)
  - Mostrar que cada linguagem $A \in L$ tem uma sequência única em B. Criamos essa correspondência com $f\colon \mathcal{L} \rightarrow$ B

Linguagens Turing-reconhecíveis

Assim, como o número de linguagens é incontável e o número de MT é contável, podemos afirmar que existem linguagens sem uma MT correspondente.

Em outras palavras, algumas linguagens não são Turing Reconhecíveis.

Linguagens Decidíveis

Teorema: uma linguagem é decidível se, e somente se, ela é Turing-Reconhecível e co-Turing-Reconhecível.

O que é uma linguagem co-Turing-Reconhecível?

Prova: mostrar a equivalência nos dois sentidos

1. Mostrar que $\mathrm{A}_{\text{decid}ível} \rightarrow A\ e\ \bar{A}\ são\ Turing\ Reconhecíveis$
2. Mostrar que $A\ e\ \bar{A}\ são\ Turing\ Reconhecíveis \rightarrow \mathrm{A}_{\text{decid}ível}$

Linguagens Decidíveis

P1. Mostrar que $\mathrm{A}_{decidível} \rightarrow A\ e\ \bar{A}\ são\ Turing\ Reconhecíveis$

Fácil percepção!

P2. Mostrar que $A\ e\ \bar{A}\ são\ Turing\ Reconhecíveis \rightarrow \mathrm{A}_{decidível}$

Criar um MT que combina M_1 – um reconhecedor para A e M_2 – um reconhecedor para $\bar{A}$.

Mas M1 ou M2 por serem Turing Reconhecíveis (e não Decidíveis) não podem entrar em loop? Como M vai ser decidível neste caso?

Co-Analizador de MTs

Corolário: $\overline{A_{MT}}$ não é Turing-Reconhecível.

Sabemos que A_{MT} é Turing-Reconhecível

Considerando a prova anterior, se também fosse Turing-Reconhecível, o que poderíamos afirmar sobre A_{MT}?

Redutibilidade

Redução ($\leq$)

Redução: é uma maneira de converter um problema em outro de forma que uma solução para o segundo problema possa ser usada para resolver o primeiro.

Exemplos:

se orientar na cidade C $\leq$ conseguir o mapa da cidade

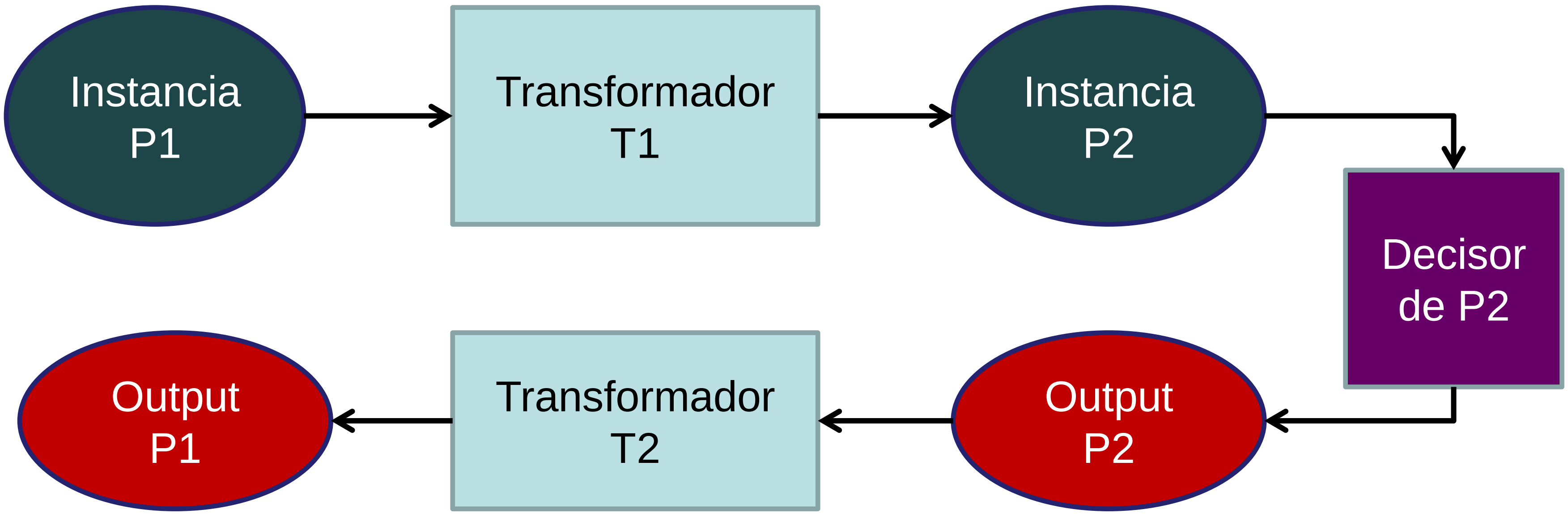
viajar para Las Vegas $\leq$ comprar passagem aérea

