Teoria da Computação

Máquinas de Turing

Aula 09

Prof. Felipe A. Louza

UFU

Roteiro

Introdução

Até agora, vimos diferentes autômatos como modelos de computação:

1. **AF**: memória pequena (estado atual)
2. **AP**: memória ilimitada, mas utilizada apenas no modelo da pilha.

Vimos que algumas tarefas simples estão além das capacidades desses modelos.

- Portanto, eles são muito restritos para servir como modelos de computadores de propósito geral.

Introdução

Até agora, vimos diferentes autômatos como modelos de computação:

1. **AF**: memória pequena (estado atual)
2. **AP**: memória ilimitada, mas utilizada apenas no modelo da pilha.

Vimos que algumas tarefas simples estão além das capacidades desses modelos.

- Portanto, eles são muito restritos para servir como modelos de computadores de propósito geral.

Introdução

Até agora, vimos diferentes autômatos como modelos de computação:

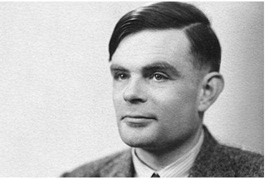
1. **AF**: memória pequena (estado atual)
2. **AP**: memória ilimitada, mas utilizada apenas no modelo da pilha.

Vimos que algumas tarefas simples estão além das capacidades desses modelos.

- Portanto, eles são muito restritos para servir como modelos de computadores de propósito geral.

Introdução

Vamos ver agora um modelo muito mais poderoso, proposto por **Alan Turing** em 1936.



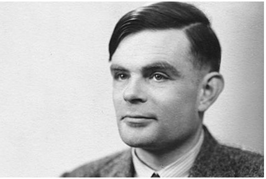
- **Máquina de Turing (MT)**:
  - Semelhante a um **AF**, mas com memória ilimitada e acesso irrestrito.

Uma máquina de Turing é capaz de fazer tudo o que um computador real pode fazer.

MTs foram propostas com o nome de **a-machines**, *"automatic machines"*, Alonzo Church quem as renomeou mais tarde.

Introdução

Vamos ver agora um modelo muito mais poderoso, proposto por **Alan Turing** em 1936.



- **Máquina de Turing (MT)**:
  - Semelhante a um **AF**, mas com memória ilimitada e acesso irrestrito.

Uma máquina de Turing é capaz de fazer tudo o que um computador real pode fazer.

MTs foram propostas com o nome de **a-machines**, *"automatic machines"*, Alonzo Church quem as renomeou mais tarde.

Introdução

Vamos utilizar esse modelo para estudar as possibilidades/limites da **Computação**:

- Computabilidade: o que é possível fazer com um computador.
- Complexidade: quão difícil é para resolver um problema computável.

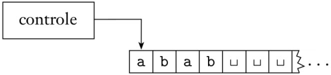
Introdução

Vamos utilizar esse modelo para estudar as possibilidades/limites da **Computação**:

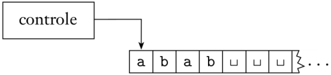
- Computabilidade: o que é possível fazer com um computador.
- Complexidade: quão difícil é para resolver um problema computável.

Introdução

O modelo de uma MT possui um controle de estados finitos e uma fita de leitura/escrita.



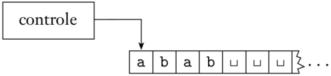
O modelo de uma MT possui um controle de estados finitos e uma fita de leitura/escrita.



- A fita é infinita (para a direita); ← a cadeia w será inserida na fita
- O cursor de leitura/escrita pode se mover para Esquerda/Direita;
- Temos dois estados especiais com efeito imediato:
  - q_{aceita}: a MT pára e
  - $q_{rejeita}$: a MT pára e rejeita a

Introdução

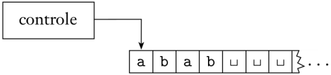
O modelo de uma MT possui um controle de estados finitos e uma fita de leitura/escrita.



- A fita é infinita (para a direita); $\leftarrow$ a cadeia w será inserida na fita
- O cursor de leitura/escrita pode se mover para **E**squerda/**D**ireita;
- Temos dois estados especiais com efeito imediato:
  - q_{aceita}: a MT pára e
  - $q_{rejeita}$: a MT pára e rejeita w;

Introdução

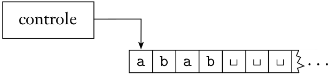
O modelo de uma MT possui um controle de estados finitos e uma fita de leitura/escrita.



- A fita é infinita (para a direita); $\leftarrow$ a cadeia w será inserida na fita
- O cursor de leitura/escrita pode se mover para **E**squerda/**D**ireita;
- Temos dois estados especiais com efeito imediato:
  1. q_{aceita}: a MT **pára** e aceita w;
  2. $q_{rejeita}$: a MT **pára** e rejeita w;

Introdução

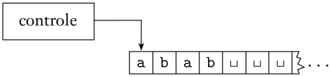
O modelo de uma MT possui um controle de estados finitos e uma fita de leitura/escrita.



- A fita é infinita (para a direita); $\leftarrow$ a cadeia w será inserida na fita
- O cursor de leitura/escrita pode se mover para **E**squerda/**D**ireita;
- Temos dois estados especiais com efeito imediato:
  1. q_{aceita}: a MT **pára** e aceita w;
  2. $q_{rejeita}$: a MT **pára** e rejeita w;

Introdução

O modelo de uma MT possui um controle de estados finitos e uma fita de leitura/escrita.



- A fita é infinita (para a direita); $\leftarrow$ a cadeia w será inserida na fita
- O cursor de leitura/escrita pode se mover para **E**squerda/**D**ireita;
- Temos dois estados especiais com efeito imediato:
  1. q_{aceita}: a MT **pára** e aceita w;
  2. $q_{rejeita}$: a MT **pára** e rejeita w;

Exemplo de MT

Vamos introduzir uma **MT** M_1 para a reconhecer a linguagem:

$$L_1 = \{w\#w \mid w \in \{0,1\}^*\}$$

A entrada (palavra) pode ser muito longa para ser memorizada.

$$0\ 1\ 1\ 0\ 0\ 0\ \#\ 0\ 1\ 1\ 0\ 0\ 0\ \sqcup\ \ldots$$

- Mas é permitido mover o **cursor** da-esquerda-para-direita e ao contrário;
- Podemos deixar marcas sobre a palavra na fita.
- Vamos adotar a estratégia de *"ziguezaguear"* em $w\#w$.

Exemplo de MT

Vamos introduzir uma **MT** M_1 para a reconhecer a linguagem:

$$L_1 = \{w\#w \mid w \in \{0,1\}^*\}$$

A entrada (palavra) pode ser muito longa para ser memorizada.

0 1 1 0 0 0 # 0 1 1 0 0 0 ␣ . . .

- Mas é permitido mover o **cursor** da-esquerda-para-direita e ao contrário;
- Podemos deixar marcas sobre a palavra na fita.
- Vamos adotar a estratégia de *"ziguezaguear"* em $w\#w$.

Exemplo de MT

Vamos introduzir uma **MT** M_1 para a reconhecer a linguagem:

$$L_1 = \{w\#w \mid w \in \{0,1\}^*\}$$

A entrada (palavra) pode ser muito longa para ser memorizada.

$$\overset{\downarrow}{0}\ 1\ 1\ 0\ 0\ 0\ \#\ 0\ 1\ 1\ 0\ 0\ 0\ \sqcup\ \ldots$$

- Mas é permitido mover o **cursor** da-esquerda-para-direita e ao contrário;
- Podemos deixar marcas sobre a palavra na fita.
- Vamos adotar a estratégia de *"ziguezaguear"* em $w\#w$.

