Teoria da Computação
Tese de Church-Turing e Máquinas de Turing

Leonardo Takuno
{leonardo.takuno@gmail.com}

Centro Universitário Senac

Sumário

Sumário

Definição

Definição: Uma linguagem **Turing-reconhecível** é uma linguagem formal para o qual existe um máquina de Turing que pára e aceita dado qualquer cadeia de entrada em uma linguagem mas pode parar e rejeitar ou entrar em loop para qualquer cadeia de entrada que não pertença a linguagem. Em contraste a isso uma linguagem **Turing-decidível**, são aquelas máquinas de Turing que páram em todos os casos.

Sumário

Linguagem Decidível

Considere a linguagem C que define uma aritmética elementar.

$$C = \{a^i b^j c^k | i \times j = k \text{ e } i, j, k \geq 1\}.$$

Construir uma máquina de Turing M3 que a decida.

Linguagem Decidível

M3 = "Na string de entrada w:

1. Varrer a entrada da esquerda para a direita para ter certeza que é um membro de $a^+b^+c^+$ e *rejeite* se não for.
2. Retorne o cabeçote para a extremidade da esquerda da fita.
3. Marque um a, e faça uma varredura para a direita até que um b ocorra. Vá e volte entre os bs e os cs, marcando cada um deles até que todos os bs tenham acabado.
4. Restaure os bs marcados e repita o estágio 3 se há outro a para ser marcado. Se todos os as estão marcados, checar se todos os cs estão marcados. Se estão, então *aceita*; caso contrário, *rejeite*."

Linguagem Decidível

Vamos estudar outro problema conhecido como *problema de distinção do elemento*. Uma lista de strings sobre $\{0,1\}$ é dada, separados por $\#s$ e sua função é de aceitar, se todos os strings são diferentes. A linguagem é

$$E = \{\#x_1\#x_2\#...\#x_l | \text{cada } x_i \in \{0,1\}^* \text{ e } x_i \neq x_j \text{ para cada } i \neq j\}.$$

Uma máquina M4 funciona comparando x_1 com x_2 a x_l, e daí, comparando x_2 com x_3 a x_l, e assim por diante. Segue agora, uma descrição informal da MT M4 decidindo essa linguagem.

Linguagem Decidível

M4 = "Na entrada w:

1. Coloque uma marca sobre o símbolo mais à esquerda da fita. Se esse símbolo não for um #, *rejeite*.
2. Faça uma varredura até o próximo # e coloque uma segunda marca sobre ele. Se nenhum # é encontrado antes de um símbolo vazio, apenas x_1 estava presente, e portanto, *aceite*.
3. Faça um ziguezague, e compare os dois strings à direita dos #*s* marcados. Se eles são iguais, então *rejeite*.

Linguagem Decidível

4. Das duas marcas, mova o mais à direita para o próximo # símbolo à direita. Se nenhum símbolo # é encontrado antes de um símbolo vazio, mova a marca mais à esquerda para o próximo # à sua direita e a marca mais à direita para o # depois desse. Dessa vez, se nenhum # está disponível para a marca mais à direita, todos os strings foram comparados, e portanto, *aceite*.
5. Vá para o estágio 3."

Sumário

Variantes da Máquina de Turing

Como nos autômatos finitos podemos construir diferentes variantes da Máquina de Turing.
E, como no caso dos autômatos finitos, podemos mostrar que todas estas variantes têm o mesmo poder computacional: todos eles reconhecem a mesma linguagem.

Variantes da Máquina de Turing

As variantes mais importantes são:

- Máquinas de Turing Multifita; e
- Máquinas de Turing Não-Determinísticas.

Esta equivalência entre as variantes das MTs, fortalecem a "**robustez**" deste modelo computacional.

Máquinas de Turing Multifita

Uma máquina de Turing Multifita é como uma máquina de Turing comum com várias fitas.

- Cada fita tem sua própria cabeça de leitura e escrita.
- Inicialmente a entrada aparece sobre a fita 1, e as outras iniciam em branco.
- A função de transição é modificada para permitir ler, escrever e mover as cabeças em algumas ou todas as fitas simultaneamente.

Máquinas de Turing Multifita

Formalmente, ela é

$$\delta : Q \times \Gamma^k \rightarrow Q \times \Gamma^k \times \{E, D\}^k$$

onde k é o número de fitas.

$$\delta(q_i, a_1, ..., a_k) = (q_j, b_1, ..., b_k, E, D, ..., E)$$

Máquinas de Turing Multifita

Teorema: Toda máquina de **Turing Multifita** tem uma máquina de Turing de uma única fita que lhe é equivalente.