comprar passagem aérea $\leq$ ganhar dinheiro

ganhar dinheiro $\leq$ encontrar um trabalho

resolver um sistema linear $\leq$ inverter uma matriz (mat. inversa)

Redução (≤)



Sendo A redutível a B (A ≤ B), o que podemos afirmar sobre a dificuldade sobre A (em relação a B)?

Redução $(\leq)$

Algumas implicações lógicas:

Sendo $A \leq B$ e B decidível
o que podemos afirmar sobre A?

Sendo $A \leq B$ e A indecidível
o que podemos afirmar sobre B?

Problemas Indecidíveis

Já provamos que A_{MT} é indecidível. Agora, podemos utilizá-lo para mostrar a indecidibilidade de outros problemas.

A_{MT} => Aceitador de MTs

Teorema: $PARA_{MT}$ é indecidível

$PARA_{MT} = \{\langle M, w\rangle \mid M \text{ é } uma\ MT\ e\ M\ pára\ sobre\ a\ entrada\ w\}$

$PARA_{MT}$ é o real problema da parada

Prova por contradição: supomos que $PARA_{MT}$ seja decidível (existe uma MT R que a decide) e usamos essa suposição para mostrar que A_{MT} é decidível (contradição com a prova de que A_{MT} é indecidível)

Problemas Indecidíveis

Teorema: V_{MT} é indecidível

$V_{MT} = \{\langle M \rangle \mid M \text{ é } uma\ MT\ e\ L(M) = \emptyset\}$

V_{MT} = problema da vacuidade de MT

Prova por contradição: supomos que V_{MT} é decidível (existe uma MT que decide se outra máquina de Turing M nada reconhece) e usamos essa suposição para mostrar que A_{MT} é decidível (contradição com a prova de que A_{MT} é indecidível)

Problemas Indecidíveis

Teorema: $REGULAR_{MT}$ é indecidível

$REGULAR_{MT} = \{\langle M \rangle \mid M \text{ é } uma\ MT\ e\ L(M) \text{ é } uma\ linguagem\ regular\}$

$REGULAR_{MT}$ = determinar se uma dada MT reconhece uma linguagem que pode ser reconhecida também por um modelo computacional mais simples.

Prova por contradição: supomos que $REGULAR_{MT}$ é decidível (existe uma MT que decide se outra máquina de Turing M reconhece uma linguagem regular) e usamos essa suposição para mostrar que A_{MT} é decidível (contradição com a prova de que A_{MT} é indecidível)

Problemas Indecidíveis

Teorema: EQ_{MT} é indecidível

$EQ_{MT} = \{\langle M_1, M_2\rangle \mid M_1 \text{ e } M_2 \text{ são MTs e } L(M_1) = L(M_2)\}$

EQ_{MT} = determinar se duas MTs são equivalentes, ou seja, se elas reconhecem a mesma linguagem.

Prova por contradição: supomos que EQ_{MT} é decidível (existe uma MT que decide se duas outras MTs M_1 e M_2 reconhecem a mesma linguagem) e usamos essa suposição para mostrar que V_{MT} é decidível (contradição com a prova de que V_{MT} é indecidível)

Histórias de Computação

Definição: a história de computação para uma máquina de Turing M sobre uma entrada é a sequência de configurações (C_1, C_2, ... , C_n) pelas quais a máquina passa à medida que processa uma entrada w, onde C_1 é a configuração inicial e C_n é uma configuração de aceitação ou rejeição.

Constituição de uma Configuração:

- Estado do controle
- Posição do cabeçote
- Conteúdo da fita

Representação de configurações:

q_0bab => $\#q_1ab$ => $\#aq_2b$ => ...