Exemplo de MT

- Considere $w_1 = 011000\#011000 \in L_1$:

$$0\ 1\ 1\ 0\ 0\ 0\ \#\ 0\ 1\ 1\ 0\ 0\ 0\ \sqcup\ \ldots$$

Exemplo de MT

- Considere $w_1 = 011000\#011000 \in L_1$:

```
0 1 1 0 0 0 # 0 1 1 0 0 0 ⊔ ...
x 1 1 0 0 0 # 0 1 1 0 0 0 ⊔ ...
x 1 1 0 0 0 # x 1 1 0 0 0 ⊔ ...
x 1 1 0 0 0 # x 1 1 0 0 0 ⊔ ...
x x 1 0 0 0 # x 1 1 0 0 0 ⊔ ...
              ...
x x x x x x # x x x x x x ⊔ ...
                          aceita
```

Exemplo de MT

0 1 1 0 0 0 # 0 1 1 0 0 0 ␣ . . .

Descrição de alto-nível da **MT** M_1:

- “Sobre a cadeia de entrada w_1:

Exemplo de MT

0 1 1 0 0 0 # 0 1 1 0 0 0 ⊔ . . .

Descrição de alto-nível da **MT** M_1:

- "Sobre a cadeia de entrada w_1:
    1. Faça uma varredura na fita para se assegurar que w_1 possui somente um '#'. Caso contrário, rejeite a cadeia ✗
    2. Faça um *"ziguezague"* na fita para corresponder as posições nos dois lados de '#'. Marque-os a medida que eles forem verificados.
    3. Quando todos os símbolos à esquerda de '#' tiverem sido marcados, verifique se ainda há algum símbolo remanescente à direita de '#'. Caso haja, rejeite a cadeia ✗, senão aceite a cadeia ✓"

Exemplo de MT

0 1 1 0 0 0 # 0 1 1 0 0 0 ␣ ...

Descrição de alto-nível da **MT** M_1:

- "Sobre a cadeia de entrada w_1:
  1. Faça uma varredura na fita para se assegurar que w_1 possui somente um '#'. Caso contrário, rejeite a cadeia ✗
  2. Faça um *"ziguezague"* na fita para corresponder as posições nos dois lados de '#'. Marque-os a medida que eles forem verificados.
  3. Quando todos os símbolos à esquerda de '#' tiverem sido marcados, verifique se ainda há algum símbolo remanescente à direita de '#'. Caso haja, rejeite a cadeia ✗, senão aceite a cadeia ✓"

Exemplo de MT

0 1 1 0 0 0 # 0 1 1 0 0 0 ␣ . . .

Descrição de alto-nível da **MT** M_1:

- "Sobre a cadeia de entrada w_1:
  1. Faça uma varredura na fita para se assegurar que w_1 possui somente um '#'. Caso contrário, rejeite a cadeia ✗
  2. Faça um *"ziguezague"* na fita para corresponder as posições nos dois lados de '#'. Marque-os a medida que eles forem verificados.
  3. Quando todos os símbolos à esquerda de '#' tiverem sido marcados, verifique se ainda há algum símbolo remanescente à direita de '#'. Caso haja, rejeite a cadeia ✗, senão aceite a cadeia ✓"

Exemplo de MT

0 1 1 0 0 0 # 0 1 1 0 0 0 ⊔ . . .

Descrição de alto-nível da **MT** M_1:

- "Sobre a cadeia de entrada w_1:
  1. Faça uma varredura na fita para se assegurar que w_1 possui somente um '#'. Caso contrário, rejeite a cadeia ✗
  2. Faça um *"ziguezague"* na fita para corresponder as posições nos dois lados de '#'. Marque-os a medida que eles forem verificados.
  3. Quando todos os símbolos à esquerda de '#' tiverem sido marcados, verifique se ainda há algum símbolo remanescente à direita de '#'. Caso haja, rejeite a cadeia ✗, senão aceite a cadeia ✓"

Tipos de descrições de uma MT

Essa descrição de alto-nível (algoritmo) esboça a maneira como a MT funciona, sem muitos detalhes.

- Vamos ver agora, como descrever MTs em todos os detalhes, com notações formais[1].

[1] Análogas àquelas notações introduzidas para os AFs e APs.

Roteiro

Formalização da Máquina de Turing

Definição

Uma *Máquina de Turing* é uma 7-upla
$M = (Q, \Sigma, \Gamma, \delta, q_0, q_{aceita}, q_{rejeita})$, para Q, Σ e Γ conjuntos finitos

1. Q o conjunto dos estados
2. Σ o alfabeto de entrada (sem símbolo em branco ␣)
3. Γ é o alfabeto da fita (␣$\in \Gamma$ e $\Sigma \subseteq \Gamma$)
4. $\delta : Q \times \Gamma \rightarrow Q \times \Gamma \times \{E, D\}$ é a função de transição, no formato:

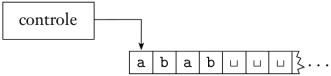
$$\delta(q, a) = (p, b, S)$$

5. q_0 é o estado inicial
6. $q_{aceita} \in Q$ é o estado de aceitação
7. $q_{rejeita} \in Q$ é o estado de rejeição, $q_{aceita} \neq q_{rejeita}$

S assume $\{E, D\}$, os lados esquerdo e direito.

Formalização da Máquina de Turing

A função de transição δ é o "cérebro" da MT (controle).

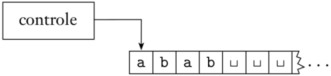


- Durante a aplicação de δ, podemos mudar:
  - o estado atual $q \in Q$;
  - o conteúdo da fita; e
  - a posição do cursor de leitura/escrita.
- Exemplo:

$$\delta(q, a) = (p, b, D)$$

Formalização da Máquina de Turing

A função de transição δ é o "cérebro" da MT (controle).

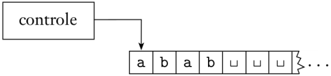


- Durante a aplicação de δ, podemos mudar:
  - o estado atual $q \in Q$;
  - o conteúdo da fita; e
  - a posição do cursor de leitura/escrita.
- Exemplo:

$$\delta(q, a) = (p, b, D)$$

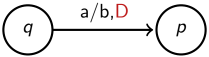
Formalização da Máquina de Turing

A função de transição δ é o "cérebro" da MT (controle).



- Durante a aplicação de δ, podemos mudar:
  - o estado atual $q \in Q$;
  - o conteúdo da fita; e
  - a posição do cursor de leitura/escrita.
- Exemplo:

$$\delta(q, a) = (p, b, D)$$



Formalização da Máquina de Turing

Funcionamento de uma MT M:

1. Primeiro, $w = w_1 w_2 \ldots w_n$ é escrita na fita $\leftarrow$ posição mais à esquerda
2. A MT começa com o cursor na primeira posição da fita.
3. A computação procede de acordo a função de transição:

$$\delta(q, a) = (p, b, \{E, D\})$$

   – Quando M está em q, e lê ‘a’ da fita:
   1. M vai para o estado p;
   2. M escreve ‘b’ na fita (no lugar de ‘a’); e
   3. O cursor vai para a Esquerda ou Direita.
4. A computação continua até que M alcance q_{aceita} ou q_{aceita}, e a máquina pára $\leftarrow$efeito imediato.