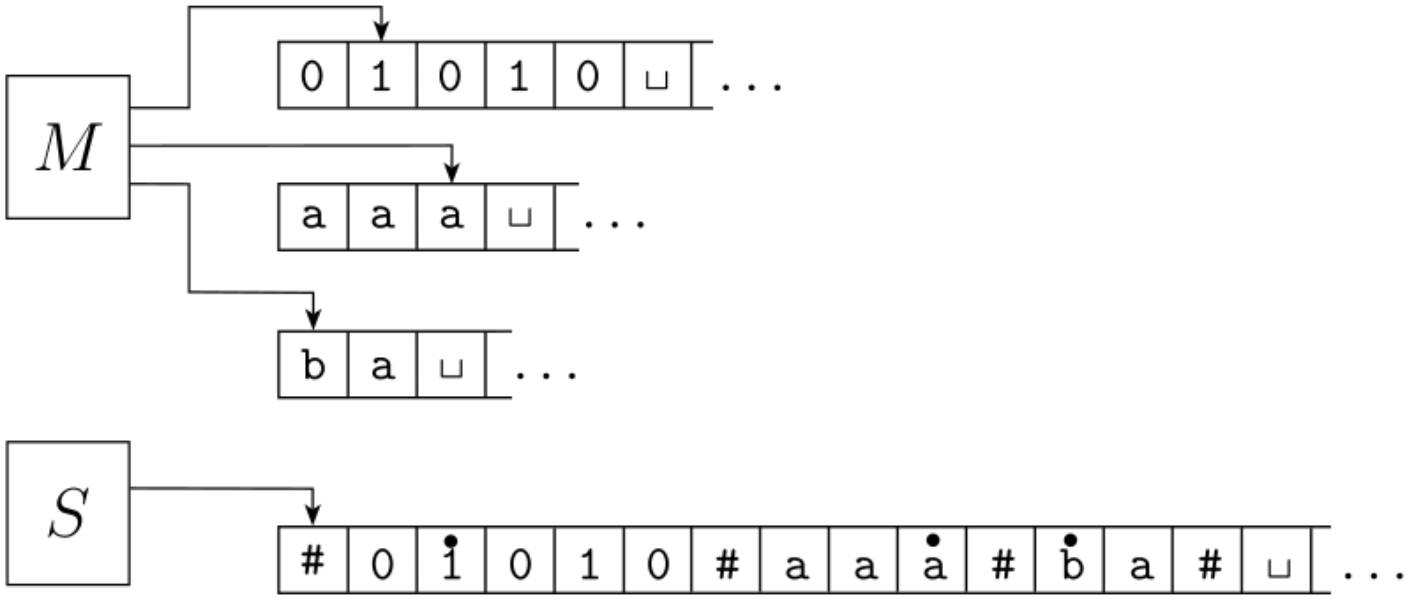
Idéia da prova: Mostramos a equivalência simulando uma MT Multifita M com uma MT S de única fita.

(a) Para mostrar que uma MT Multifita pode simular um MT de única fita é trivial pois uma MT de única fita é um caso especial de um MT Multifita.

(b) Uma MT de fita única pode simular uma MT Multifita por simular k fitas da multifita em uma única fita. Isto requer a separação apropriada dos conteúdos das diferentes fitas. Além do conteúdo dessas fitas, S tem de manter o registro das posições das cabeças.

Máquinas de Turing Multifita

Graficamente :



Simulando Máquinas Multifita

S = "Sobre a entrada $w = w_1 \ldots w_n$:

1. Primeiro S põe sua fita no fomato que representa todas as k fitas de M. a fita formatada contém

$$\# \dot{w}_1 \; w_2 \ldots w_n \# \dot{\sqcup} \# \dot{\sqcup} \# \ldots \#$$

2. Comece no primeiro símbolo #
3. Faça uma varredura até o (k+1)-ésimo # que marca a extremidade direita

Simulando Máquinas Multifita

4. Faça uma segunda passagem para atualizar as fitas conforme a função de transição estabelece
5. Se S move uma das cabeças virtuais sobre um #, essa ação significa que M moveu a cabeça para uma parte previamente não lida em branco daquela fita. Então S desloca o conteúdo da fita, a partir dessa célula até o # mais à direita, uma posição para a direita. Então ela continua a simulação tal qual anteriormente.

Máquinas de Turing Multifita

Corolário: Uma linguagem é **Turing-reconhecível** se, e somente se, alguma máquina de Turing Multifita a reconhece.

Prova: Uma linguagem Turing-reconhecível é reconhecida por uma máquina normal (com uma fita apenas), que é um caso especial de máquinas de Turing Multifita. Isto prova uma direção deste corolário. A outra é dada no teorema anterior.