Escreva uma MT det. para decidir a linguagem

$$L_1 = \{w|w \in \{a,b\}^* \; e \; w = xax \; com \; x \in \{b\}^*\}$$

Escreva uma história de computação considerando:

$w_1 = a$

$w_2 = bab$

$w_3 = bbab$

Histórias de Computação

Propriedades:

- Histórias de computação são finitas: se *M* não para sobre *w*, nenhuma história de computação de aceitação ou rejeição existe para este caso;
- MT deterministas possuem no máximo uma história de computação sobre uma dada entrada w;
- MT não-deterministas podem ter muitas histórias de computação sobre uma determinada entrada – correspondente aos vários ramos de computação.

Autômato Linearmente Limitado

Autômato Linearmente Limitado (ALL) é um modelo de MT limitada

Propriedades:

- Cabeça de leitura-escrita não pode se mover para fora da parte da fita contendo a entrada (se a MT tentar fazer esse movimento inválido, a cabeça permanecerá onde está);
- O uso de um alfabeto de fita maior do que o alfabeto da entrada permite que a memória disponível seja incrementada de, no máximo, um fator constante, i.e. a quantidade de memória disponível é linear em *n*.

Pode-se entender o ALL como uma MT com quantidade limitada de memória. Logo, ela só pode resolver problemas que requerem memória que podem caber dentro da fita de entrada.

Autômato Linearmente Limitado

Apesar de parecer limitado, ALLs são bastante poderosos!!

Toda LLC pode ser decidida por um ALL

(Linguagem Livre do Contexto)

Agora considere A_{ALL} como problema de se determinar se um ALL aceita sua entrada ou não.

$$A_{ALL} = \{\langle M, w\rangle \mid M \text{ é } um\ ALL\ que\ aceita\ a\ cadeia\ w\}$$

Este problema é decidível ou indecidível?

Autômato Linearmente Limitado

Teorema: A_{ALL} é decidível

Para provarmos este teorema devemos pensar na quantidade máxima possível de configurações!

Considerando um ALL com q estados, g símbolos no alfabeto de fita e uma entrada w de tamanho n (tamanho da fita ou tamanho de memória), quantas configurações distintas existem?

Autômato Linearmente Limitado

Teorema: A_{ALL} é decidível

Lema: seja M um ALL com q estados, g símbolos no alfabeto de fita e uma memória de tamanho n, existem exatamente:

qng^n configurações distintas

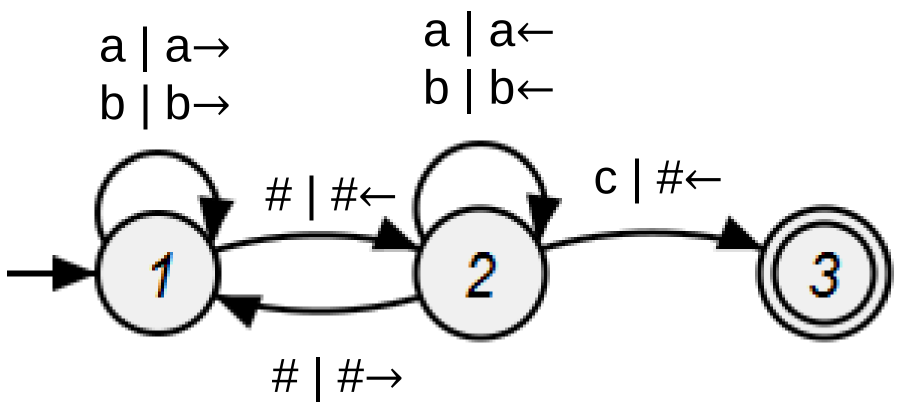
Prova: o algoritmo que decide A_{ALL} é:

L = "Sobre a entrada , onde M é um ALL e w é uma cadeia:

1. Simule M sobre w por qng^n passos ou até que ela pare.
2. Se M parou, *aceite* se ela aceitou e *rejeite* se ela rejeitou. Se ela não parou, *rejeite*.

Autômato Linearmente Limitado

Teorema: A_{ALL} é decidível



$\Sigma = \{a, b, c\}$

denota um marcador de início e final de fita

w1 = aba |03|

w2 = aababaaaabbaba |14|

Problemas Indecidíveis

Teorema: V_{ALL} é indecidível

$V_{ALL} = \{\langle M \rangle \mid M \text{ é } um\ ALL\ onde\ L(M) = \emptyset\}$

V_{ALL} = determinar se existe uma MT M que decide se, dado um ALL qualquer, este não aceita algo.