Formalização da Máquina de Turing

Funcionamento de uma MT M:

1. Primeiro, $w = w_1 w_2 \ldots w_n$ é escrita na fita $\leftarrow$ posição mais à esquerda
2. A MT começa com o cursor na primeira posição da fita.
3. A computação procede de acordo a função de transição:

$$\delta(q, a) = (p, b, \{E, D\})$$

   – Quando M está em q, e lê 'a' da fita:
   1. M vai para o estado p;
   2. M escreve 'b' na fita (no lugar de 'a'); e
   3. O cursor vai para a Esquerda ou Direita.
4. A computação continua até que M alcance q_{aceita} ou q_{aceita}, e a máquina pára $\leftarrow$efeito imediato.

Formalização da Máquina de Turing

Funcionamento de uma MT M:

1. Primeiro, $w = w_1 w_2 \dots w_n$ é escrita na fita ← posição mais à esquerda
2. A MT começa com o cursor na primeira posição da fita.
3. A computação procede de acordo a função de transição:

$$\delta(q, a) = (p, b, \{E, D\})$$

– Quando M está em q, e lê ‘a’ da fita:

1. M vai para o estado p;
2. M escreve ‘b’ na fita (no lugar de ‘a’); e
3. O cursor vai para a Esquerda ou Direita.

4. A computação continua até que M alcance q_{aceita} ou q_{aceita}, e a máquina pára ←efeito imediato.

Formalização da Máquina de Turing

Funcionamento de uma MT M:

1. Primeiro, $w = w_1 w_2 \ldots w_n$ é escrita na fita $\leftarrow$ posição mais à esquerda
2. A MT começa com o cursor na primeira posição da fita.
3. A computação procede de acordo a função de transição:

$$\delta(q, a) = (p, b, \{E, D\})$$

   – Quando M está em q, e lê 'a' da fita:
      1. M vai para o estado p;
      2. M escreve 'b' na fita (no lugar de 'a'); e
      3. O cursor vai para a Esquerda ou Direita.

4. A computação continua até que M alcance q_{aceita} ou q_{aceita}, e a máquina pára $\leftarrow$efeito imediato.

Formalização da Máquina de Turing

Funcionamento de uma MT M:

1. Primeiro, $w = w_1 w_2 \dots w_n$ é escrita na fita $\leftarrow$ posição mais à esquerda
2. A MT começa com o cursor na primeira posição da fita.
3. A computação procede de acordo a função de transição:

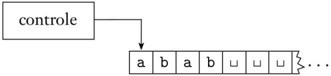
$$\delta(q, a) = (p, b, \{E, D\})$$

   – Quando M está em q, e lê 'a' da fita:
      1. M vai para o estado p;
      2. M escreve 'b' na fita (no lugar de 'a'); e
      3. O cursor vai para a Esquerda ou Direita.

4. A computação continua até que M alcance q_{aceita} ou q_{aceita}, e a máquina pára $\leftarrow$efeito imediato.

Formalização da Máquina de Turing

Algumas observações:



- Se M está na primeira posição da fita, e tentar fazer um movimento E, M permanece no mesmo lugar.

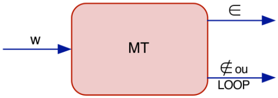
$$\delta(q, a) = (p, b, E)$$

1. M vai para o estado p;
2. M escreve 'b' na fita (no lugar de 'a'); e
3. Nenhum movimento é feito.

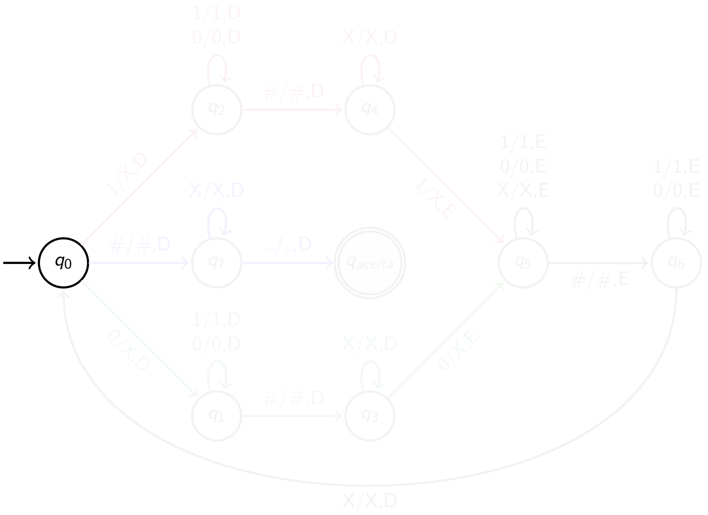
Formalização da Máquina de Turing

Algumas observações:

- A MT M pode não parar NUNCA.
  - Isso porquê não processamos $w = w_1 w_2 \ldots w_n$ uma única vez, podemos percorrer a fita indefinidamente.
  - M fica processando em ciclo (ou loop) infinito.

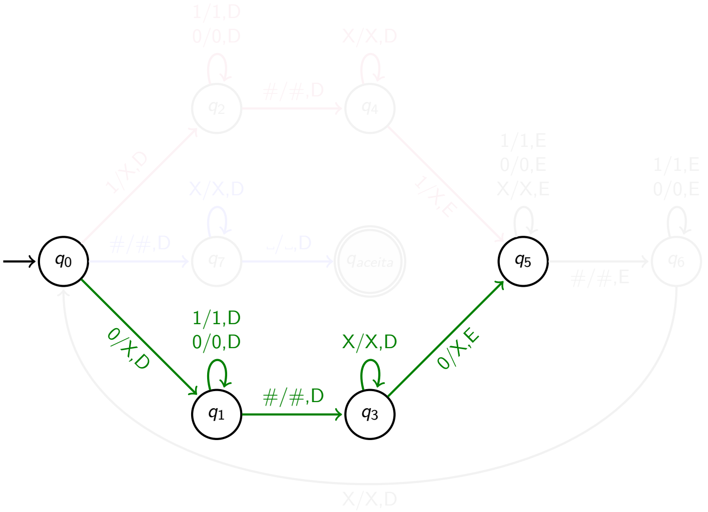


Descrição formal da **MT** M_1 para $L_1 = \{w\#w \mid w \in \{0,1\}^*\}$



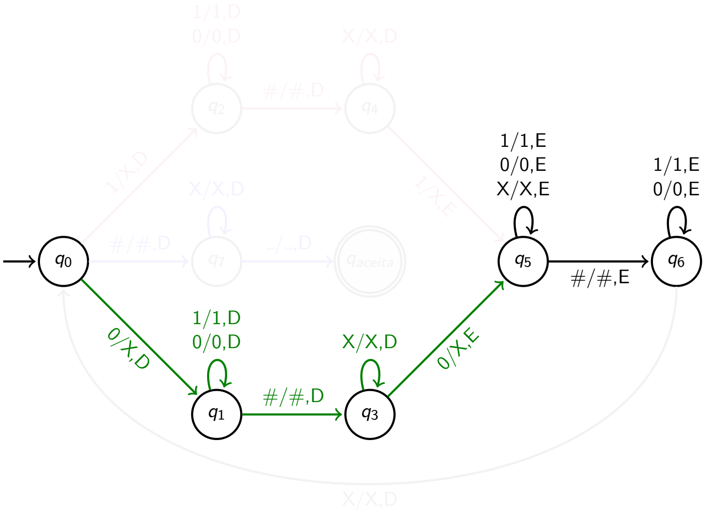
Transições implícitas para $q_{rejeita}$ sempre que $\delta(q, a) = \perp$.