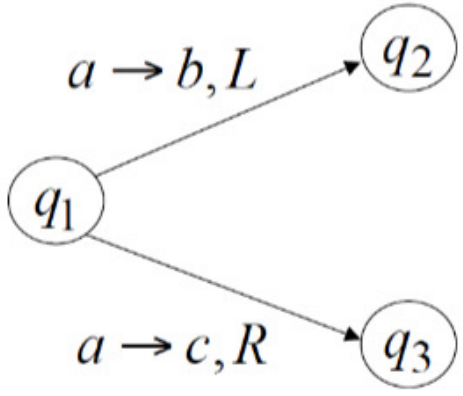
Máquinas de Turing Não-Determinísticas

- Função de transição de uma máquina de Turing não-determinística tem a forma:

$$\delta : Q \times \Gamma \rightarrow \mathcal{P}(Q \times \Gamma \times \{E, D\}).$$

Máquinas de Turing Não-Determinísticas

- Esta definição altera, novamente, a função de transição, que agora passa a ser uma **função parcial**.

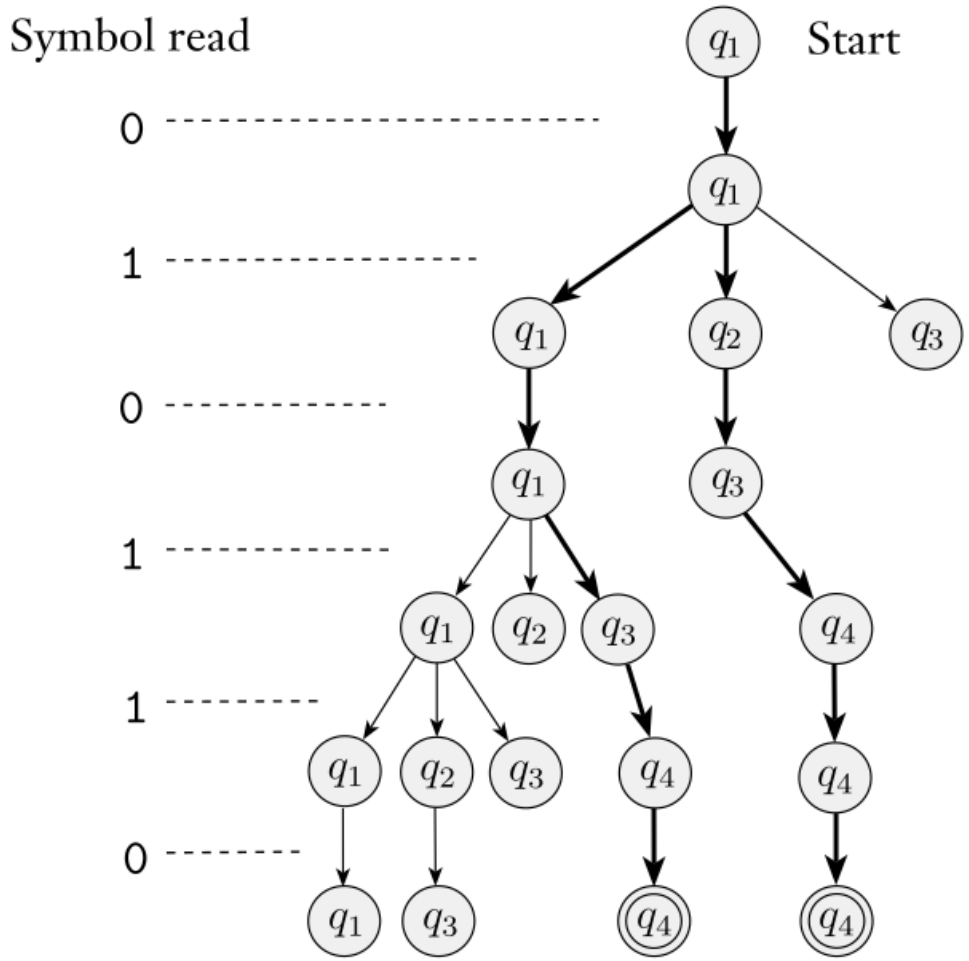


- Isto permite, em particular termos mais de uma transição com o mesmo símbolo do alfabeto a partir de um estado ou mesmo não ter um determinado símbolo do alfabeto na transição.

Máquinas de Turing Não-Determinísticas

- A computação por uma máquina de Turing determinística pode ser representada por uma árvore, cujos ramos correspondem às diferentes possibilidades de computação;
- Se algum desses ramos da computação leva a um estado de aceitação, então a cadeia de entrada é aceita pela máquina.