Prova por contradição: mostrar que se V_{ALL} fosse decidível então A_{MT} também seria (contradição com a prova de que A_{MT} é indecidível).

Problemas Indecidíveis

Teorema: $\text{TODAS}_{\text{GLC}}$ é indecidível

$\text{TODAS}_{\text{GLC}} = \{\langle G\rangle \mid G \text{ é } uma\ GLC\ e\ L(G) = \Sigma^*\}$

$\text{TODAS}_{\text{GLC}}$ = determinar se uma gramática-livre-do-contexto gera todas as cadeias possíveis.

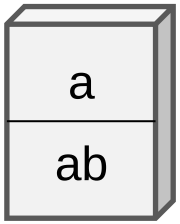
Prova por contradição: supomos que $\text{TODAS}_{\text{GLC}}$ seja decidível e então usamos esta suposição para mostrar que A_{MT} também seria decidível (contradição com a prova de que A_{MT} é indecidível).

UDESC

Um Problema Indecidível Simples
Problema da Correspondência de Post

PCP

- A indecidibilidade não está presente apenas em problemas envolvendo autômatos diretamente
- PCP é um problema de estilo charada envolvendo dominós:
  - ❑ Começamos com uma coleção de tipos de dominós, cada um contendo duas cadeias
  - ❑ Considere que temos infinitas peças de cada dominó
  - ❑ Não é possível usar uma peça virada (de ponta-cabeça)
  - ❑ O objetivo é encontrar uma sequencia de dominós que quando lado a lado formam a mesma cadeia tanto na parte superior como na inferior, i.e. um emparelhamento



$$\equiv \left[\frac{a}{ab}\right]$$

PCP

Formalização:

- Uma instância do PCP é uma coleção P de k dominós:

$$P = \left\{ \left[\frac{t_1}{b_1}\right], \left[\frac{t_2}{b_2}\right], \dots, \left[\frac{t_k}{b_k}\right] \right\}$$

- Um emparelhamento é uma sequência $i_1, i_2, \dots, i_m$ onde: $t_{i_1} t_{i_2} \dots t_{i_m} = b_{i_1} b_{i_2} \dots b_{i_m}$
- O problema é determinar se P possui um emparelhamento

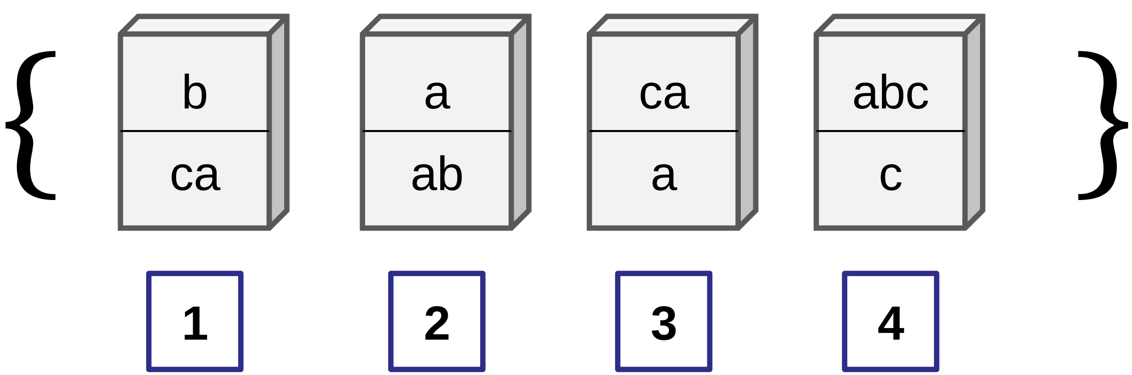
$$PCP = \left\{ \begin{array}{c} \langle P \rangle \mid P \text{ é } uma\ instância\ do\ problema\ da\ correspondência \\ de\ Post\ com\ um\ emparelhamento \end{array} \right\}$$

PCP é um problema indecidível

PCP

- PCP = determinar se uma coleção de dominós possui emparelhamento ou não.
- Exemplo:

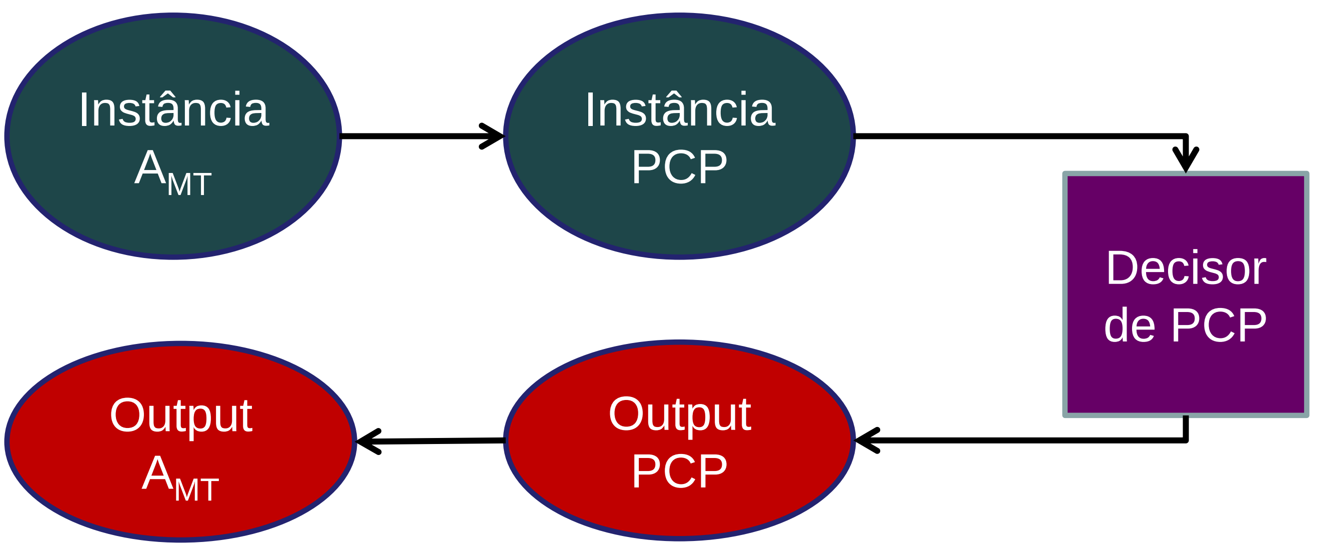
Existe um emparelhamento para a seguinte instância?



PCP é indecidível

Prova: PCP é indecidível

- Ideia:
  - Reduzir a partir de A_{MT} e histórias de computação de aceitação
  - Dados MT *M* e entrada *w*, construímos uma instância *P* onde um emparelhamento é uma história de computação de aceitação para *M* sobre *w*.



PCP é indecidível

Prova: PCP é indecidível

- Três adaptações simplificadoras:
  - Supomos que M nunca tenta mover o cabeçote além da extremidade esquerda sobre a entrada *w*
  - Se $w = \varepsilon$ então usamos _ em seu lugar
  - Exigimos que um emparelhamento comece com o primeiro dominó (PCPM)

$$PCPM = \left\{ \begin{matrix} \langle P \rangle \mid P \text{ é } uma\ instância\ do\ PCP\ com\ um emparelhamento \\ que\ começa\ com\ o\ 1^{\underline{o}}\ dominó \end{matrix} \right\}$$

Relembrando: $M = \{Q, \Sigma, \Gamma, \delta, q_0, q_{aceita}, q_{rejeita}\}$

PCP é indecidível

Prova: PCP é indecidível

- Supomos que a MT ***R*** decide o PCP e construímos ***S*** que decide A_{MT}. ***S*** constrói uma instância do PCP ***P*** que tem um emparelhamento se e somente se ***R*** aceita ***w***.
- Para tanto, ***S*** constrói primeiro uma instância ***P’*** do PCPM

➢ **Passo 1**: adicione o seguinte modelo de dominó à P’, forçando o emparelhamento a começar com este dominó.

$$\left[\frac{\#}{\#q_0w_1w_2\ldots w_n\#}\right]$$

PCP é indecidível

Prova: PCP é indecidível

- **Passo 2**: adicionamos dominós que realizam a parte principal da simulação fazendo com que a próxima simulação de R (um único passo) apareça na cadeia inferior:

  - Para todo $a, b \in \Gamma$ e todo $q, r \in Q$ onde $q \neq q_{rejeita}$ se $\delta(q, a) = (r, b, D)$ adicione o dominó ao lado

$$\left[\frac{qa}{br}\right]$$

  - Para todo $a, b, c \in \Gamma$ e todo $q, r \in Q$ onde $q \neq q_{rejeita}$ se $\delta(q, a) = (r, b, E)$ adicione o dominó ao lado