Descrição formal da **MT** M_1 para $L_1 = \{w\#w \mid w \in \{0,1\}^*\}$



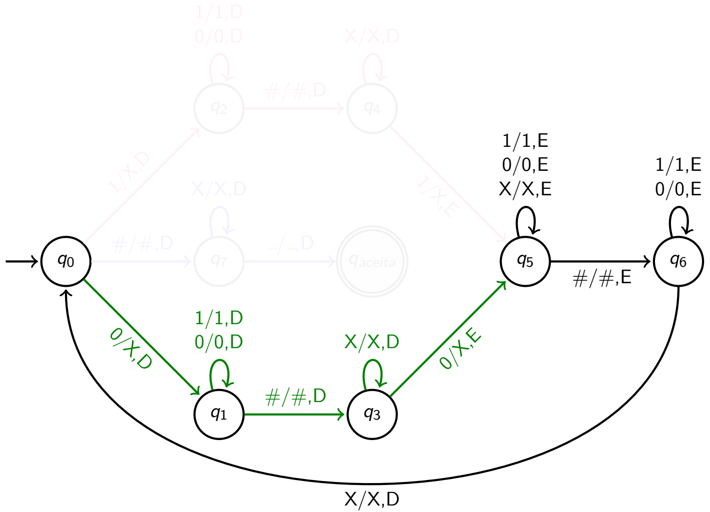
0 1 1 0 0 0 # 0 1 1 0 0 0 ␣ . . .

Descrição formal da **MT** M_1 para $L_1 = \{w\#w \mid w \in \{0,1\}^*\}$



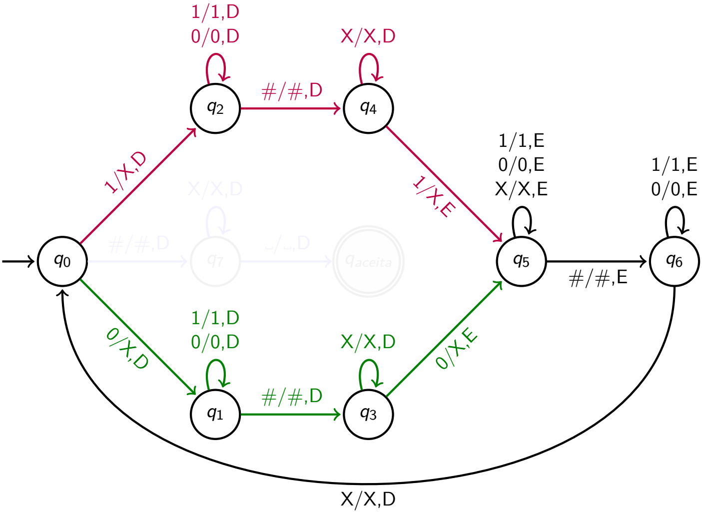
0 1 1 0 0 0 # 0 1 1 0 0 0 ⊔ . . .

Descrição formal da **MT** M_1 para $L_1 = \{w\#w \mid w \in \{0,1\}^*\}$



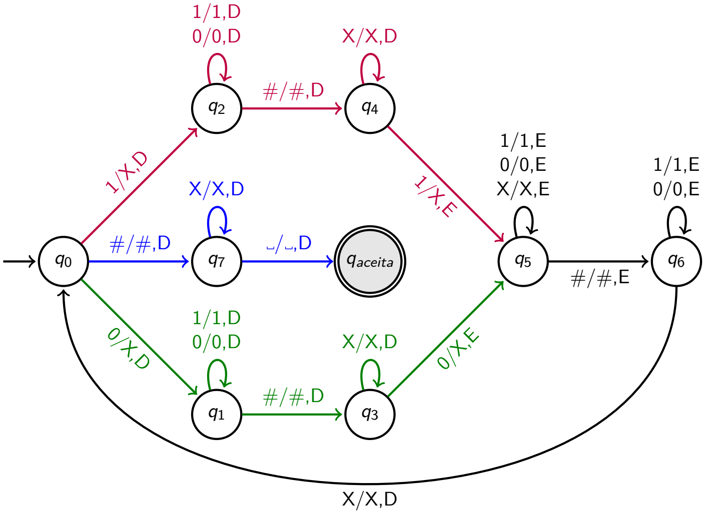
0 1 1 0 0 0 # 0 1 1 0 0 0 ␣ . . .

Descrição formal da **MT** M_1 para $L_1 = \{w\#w \mid w \in \{0,1\}^*\}$



0 1 1 0 0 0 # 0 1 1 0 0 0 ␣ . . .

Descrição formal da **MT** M_1 para $L_1 = \{w\#w \mid w \in \{0,1\}^*\}$



0 1 1 0 0 0 # 0 1 1 0 0 0 ␣ . . .

Roteiro

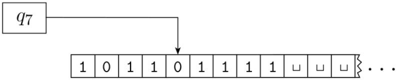
Configuração de uma MT

A configuração de uma MT descreve a situação atual da máquina:

- estado atual;
- conteúdo da fita; e
- posição do cursor de I/O

Vamos descrever essa configuração por uqv, $u, v \in \Gamma^*$ e $q \in Q$.

- Exemplo:



  - Configuração: $\underbrace{1011}_{u}\, q_7\, \underbrace{01111}_{v}$

O cursor está posicionado na 1ª posição de v.

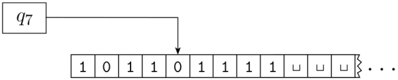
Configuração de uma MT

A configuração de uma MT descreve a situação atual da máquina:

- estado atual;
- conteúdo da fita; e
- posição do cursor de I/O

Vamos descrever essa configuração por uqv, $u, v \in \Gamma^*$ e $q \in Q$.

- Exemplo:



  – Configuração: $\underbrace{1011}_{u}\, q_7\, \underbrace{01111}_{v}$

O cursor está posicionado na 1ª posição de v.

Configuração de uma MT

A medida que a MT computa, mudanças ocorrem em sua configuração.

1. $q_0 101101111$
2. $0q_1 01101111$
3. $00q_3 1101111$
4. $0q_2 00101111$
5. ...
6. $0000q_7 01111$

Configuração de uma MT

A medida que a MT computa, mudanças ocorrem em sua configuração.

1. $q_0 101101111$
2. $0 q_1 01101111$
3. $00 q_3 1101111$
4. $0 q_2 00101111$
5. ...
6. $0000 q_7 01111$

Configuração de uma MT

A medida que a MT computa, mudanças ocorrem em sua configuração.

1. $q_0101101111$
2. $0q_101101111$
3. $00q_31101111$
4. $0q_200101111$
5. ...
6. $0000q_701111$

Configuração de uma MT

A medida que a MT computa, mudanças ocorrem em sua configuração.

1. $q_0 101101111$
2. $0 q_1 01101111$
3. $00 q_3 1101111$
4. $0 q_2 00101111$
5. ...
6. $0000 q_7 01111$

Configuração de uma MT

A medida que a MT computa, mudanças ocorrem em sua configuração.