Máquinas de Turing Não-Determinísticas



Máquinas de Turing Não-Determinísticas

- Árvore de computação não determinística
  - Cada nó representa uma configuração
  - Um nó para uma configuração C_1 tem um filho para cada configuração C_2 tal que C_1 produz C_2
  - A raiz da árvore é $q_1 w$
  - Uma configuração pode aparecer mais de uma vez na árvore.

Máquinas de Turing Não-Determinísticas

Teorema: Toda máquina de **Turing Não-Determinística** tem uma máquina de Turing de uma única fita que lhe é equivalente.

Idéia da prova: Mostramos que uma MT determinística é equivalente a uma MT não-determinística.

(a) Para mostrar que uma MT determinística pode simular uma MT não-determinística é trivial pois MTs determinísticas são casos especiais de MTs não-determinísticas.

Máquinas de Turing Não-Determinísticas

Teorema: Toda máquina de **Turing Não-Determinística** tem uma máquina de Turing de uma única fita que lhe é equivalente.

Idéia da prova:

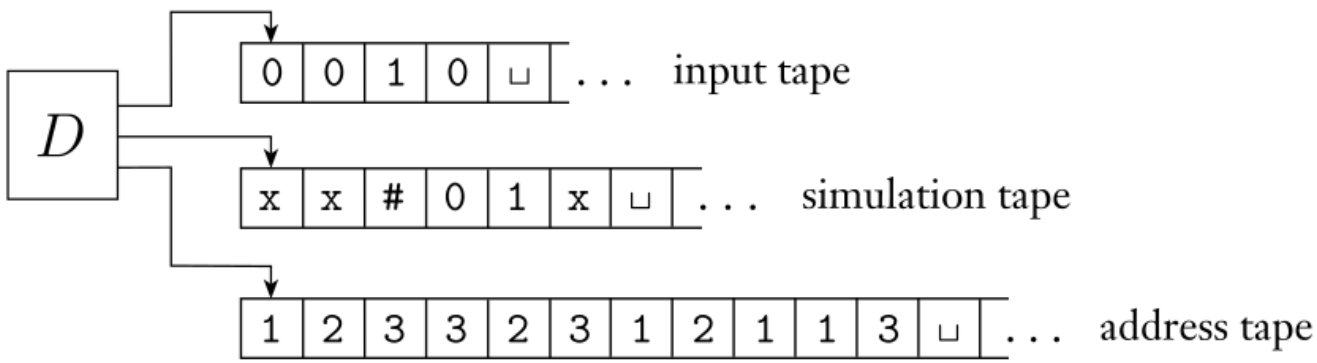
(b) Podemos simular uma MT não-determinística N com uma MT D por ter D fazendo busca através da árvore não-determinística. Cada nó da árvore que representa a computação N é uma configuração de N. D percorre essa árvore, buscando uma configuração de aceitação;

(c) Para que D possa simular N corretamente, essa busca deve ser em largura, e não em profundidade.

Máquinas de Turing Não-Determinísticas

Teorema: Toda máquina de **Turing Não-Determinística** tem uma máquina de Turing de uma única fita que lhe é equivalente.

Prova: A MT determinística simuladora tem três fitas.

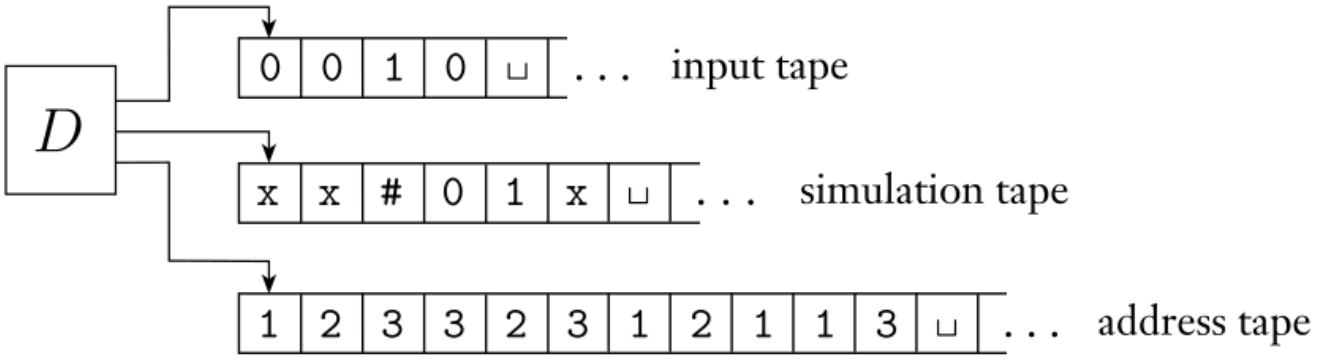


fita1 : contém a cadeia de entrada e nunca é alterada

Máquinas de Turing Não-Determinísticas

Teorema: Toda máquina de **Turing Não-Determinística** tem uma máquina de Turing de uma única fita que lhe é equivalente.