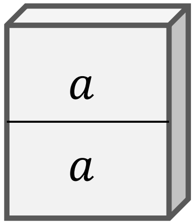
$$\left[\frac{cqa}{rcb}\right]$$

PCP é indecidível

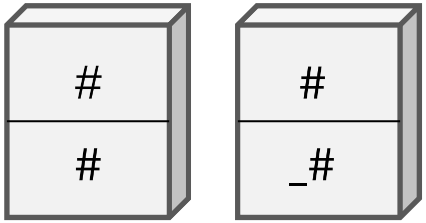
Prova: PCP é indecidível

- **Passo 2**: adicionamos dominós que realizam a parte principal da simulação fazendo com que a próxima simulação de R (um único passo) apareça na cadeia inferior:

  - Para todo $a \in \Gamma$ adicione o dominó ao lado



  - Para permitir a continuação da simulação passo-a-passo, adicione os dominós ao lado



(OBS: o segundo dominó serve para simular a quantidade infinita de brancos a direita que são suprimidos quando escrevemos a configuração)

PCP é indecidível

Prova: PCP é indecidível

- **Passo 3**: quando um estado de aceitação ocorrer, queremos fazer que a parte superior "acompanhe" a parte inferior fechando um emparelhamento completo:

  - ❖ Para todo $a \in \Gamma$ adicione os dominós ao lado (adiciona "pseudopassos" na MT depois que ela parou, "comendo" um a um todos os símbolos adjacentes a cabeça)

    $$\left[\frac{a q_{aceita}}{q_{aceita}}\right] \quad \left[\frac{q_{aceita} a}{q_{aceita}}\right]$$

  - ❖ Adicione o dominó ao lado (permite a finalização da simulação)

    $$\left[\frac{q_{aceita}\#\#}{\#}\right]$$

Chegamos a uma instância de PCPM equivalente

PCP é indecidível

Prova: PCP é indecidível

- **Passo 4**: converter a instância de PCPM em PCP

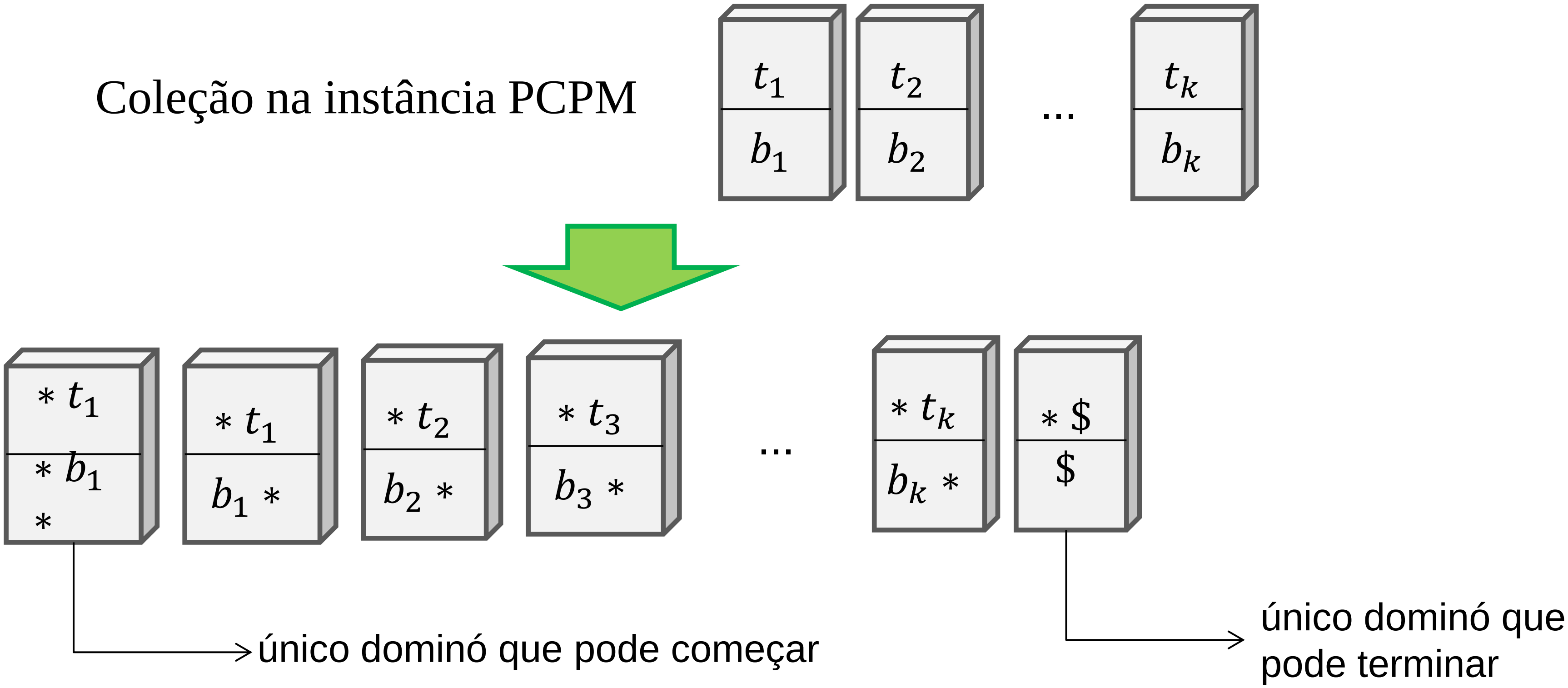
  - Embutir implicitamente a necessidade de se começar com o primeiro dominó adicionando um símbolo especial estrela $(*)$