1. $q_0 101101111$
2. $0 q_1 01101111$
3. $00 q_3 1101111$
4. $0 q_2 00101111$
5. ...
6. $0000 q_7 01111$

Configuração de uma MT

A medida que a MT computa, mudanças ocorrem em sua configuração.

1. $q_0 101101111$
2. $0 q_1 01101111$
3. $00 q_3 1101111$
4. $0 q_2 00101111$
5. ...
6. $0000 q_7 01111$

Configuração de uma MT

Dizemos que uma configuração C_1 origina C_2 se a MT M puder ir de C_1 para C_2 em um único passo, denotado por $C_1 \Rightarrow C_2$.

- Exemplo:

$$\underbrace{1011}_{u} q_i \underbrace{01111}_{v} \Rightarrow \underbrace{1011\#}_{u'} q_j \underbrace{1111}_{v'}$$

$$\delta(q_i, 0) = (q_j, \#, D)$$

Configuração de uma MT

Dizemos que uma configuração C_1 origina C_2 se a MT M puder ir de C_1 para C_2 em um único passo, denotado por $C_1 \Rightarrow C_2$.

- Exemplo:

$$\underbrace{1011}_{u}\, q_i\, \underbrace{01111}_{v} \Rightarrow \underbrace{1011\#}_{u'}\, q_j\, \underbrace{1111}_{v'}$$

$$\delta(q_i, 0) = (q_j, \#, D)$$

Configuração de uma MT

Caso especial:

$$\underbrace{\quad\quad}_{u} \; q_i \; \underbrace{101101111}_{v} \Rightarrow \underbrace{\quad\quad}_{u'} \; q_j \; \underbrace{\#01101111}_{v'}$$

$$\delta(q_i, 1) = (q_j, \#, E)$$

Configuração inicial:

$$\underbrace{\quad\quad}_{u} \; q_0 \; \underbrace{101101111}_{v}$$

O cursor nunca ultrapassa a extremidade à esquerda.

Configuração de uma MT

Caso especial:

$$\underbrace{\quad}_{u}\ q_i\ \underbrace{101101111}_{v} \Rightarrow \underbrace{\quad}_{u'}\ q_j\ \underbrace{\#01101111}_{v'}$$

$$\delta(q_i, 1) = (q_j, \#, E)$$

Configuração inicial:

$$\underbrace{\quad}_{u}\ q_0\ \underbrace{101101111}_{v}$$

O cursor nunca ultrapassa a extremidade à esquerda.

Configuração de uma MT

Configuração de parada:

- Aceitação: se o estado da configuração é q_{aceita}.
- Rejeição: se o estado da configuração é $q_{rejeita}$.

Essas configurações não originam outras (a máquina pára).

- Poderíamos redefinir:

$$\delta : Q' \times \Gamma \to Q \times \Gamma \times \{E, D\}$$

para

$$Q' = Q \setminus \{q_{aceita}, q_{rejeita}\}$$

Acessar um desses estados tem efeito instantâneo.

Configuração de uma MT

Configuração de parada:

- Aceitação: se o estado da configuração é q_{aceita}.
- Rejeição: se o estado da configuração é $q_{rejeita}$.

Essas configurações não originam outras (a máquina pára).

- Poderíamos redefinir:

$$\delta : Q' \times \Gamma \to Q \times \Gamma \times \{E, D\}$$

para

$$Q' = Q \setminus \{q_{aceita}, q_{rejeita}\}$$

Acessar um desses estados tem efeito instantâneo.

Configuração de uma MT

Configuração de parada:

- Aceitação: se o estado da configuração é q_{aceita}.
- Rejeição: se o estado da configuração é $q_{rejeita}$.

Essas configurações não originam outras (a máquina pára).

- Poderíamos redefinir:

$$\delta : Q' \times \Gamma \to Q \times \Gamma \times \{E, D\}$$

para

$$Q' = Q \setminus \{q_{aceita}, q_{rejeita}\}$$

Acessar um desses estados tem efeito instantâneo.

Configuração de uma MT

Definição

Uma MT M aceita uma cadeia $w = w_1 w_2 \dots w_n$ se existir uma sequência de configurações:

$$C_1 \Rightarrow C_2 \Rightarrow \cdots \Rightarrow C_k$$

onde:

1. C_1 é a configuração inicial $\underbrace{\quad}_{u} q_0 \underbrace{w_1 w_2 \dots w_n}_{v}$
2. Cada C_i origina C_{i+1}
3. C_k é uma configuração de aceitação: $\underbrace{w_1 \dots w_{i-1}}_{u} q_{aceita} \underbrace{w_i \dots w_n}_{v}$

- A MT M rejeita w se em (3) o estado da configuração for $q_{rejeita}$

Configuração de uma MT

Definição

Uma MT M aceita uma cadeia $w = w_1 w_2 \dots w_n$ se existir uma sequência de configurações:

$$C_1 \Rightarrow C_2 \Rightarrow \cdots \Rightarrow C_k$$

onde:

1. C_1 é a configuração inicial $\underbrace{}_{u} q_0 \underbrace{w_1 w_2 \dots w_n}_{v}$
2. Cada C_i origina C_{i+1}
3. C_k é uma configuração de aceitação: $\underbrace{w_1 \dots w_{i-1}}_{u} q_{aceita} \underbrace{w_i \dots w_n}_{v}$

- A MT M rejeita w se em (3) o estado da configuração for $q_{rejeita}$

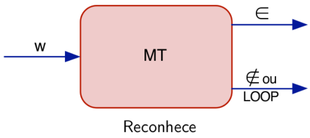
Linguagem reconhecida por uma MT

Definição

O conjundo das cadeias aceitas por uma MT M é a linguagem de M, ou a linguagem reconhecida por M, denotada por $\mathbf{L(M)}$.

Se $w \notin L(M)$, então M pode:

- Parar no estado de rejeição, **ou**
- Entrar em loop.



Roteiro

Linguagens Recursivamente Enumeráveis

Definição

Uma linguagem L é chamada de **Turing-reconhecível** se existe alguma MT M tal que $L = L(M)$, ou seja:

- Se $w \in L$, então M:
  - Pára no estado de aceitação.

Linguagens Recursivamente Enumeráveis

Linguagens **Turing-reconhecíveis** também são chamadas de:

- Linguagens Recursivamente Enumeráveis (LRE); ou
- Linguagens Irrestritas; $\leftarrow$ geradas por Gramáticas Irrestritas (**Tipo 0**).

Gramáticas Irrestritas:

- São a maneira mais geral de se definir um conjunto de palavras recursivamente.

Linguagens Recursivamente Enumeráveis

Linguagens **Turing-reconhecíveis** também são chamadas de:

- Linguagens Recursivamente Enumeráveis (LRE); ou
- Linguagens Irrestritas; ← geradas por Gramáticas Irrestritas (**Tipo 0**).

Gramáticas Irrestritas:

- São a maneira mais geral de se definir um conjunto de palavras recursivamente.

Linguagens Recursivamente Enumeráveis

Linguagens **Turing-reconhecíveis** também são chamadas de:

- Linguagens Recursivamente Enumeráveis (LRE); ou
- Linguagens Irrestritas; $\longleftarrow$ geradas por Gramáticas Irrestritas (**Tipo 0**).

Gramáticas Irrestritas:

- São a maneira mais geral de se definir um conjunto de palavras recursivamente.