Prova: A MT determinística simuladora tem três fitas.

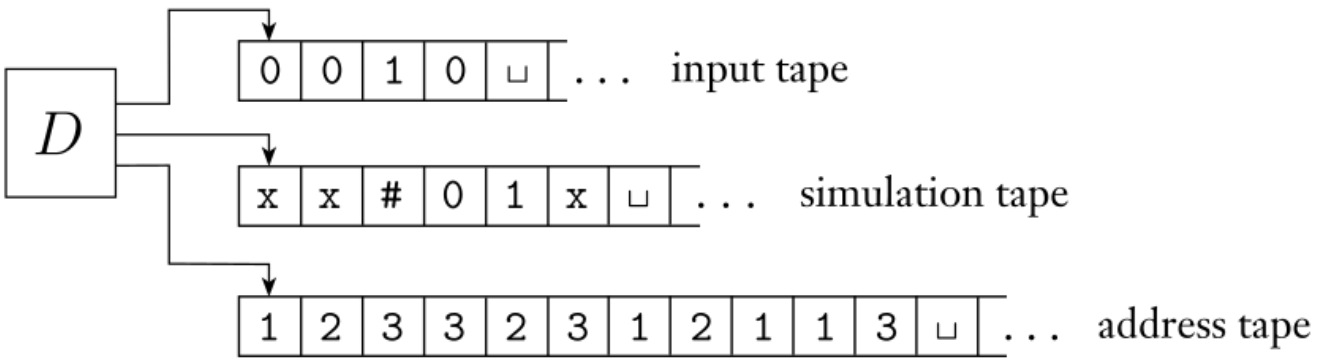


fita2 : mantém a cópia da fita de N em algum ramo de sua computação não determinística.

Máquinas de Turing Não-Determinísticas

Teorema: Toda máquina de **Turing Não-Determinística** tem uma máquina de Turing de uma única fita que lhe é equivalente.

Prova: A MT determinística simuladora tem três fitas.



fita3 : mantém o registro da posição de D na árvore de computação não-determinística de N.

Máquinas de Turing Não-Determinísticas

Teorema: Toda máquina de **Turing Não-Determinística** tem uma máquina de Turing de uma única fita que lhe é equivalente.

Prova: (... continuação)
Primeiro considerar a representação de dados na fita 3. Todo nó da árvore pode ter no máximo b filhos, onde b é o tamanho do maior conjunto de possível escolhas dado pela função de transição de N.
A cada nó associamos um endereço que é uma cadeia sobre $\Sigma_b = \{1, 2, \cdots, b\}$

Máquinas de Turing Não-Determinísticas

Teorema: Toda máquina de **Turing Não-Determinística** tem uma máquina de Turing de uma única fita que lhe é equivalente.

Prova: (... continuação)
Associamos o endereço 231 ao nó ao qual chegamos iniciando na raiz, indo para o seu 2^o filho, indo para o 3^o filho desse nó, e, finalmente, para o 1^o filho desse nó. Cada símbolo na cadeia nos diz que escolha fazer a seguir quando simulamos um passo em um ramo de computção não determinística de N.

Máquinas de Turing Não-Determinísticas

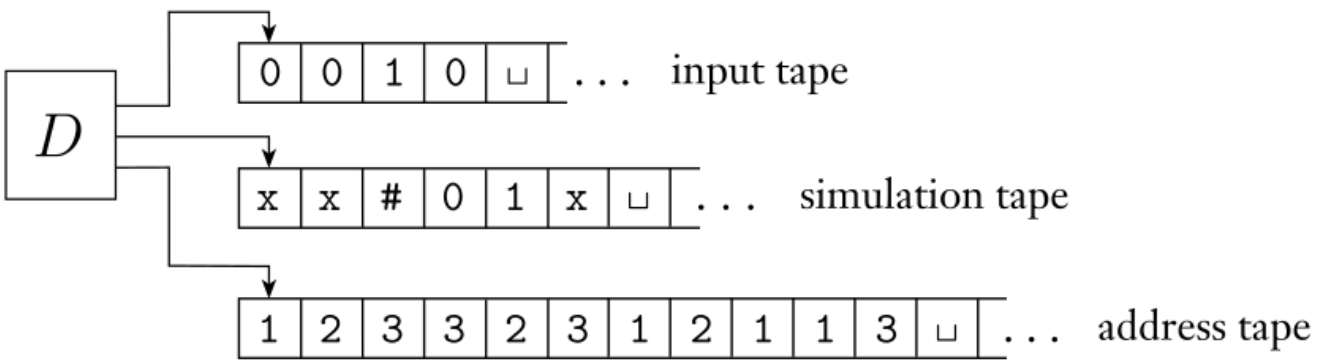
Prova: (... continuação) Agora, vamos descrever D:

1. Inicialmente, a fita 1 contém a entrada w e as fitas 2 e 3 estão vazias.
2. Copie a fita 1 para a fita 2.

Máquinas de Turing Não-Determinísticas

3. Use a fita 2 para simular N com a entrada w sobre um ramo de sua computação não-determinística. Antes de cada passo de N, consulte o próximo símbolo na fita 3 para determinar qual escolha fazer entre aquelas permitidas pela função de transição de N. Se não restam mais símbolos na fita 3 ou se essa escolha não-determinístida for inválida, aborte esse ramo indo para o estágio 4. Também vá para o estágio 4 se uma configuração de rejeição for encontrada. Se uma configuração de aceitação for encontrada aceite a entrada.
4. Substitua a cadeia na fita 3 pela próxima cadeia na ordem lexicográfica. Simule o próximo ramo da computação de N indo para o estágio 2.

Máquinas de Turing Não-Determinísticas



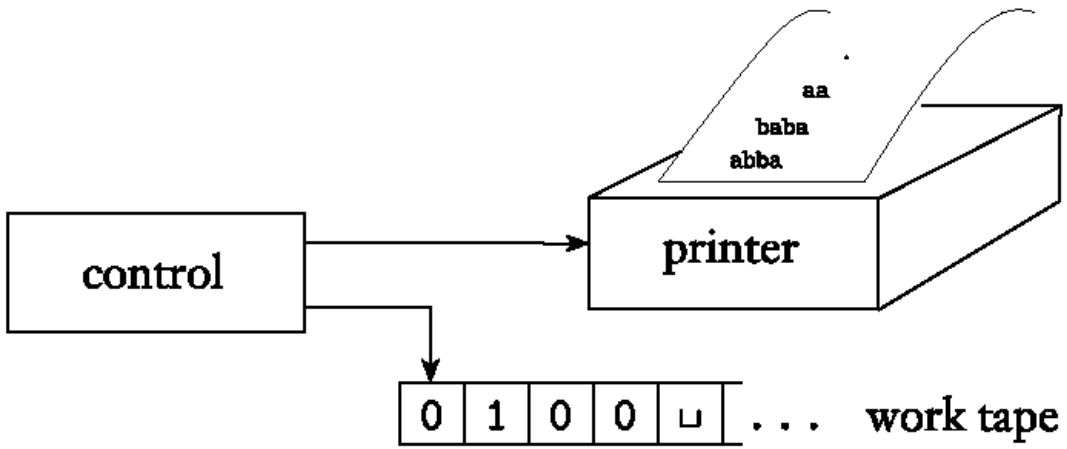
Máquinas de Turing Não-Determinísticas

Corolário: Uma linguagem é Turing-reconhecível se, e somente se, uma MT não-determinística a reconhece.

Corolário: Uma linguagem é decidível se, e somente se, uma MT não-determinística a decide.

Enumerador

- Alguns autores usam o termo **recursivamente enumerável** para linguagens Turing-reconhecíveis;
- Esse termo origina-se de uma variante de Máquina de Turing chamada **enumerador**. Definida de forma livre, esta máquina é uma MT com uma impressora acoplada a ele;



Enumerador

- Um enumerador E começa a computação com fita **vazia** e imprime uma lista de strings, que pode ser infinita, se a computação não parar.
- A linguagem enumerada por E é o conjunto das cadeias que ele imprime na impressora. As cadeias podem ser impressas em qualquer ordem e, possivelmente, com repetições de string, etc.

Enumerador e Enumerabilidade

Teorema: Uma linguagem é Turing-reconhecível se, e somente se, existe um enumerador que a enumera.

Primeiro, mostramos que se um enumerador E enumera uma linguagem A, uma máquina de Turing reconhece A.
A MT M funciona da seguinte forma:
M = " Sobre a entrada w:

1. Execute E. Para cada string impresso por E, compara-o com w ($w \in A$);
2. Aceita w se o string impresso por E for igual a w."

Enumerador e Enumerabilidade

Teorema: Uma linguagem é Turing-reconhecível se, e somente se, existe um enumerador que a enumera.

Agora, provamos a outra direção da prova. Se uma MT M reconhece uma dada linguagem A, podemos construir um enumerador E que gere as palavras de A.
Digamos que s_1,s_2,..., s_n uma enumeração de todos os strings sobre Σ^*.