$$
\begin{aligned}
*u &= *u_1 * u_2 * \ldots * u_n \\
u* &= u_1 * u_2 * \ldots * u_n * \\
u &= *u_1 * u_2 * \ldots * u_n *
\end{aligned}
$$

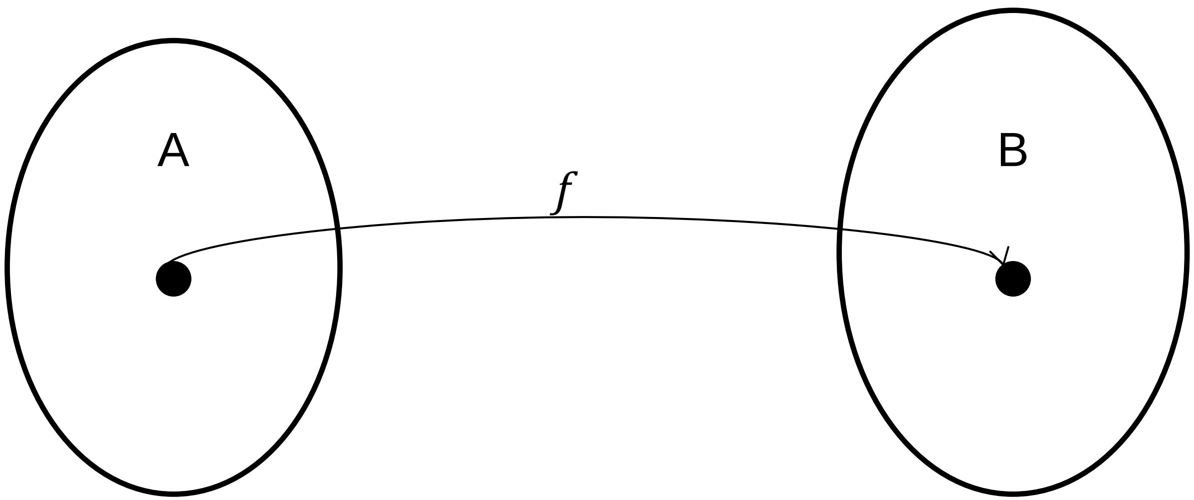
PCP é indecidível

Prova: PCP é indecidível

- **Passo 4**: converter a instância de PCPM em PCP



Redução por Mapeamento ($\leq_m$)



Definição: a linguagem A é <u>redutível por mapeamento</u> à linguagem B (A $\leq_m$ B) se existe uma função computável $f : \Sigma^* \rightarrow \Sigma^*$ onde para todo w $\quad w \in A \Leftrightarrow f(w) \in B$.

A função f é denominada como <u>redução</u> de A para B.

Redução por Mapeamento ($\leq_m$)

- Provê uma forma de converter questões sobre A em questões sobre B, de tal forma que para testar se $w \in A$ usamos a redução f para mapear w para $f(w)$ e testamos se $f(w) \in B$

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

Teorema: Se $A \leq_m B$ e A é indecidível, então B é?