Linguagens Recursivamente Enumeráveis

Linguagens **Turing-reconhecíveis** também são chamadas de:

- Linguagens Recursivamente Enumeráveis (LRE); ou
- Linguagens Irrestritas; $\leftarrow$ geradas por Gramáticas Irrestritas (**Tipo 0**).

Gramáticas Irrestritas:

- São a maneira mais geral de se definir um conjunto de palavras recursivamente.

Gramáticas Irrestritas

Definição:

Qualquer gramática $G = (V, \Sigma, S, P)$ é Irrestrita (ou **Tipo 0**).

Não existem restrições na forma das produções para as gramáticas desta classe, apenas:

$$\alpha \rightarrow \beta$$

com $\alpha \neq \mathcal{E}$ e $\alpha, \beta \in (\Sigma \cup V)^*$.

Exemplo:

$$G_1 = (\{S, A, B\}, \{a, b, c\}, P, S)$$

$$\begin{aligned} P: \quad & S \rightarrow \mathbf{a}SB\mathbf{c} \mid \mathbf{abc} \mid \mathcal{E} \\ & \mathbf{c}B \rightarrow B\mathbf{c} \\ & \mathbf{b}B \rightarrow \mathbf{bb} \end{aligned}$$

Linguagens Recursivamente Enumeráveis

LREs englobam todas as linguagens que podem ser reconhecidas computacionalmente.

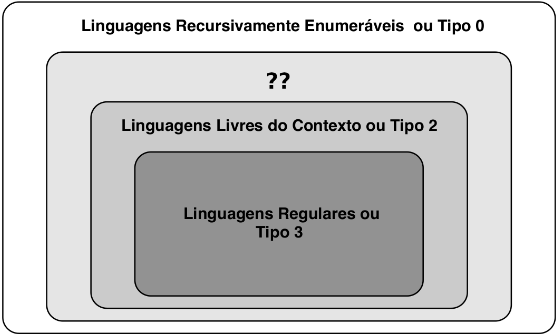


Figura: Hierarquia de Chomsky

Antes do **Tipo 1**, vamos ver uma sub-classe das LREs.

Linguagens Turing-Decidíveis

Quando iniciamos uma MT, podemos ter três resultado ao processar w:

- aceita ✓
- rejeita ✗
- entra em loop.

Linguagens Turing-Decidíveis

Pode ser difícil decidir se uma MT está em loop ou apenas levando muito tempo para processar uma cadeia.

- MTs que sempre param são mais interessantes.
- Essas máquinas são chamadas de Decisores.

A MT que *"decide"* uma linguagem sempre pára.

Linguagens Turing-Decidíveis

Pode ser difícil decidir se uma MT está em loop ou apenas levando muito tempo para processar uma cadeia.

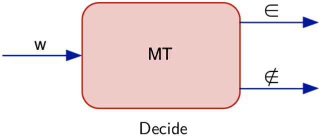
- MTs que sempre param são mais interessantes.
- Essas máquinas são chamadas de Decisores.

A MT que *"decide"* uma linguagem sempre pára.

Linguagens Turing-Decidíveis

Definição

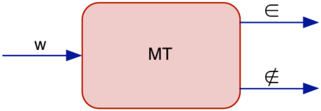
Uma linguagem L é chamada de **Turing-decidível** se existe alguma MT M que a decide.



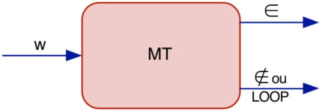
Linguagens Decidíveis também são chamadas de Recursivas.

Linguagens Turing-Decidíveis

Decidir uma linguagem L é <u>mais forte</u> do que reconhecer L.



Decide



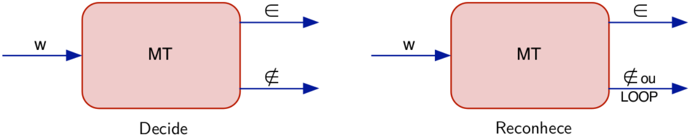
Reconhece

Toda linguagem *“Turing-decidível”* é também *“Turing-reconhecível”*.

- Mas o contrário não vale.

Linguagens Turing-Decidíveis

Decidir uma linguagem L é mais forte do que reconhecer L.



Toda linguagem *"Turing-decidível"* é também *"Turing-reconhecível"*.

- Mas o contrário não vale.

Linguagens Turing-Decidíveis

As linguagens *"Turing-decidíveis"* (ou Recursivas) não foram incluídas na Hierarquia de Chomsky:

- Como uma subclasse do **Tipo 0** (entre o **Tipo 1**).

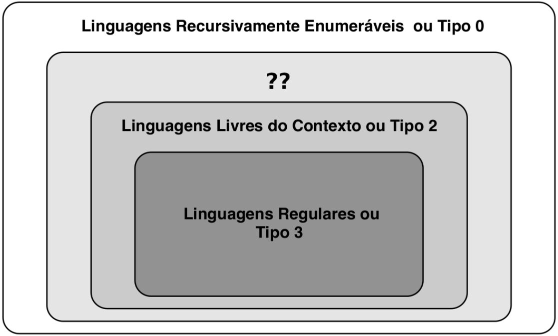


Figura: Hierarquia de Chomsky

Roteiro

1. Máquinas de Turing
   - Formalização da Máquina de Turing
   - Configuração de uma Máquina de Turing
   - Linguagens Recursivamente Enumeráveis

2. Linguagens Sensíveis ao Contexto
   - Gramáticas Sensíveis ao Contexto
   - Autômatos Limitados Linearmente
   - Hierarquia de Chomsky

3. Referências

Roteiro

1. Máquinas de Turing
   - Formalização da Máquina de Turing
   - Configuração de uma Máquina de Turing
   - Linguagens Recursivamente Enumeráveis
2. Linguagens Sensíveis ao Contexto
   - Gramáticas Sensíveis ao Contexto
   - Autômatos Limitados Linearmente
   - Hierarquia de Chomsky
3. Referências

Gramáticas Sensíveis ao Contexto

Definição:

Uma gramática $G = (V, \Sigma, S, P)$ é do **Tipo 1** se toda produção de P for da forma:

$$vAw \rightarrow vzw$$

com $A \in V$, $v, w \in (\Sigma \cup V)^*$ e
$z \in (\Sigma \cup V)^+$ (isto é, $z \neq \mathcal{E}$).

Além disso, permite-se uma única regra $S \rightarrow \mathcal{E}$ quando S não aparece do lado direito de nenhuma produção.

Estas são as **Gramáticas Sensíveis de Contexto (GSC)**.

Como $z \neq \mathcal{E}$, o comprimento das formas sentencias em uma derivação nunca diminui.

Gramáticas Sensíveis ao Contexto

Definição:

Uma gramática $G = (V, \Sigma, S, P)$ é do **Tipo 1** se toda produção de P for da forma:

$$vAw \rightarrow vzw$$

com $A \in V$, $v, w \in (\Sigma \cup V)^*$ e
$z \in (\Sigma \cup V)^+$ (isto é, $z \neq \mathcal{E}$).

Além disso, permite-se uma única regra $S \rightarrow \mathcal{E}$ quando S não aparece do lado direito de nenhuma produção.

Estas são as **Gramáticas Sensíveis de Contexto (GSC)**.

Como $z \neq \mathcal{E}$, o comprimento das formas sentencias em uma derivação nunca diminui.