Enumerador e Enumerabilidade

Teorema: Uma linguagem é Turing-reconhecível se, e somente se, existe um enumerador que a enumera.

$E =$ "Ignore a entrada.

1. Repita este passo $i = 1, 2, ...$;
2. Executa M por i passos em cada entrada $s_1, s_2, ..., s_i$.
3. Se quaisquer computações aceitam, imprima o s_j correspondente."

Se M aceita uma cadeia específica s, eventualmente ela aparecerá na lista gerada por E.

Sumário

Equivalência com outros modelos

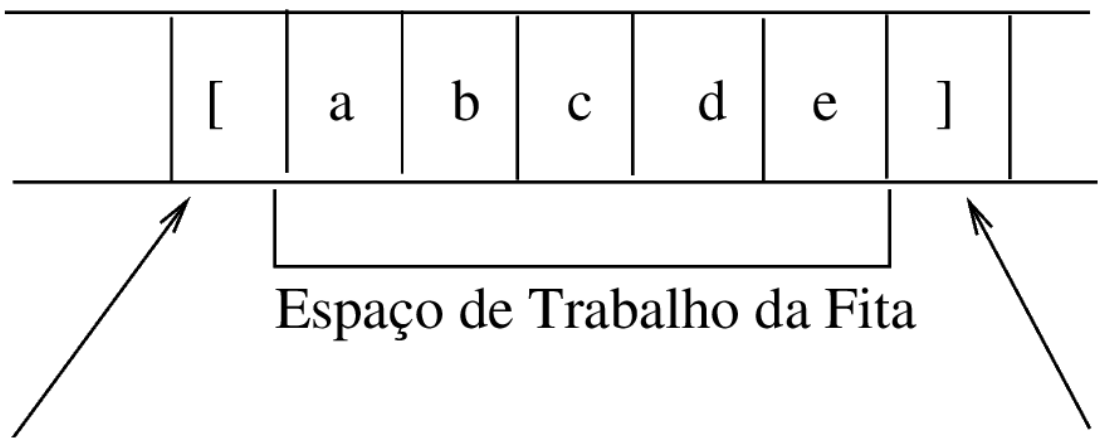
- Há outros modelos de computação de propósito geral, bem como variantes sob a MT original;
- Todos compartilham umas das características essenciais da máquina de Turing: acesso irrestrito e memória ilimitada;
- Para entender esse fenômeno, considere a situação análoga das linguagens de programação. Muitas, tais como Pascal, Smalltalk, Prolog, Haskell, LISP, etc, parecem ser muito diferentes no que se refere a **estilo** e **estrutura**. Pode algum algoritmo ser programado em uma dessas linguagens e não na outra ? Claro que não.

Equivalência com outros modelos

- Podemos converter programas entre as duas linguagens, o que quer dizer que as duas reconhecem exatamente a mesma classe de algoritmos. Da mesma forma acontece com todas as outras linguagens de programação. A equivalência encontrada entre os modelos computacionais segue o mesmo princípio.
- Quaisquer dois modelos computacionais que satisfaçam alguns requerimentos podem simular um ao outro e assim são equivalentes quanto ao poder.
- Esse fenômeno de equivalência tem implicação profunda para os matemáticos - logo a classe de problemas será descrita de maneira próxima - por um algoritmo.

Autômatos Linearmente Limitados

- Do inglês: Linear Bounded Automata (LBA)
- São semelhantes às máquinas de Turing com uma diferença:
  - O espaço da fita da cadeia de entrada é o único espaço da fita permitido para uso.



Sumário

Algoritmo: Uma definição

- Algoritmo: uma coleção de instruções simples para a realização de alguma tarefa.
- Uma noção antiga e intuitiva até o século XX, a qual não era suficiente para ter noção mais profunda sobre o algoritmo.

O décimo problema de Hilbert

- Em 1900, David Hilbert (1862-1943) apresentou 23 problemas matemáticos. O décimo problema em sua lista dizia respeito aos algoritmos;
- Seja $6x^3yz^2 + 3xy^2 - x^3 - 10$ um polinômio de quatro termos sobre as variáveis x, y, e z. As raízes de um polinômio é uma atribuição de valores às variáveis de forma que o resultado do polinômio seja zero, isto é $P(x, y, z) = 0$.
- Esse polinômio tem uma raiz dada por x=5, y=3 e z=0;
- O décimo problema de Hilbert era conceber um algoritmo que testasse se um polinômio admitia raiz inteira ou não. Ele não usou o termo algoritmo, mas *um processo que através do qual a solução possa ser determinada com um número finito de operações*.