Gramáticas Sensíveis ao Contexto

Definição:

Uma gramática $G = (V, \Sigma, S, P)$ é do **Tipo 1** se toda produção de P for da forma:

$$vAw \to vzw$$

com $A \in V$, $v, w \in (\Sigma \cup V)^*$ e
$z \in (\Sigma \cup V)^+$ (isto é, $z \neq \mathcal{E}$).

Além disso, permite-se uma única regra $S \to \mathcal{E}$ quando S não aparece do lado direito de nenhuma produção.

Estas são as **Gramáticas Sensíveis de Contexto (GSC)**.

Como $z \neq \mathcal{E}$, o comprimento das formas sentencias em uma derivação nunca diminui.

Linguagens Sensíveis ao Contexto

Definição:

Seja $G = (V, \Sigma, P, S)$ uma GSC.
A linguagem gerada por G,

$$L(G) = \{w \in \Sigma^* \mid S \Rightarrow^+ w\}$$

é chamada de Linguagem Sensível ao Contexto (LSC), ou linguagem do **Tipo 1**.

Exemplo

$$L_2 = \{a^n b^n c^n \mid n \geq 1\}$$

$$G_2 = (\{S, A, B, C\}, \{a, b, c\}, P, S)$$

$$\begin{aligned} P: \quad & S \to \mathbf{a}SBC \mid \mathbf{a}BC \\ & CB \to BC \\ & \mathbf{a}B \to \mathbf{ab} \\ & \mathbf{b}B \to \mathbf{bb} \\ & \mathbf{b}C \to \mathbf{bc} \\ & \mathbf{c}C \to \mathbf{cc} \end{aligned}$$

Cadeias geradas:

$$S \Rightarrow aBC \Rightarrow abC \Rightarrow abc$$
$$S \Rightarrow aSBC \Rightarrow aaBCBC \Rightarrow^* aabbcc$$
$$S \Rightarrow^* aaabbbccc$$
$$\ldots$$

Vimos que essa linguagem não é LLC pelo lema do bombeamento.

Exemplo

$$L_2 = \{a^n b^n c^n \mid n \geq 1\}$$

$$G_2 = (\{S, A, B, C\}, \{a, b, c\}, P, S)$$

$$\begin{aligned} P: \quad S &\rightarrow \mathbf{a}SBC \mid \mathbf{a}BC \\ CB &\rightarrow BC \\ \mathbf{a}B &\rightarrow \mathbf{ab} \\ \mathbf{b}B &\rightarrow \mathbf{bb} \\ \mathbf{b}C &\rightarrow \mathbf{bc} \\ \mathbf{c}C &\rightarrow \mathbf{cc} \end{aligned}$$

Cadeias geradas:

$$S \Rightarrow aBC \Rightarrow abC \Rightarrow abc$$

$$S \Rightarrow aSBC \Rightarrow aaBCBC \Rightarrow^* aabbcc$$

$$S \Rightarrow^* aaabbbccc$$

$\ldots$

Vimos que essa linguagem não é LLC pelo lema do bombeamento.

Exemplo

$$L_2 = \{a^n b^n c^n \mid n \geq 1\}$$

$$G_2 = (\{S, A, B, C\}, \{a, b, c\}, P, S)$$

$$\begin{aligned} P: \quad & S \rightarrow \mathbf{a}SBC \mid \mathbf{a}BC \\ & CB \rightarrow BC \\ & \mathbf{a}B \rightarrow \mathbf{ab} \\ & \mathbf{b}B \rightarrow \mathbf{bb} \\ & \mathbf{b}C \rightarrow \mathbf{bc} \\ & \mathbf{c}C \rightarrow \mathbf{cc} \end{aligned}$$

Cadeias geradas:

$$S \Rightarrow aBC \Rightarrow abC \Rightarrow abc$$
$$S \Rightarrow aSBC \Rightarrow aaBCBC \Rightarrow^* aabbcc$$
$$S \Rightarrow^* aaabbbccc$$
$$\dots$$

Vimos que essa linguagem não é LLC pelo lema do bombeamento.

Exemplo

$$L_2 = \{a^n b^n c^n \mid n \geq 1\}$$

$$G_2 = (\{S, A, B, C\}, \{a, b, c\}, P, S)$$

$$\begin{aligned} P: \quad & S \to \mathbf{a}SBC \mid \mathbf{a}BC \\ & CB \to BC \\ & \mathbf{a}B \to \mathbf{ab} \\ & \mathbf{b}B \to \mathbf{bb} \\ & \mathbf{b}C \to \mathbf{bc} \\ & \mathbf{c}C \to \mathbf{cc} \end{aligned}$$

Cadeias geradas:

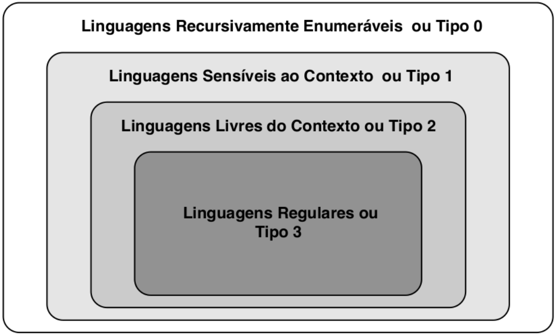
$$\begin{aligned} & S \Rightarrow aBC \Rightarrow abC \Rightarrow abc \\ & S \Rightarrow aSBC \Rightarrow aaBCBC \Rightarrow^* aabbcc \\ & S \Rightarrow^* aaabbbccc \\ & \ldots \end{aligned}$$

Vimos que essa linguagem não é LLC pelo lema do bombeamento.

Linguagens Sensíveis ao Contexto

Se uma linguagem L é gerada por uma **GSC**, logo L é do **Tipo 1**.

- Pela Hierarquia de Chomsky, toda **LLC** é **LSC**, mas não o contrário.
- Além disso, toda LSC é **Turing-reconhecível** e **Turing-decidível**

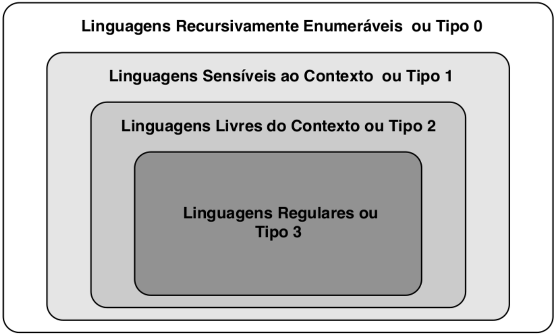


$\mathcal{L}_3 \subsetneq \mathcal{L}_2 \subsetneq \mathcal{L}_1 \subsetneq \mathcal{L}_0$

Linguagens Sensíveis ao Contexto

Se uma linguagem L é gerada por uma **GSC**, logo L é do **Tipo 1**.

- Pela Hierarquia de Chomsky, toda **LLC** é **LSC**, mas não o contrário.
- Além disso, toda LSC é **Turing-reconhecível** e **Turing-decidível**

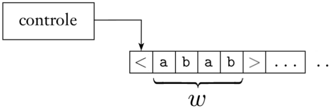


$\mathcal{L}_3 \subsetneq \mathcal{L}_2 \subsetneq \mathcal{L}_1 \subsetneq \mathcal{L}_0$

Roteiro

Autômatos Limitados Linearmente

Ou simplesmente, MT com fita limitada:

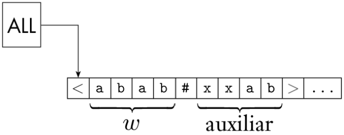


- Um Autômato Limitado Linearmente (ALL) é uma MT com fita limitada ao tamanho da entrada $|w| + 2$.
  - O funcionamento é igual à uma MT;
  - Dois delimitadores $<$ e $>$ são adicionados na fita entre o início e fim da cadeia w.
  - O cursor de I/O **não** é permitido mover para fora dos delimitadores.

Autômatos Limitados Linearmente

Um **ALL** só pode resolver problemas que requerem memória proporcional à usada para a entrada.

- Na verdade, é permitido que a memória disponível seja incrementada de no máximo um fator constante, linear em $|w|$ (daí o nome desse modelo).



Autômatos Limitados Linearmente

Definição

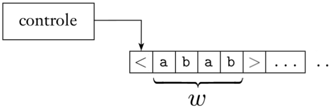
Um *Autômato Limitado Linearmente* (ALL) é uma 7-upla $A = (Q, \Sigma, \Gamma, \delta, q_0, q_{aceita}, q_{rejeita})$, para Q, Σ e Γ conjuntos finitos

1. Q o conjunto dos estados
2. Σ o alfabeto de entrada (sem símbolo em branco $\sqcup$)
3. Γ é o alfabeto da fita ($\sqcup \in \Gamma$ e $\Sigma \subseteq \Gamma$)
4. $\delta : Q \times \Gamma \to Q \times \Gamma \times \{E, D\}$ é a função de transição (próx. slide)
5. q_0 é o estado inicial
6. $q_{aceita} \in Q$ é o estado de aceitação
7. $q_{rejeita} \in Q$ é o estado de rejeição, $q_{aceita} \neq q_{rejeita}$

- Acrescentaremos os delimitadores na fita: '$<$' início da cadeia e, '$>$' final da cadeia.

Autômatos Limitados Linearmente

Em particular:



- δ é a função de transição redefinida:
  - $Q \times \Gamma \to 2^{Q \times \Gamma \times \{E,D\}}$
  - $Q \times \{<\} \to 2^{Q \times \{<\} \times \{D\}}$
  - $Q \times \{>\} \to 2^{Q \times \{>\} \times \{E\}}$
- Ou seja, os símbolos especiais só podem ser gravados na posição original

Não é conhecido se o não-determinísmo aumenta o poder computacional dos ALLs.

Autômatos Limitados Linearmente

Teorema

Uma linguagem L é uma **LSC** (ou **Tipo 1**), se e somente se, L é decidida por um **ALL**.

Recordando, podemos ter dois resultado ao processar w:

- aceita ✓
- rejeita ✗

Vamos omitir a prova (Hopcroft, 1ª edição, Capítulo 9).

Autômatos Limitados Linearmente

Teorema

Uma linguagem L é uma **LSC** (ou **Tipo 1**), se e somente se, L é decidida por um **ALL**.

Recordando, podemos ter dois resultado ao processar w:

- aceita ✓
- rejeita ✗

Vamos omitir a prova (Hopcroft, 1ª edição, Capítulo 9).

Autômatos Limitados Linearmente

Observação

MTs que decidem (decisores) linguagens do **Tipo 3**, **Tipo 2**, e **Tipo 1**, são na verdade **ALLs**.

É difícil mostrar uma linguagem decidível do **Tipo 0** que não possa ser decidida por um ALL (Spiser, Capítulo 9).

Exemplo

$$L_2 = \{a^n b^n c^n \mid n \geq 0\}$$

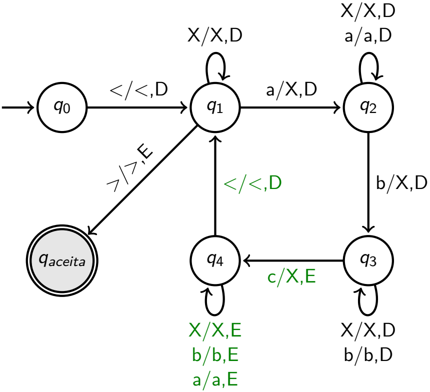
$$< \mathtt{a\ a\ a\ a\ b\ b\ b\ b\ c\ c\ c\ c} > \ldots$$

Descrição de alto-nível do **ALL** A_2:

- "Sobre a cadeia de entrada w:
  1. Mova o cursor para a Direita;
  2. Marque um **a** com **X** e mova o cursor para a Direita, ignorando **a**'s e **X**'s até encontra um **b**;
  3. Marque um **b** com **X** e mova o cursor para a Direita, ignorando **b**'s e **X**'s até encontra um **c**;
  4. Marque um **c** com **X** e mova o cursor para Esquerda até encontrar $<$
  5. Mova o cursor para a Direita;
  6. **Enquanto** houver **X**, mova o cursor para a Direita. Em seguida:
     - Se encontrar **a**, vá para o Passo (2);
     - Se encontrar $>$, aceite a cadeia ✓
     - Senão, rejeite a cadeia ✗

  "

Descrição formal do **ALL** A_2 para $L_2 = \{a^n b^n c^n \mid n \geq 0\}$



$< \texttt{a a a a b b b b c c c c} > \ldots$

Transições implícitas para $q_{rejeita}$ sempre que $\delta(q, a) = \perp$.

Roteiro

Hierarquia de Chomsky

Não computáveis

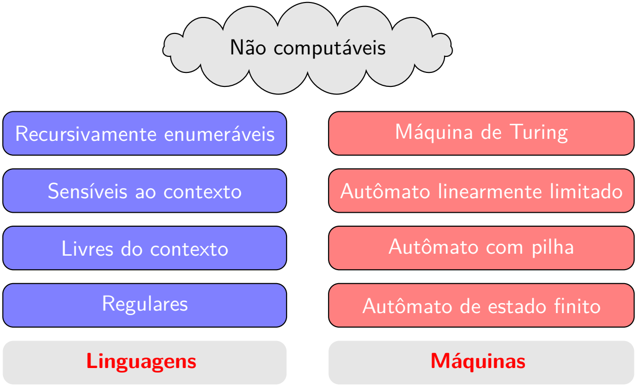
Linguagens	Máquinas
Recursivamente enumeráveis	Máquina de Turing
Sensíveis ao contexto	Autômato linearmente limitado
Livres do contexto	Autômato com pilha
Regulares	Autômato de estado finito

Hierarquia de Chomsky

Não computáveis

Linguagens	Máquinas
Recursivamente enumeráveis	Máquina de Turing
Sensíveis ao contexto	Autômato linearmente limitado
Livres do contexto	Autômato com pilha
Regulares	Autômato de estado finito

Hierarquia de Chomsky



Fim

Dúvidas?

Roteiro

1. Máquinas de Turing
   - Formalização da Máquina de Turing
   - Configuração de uma Máquina de Turing
   - Linguagens Recursivamente Enumeráveis
2. Linguagens Sensíveis ao Contexto
   - Gramáticas Sensíveis ao Contexto
   - Autômatos Limitados Linearmente
   - Hierarquia de Chomsky
3. Referências

Referências

Referências:

1. *"Introdução à Teoria da Computação"* de M. Sipser, 2007.
2. *"Introdução à Teoria de Autômatos, Linguagens e Computação"* de J. E. Hopcroft, R. Motwani, e J. D. Ullman, 2003.
3. *"Linguagens formais e autômatos"* de Paulo F. B. Menezes, 2002.
4. Materiais adaptados dos slides do Prof. Evandro E. S. Ruiz, da USP.