Tese de Church-Turing

- Assim, Hilbert aparentemente assumiu que tal algoritmo deveria existir $\Rightarrow$ alguém só precisava encontrá-lo!
- Para o 10^o problema de Hilbert, não existe algoritmo que realize essa tarefa, ele é insolúvel por um algoritmo. Mas essa insolubilidade só poderia ser mostrada se o conceito de algoritmo estivesse conhecido.
- Alonzo Church e Alan Turing em 1936 fizeram suas propostas.
- A conexão entre a noção formal de algoritmos e a noção precisa veio a ser chamada de **Tese de Church-Turing**.

Noção intuitiva de algoritmos	é igual a	algoritmos de máquina de Turing

Tese de Church-Turing

- A tese de Church-Turing provê a definição de algoritmo necessária para resolver o décimo problema de Hilbert.
- Em 1970, Yuri Matijasevic, baseado no trabalho de Martin Davis, Hilary Putnam e Julia Robinson, mostrou que nenhum algoritmo existe para se testar se um polinômio tem raízes inteiras.

Tese de Church-Turing

- O estudo das máquinas de Turing se volta para a formalização da idéia de máquina.
- Há aspectos dos computadores que Máquinas de Turing não modelam. Um deles é eficiência computacional.
- As máquinas de Turing respondem, entretanto, a questões sobre a aceitação de linguagens e sobre computação de funções.

Tese de Church-Turing

- Tese de Church: as Máquinas de Turing são versões formais de algoritmos, e nenhum procedimento computacional é considerado um algoritmo a não ser que possa ser representado na forma de uma máquina de Turing.
- A Tese de Church não é um teorema. No entanto é pouco provável que se venha propor um modelo alternativo que execute alguma operação não executável por máquina de Turing.

Descrição da Máquinas de Turing

- Continuaremos a falar de máquinas de Turing, mas o nosso verdadeiro foco a partir de agora é algoritmos. Ou seja, a máquina de Turing simplesmente serve como modelo preciso para a definição de algoritmo.
- Qual é o nível certo de detalhamento que se deve usar quando descreve tais algoritmos ?
- Consideremos três níveis: **descrição formal** diagramas de estados da MT, o **nível de implementação** na MT, e o **alto-nível**, quando usamos linguagem natural.
- A descrição de alto-nível tem sido e é o suficiente, com uma boa dose de paciência e cuidado chega-se a outras descrições.

Exemplo

Seja A a linguagem consistindo em todas as cadeias representando grafos não-direcionados que são conexos. Escrevemos:

$$A = \{\langle G \rangle | G \text{ é um grafo não-direcionado conexo}\}.$$

Exemplo

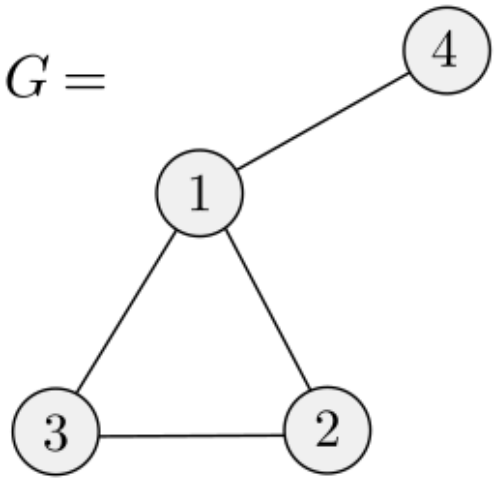
O que se segue é uma descrição de alto nível de uma MT M que decide A.
M = "Sobre a entrada $\langle G \rangle$, a codificação de um grafo G:

1. Selecione o primeiro nó de G e marque-o;
2. Repita o seguinte estágio até que nenhum novo nó seja marcado:
3. Para cada nó em G, marque-o, se ele estiver ligado por um aresta a um nó que já esteja marcado.
4. Faça uma varredura em todos os nós de G para determinar se eles estão todos marcados. Se estiverem, *aceite*; caso contrário, *rejeite*;"

Exemplo

Um grafo G e sua codificação $\langle G \rangle$

$G =$



$\langle G \rangle =$

(1,2,3,4)((1,2),(2,3),(3,1),(1,4))

Sumário

Exercícios

1) Construa uma máquina de Turing multifita que decida a seguinte linguagem:

$$L = \{\omega\#\omega | \omega \in \{0,1\}^*\}$$

2) Um palíndromo é uma palavra cuja leitura é a mesma tanto da esquerda para a direita quanto da direita para a esquerda. Por exemplo, a palavra 1001001 é um palíndromo. Construa uma máquina de Turing multifita que decida a seguinte linguagem:

$$L = \{\omega | \omega \in \{0,1\}^* \text{ e } \omega \text{ é um palíndromo}\}$$