MAC 414
Autômatos, Computabilidade e Complexidade
aula 12 — 26/10/2020

Tempo e espaço

Vai ser importante para Complexidade prestar atenção ao número de passos (comprimento) das computações em MTs .

Tempo e espaço

Vai ser importante para Complexidade prestar atenção ao número de passos (comprimento) das computações em MTs .

Quando escrevemos $C \vdash_M^* C'$, queremos dizer que existe uma sequência de configurações $C \vdash_M C_1 \vdash_M C_2 \cdots \vdash_M C_{n-1} \vdash_M C'$, e nesse caso n é o comprimento (ou *tempo*) da computação.

Tempo e espaço

Vai ser importante para Complexidade prestar atenção ao número de passos (comprimento) das computações em MTs .

Quando escrevemos $C \vdash_M^* C'$, queremos dizer que existe uma sequência de configurações $C \vdash_M C_1 \vdash_M C_2 \cdots \vdash_M C_{n-1} \vdash_M C'$, e nesse caso n é o comprimento (ou *tempo*) da computação.

O tamanho da fita em uma configuração $C = (p, w, u)$ é $\#C = |wu|$.

Tempo e espaço

Vai ser importante para Complexidade prestar atenção ao número de passos (comprimento) das computações em MTs .

Quando escrevemos $C \vdash_M^* C'$, queremos dizer que existe uma sequência de configurações $C \vdash_M C_1 \vdash_M C_2 \cdots \vdash_M C_{n-1} \vdash_M C'$, e nesse caso n é o comprimento (ou *tempo*) da computação.

O tamanho da fita em uma configuração $C = (p, w, u)$ é $\#C = |wu|$.

Cada passo de computação aumenta o comprimento da fita de no máximo 1, logo, no exemplo acima,

$$\#C' \leq \#C + n.$$

Variações e extensões de MTs

Variações e extensões de MTs

MTs são muito primitivas, difíceis de programar.

Precisa melhorar a expressividade.

Variações e extensões de MTs

MTs são muito primitivas, difíceis de programar.
Precisa melhorar a expressividade.

Tema comum:

1. Define-se um novo modelo de computação.

Variações e extensões de MTs

MTs são muito primitivas, difíceis de programar.
Precisa melhorar a expressividade.

Tema comum:

1. Define-se um novo modelo de computação.
2. Mostra-se que dada uma instância do modelo, existe uma MT que faz a mesma coisa

Variações e extensões de MTs

MTs são muito primitivas, difíceis de programar.
Precisa melhorar a expressividade.

Tema comum:

1. Define-se um novo modelo de computação.
2. Mostra-se que dada uma instância do modelo, existe uma MT que faz a mesma coisa

Variações e extensões de MTs

MTs são muito primitivas, difíceis de programar.
Precisa melhorar a expressividade.

Tema comum:

1. Define-se um novo modelo de computação.
2. Mostra-se que dada uma instância do modelo, existe uma MT que faz a mesma coisa (decide ou semi-decide linguagem, computa função)

MT multifita

Idéia: k fitas, com respectivas cabeças de leitura, que se movem independentemente. Transições levam em conta o conteúdo de todas as cabeças.

MT multifita

Idéia: k fitas, com respectivas cabeças de leitura, que se movem independentemente. Transições levam em conta o conteúdo de todas as cabeças.

$k \geq 1$ inteiro. Uma máquina de Turing com k fitas é uma quíntupla $(K, \Sigma, \delta, s, H)$, onde K, Σ, s, H são como em MT comum e

$\delta : K \setminus H \times (\Sigma \cup \{\triangleright\})^k \to K \times (\Sigma \cup \{\leftarrow, \rightarrow\})^k$, com as devidas restrições para $\triangleright$.

MT multifita

Idéia: k fitas, com respectivas cabeças de leitura, que se movem independentemente. Transições levam em conta o conteúdo de todas as cabeças.

$k \geq 1$ inteiro. Uma máquina de Turing com k fitas é uma quíntupla $(K, \Sigma, \delta, s, H)$, onde K,Σ,s,H são como em MT comum e
$\delta : K \setminus H \times (\Sigma \cup \{\triangleright\})^k \to K \times (\Sigma \cup \{\leftarrow, \rightarrow\})^k$, com as devidas restrições para $\triangleright$.

Uma configuração de M é um elemento de

$$K \times (\triangleright\Sigma^* \times (\Sigma^* \setminus \Sigma^* \sqcup))^k.$$

isto é: (estado, fita 1, fita 2,..., fita k)

Computação

Computação

$\vdash_M$: Considere uma configuração $(p, w_1\underline{a}_1u_1, w_2\underline{a}_2u_2, \ldots, w_k\underline{a}_ku_k)$ e suponha que $\delta(p, a_1, a_2, \ldots, a_k) = (q, b_1, b_2, \ldots, b_k)$. Então, o conteúdo da fita j é alterado como se fosse uma MT em que $\delta(p, a_j) = (q, b_j)$.

Computação

$\vdash_M$: Considere uma configuração $(p, w_1\underline{a}_1u_1, w_2\underline{a}_2u_2, \ldots, w_k\underline{a}_ku_k)$ e suponha que $\delta(p, a_1, a_2, \ldots, a_k) = (q, b_1, b_2, \ldots, b_k)$. Então, o conteúdo da fita j é alterado como se fosse uma MT em que $\delta(p, a_j) = (q, b_j)$.

Dados para uma máquina com k fitas:

1. Fita 1: como a entrada de uma MT ($\triangleright\underline{\sqcup}x$)

Computação

$\vdash_M$: Considere uma configuração $(p, w_1\underline{a}_1u_1, w_2\underline{a}_2u_2, \ldots, w_k\underline{a}_ku_k)$ e suponha que $\delta(p, a_1, a_2, \ldots, a_k) = (q, b_1, b_2, \ldots, b_k)$. Então, o conteúdo da fita j é alterado como se fosse uma MT em que $\delta(p, a_j) = (q, b_j)$.

Dados para uma máquina com k fitas:

1. Fita 1: como a entrada de uma MT ($\triangleright\underline{\sqcup}x$)
2. Outras fitas em branco ($\triangleright\underline{\sqcup}$)

Computação

$\vdash_M$: Considere uma configuração $(p, w_1\underline{a}_1u_1, w_2\underline{a}_2u_2, \ldots, w_k\underline{a}_ku_k)$ e suponha que $\delta(p, a_1, a_2, \ldots, a_k) = (q, b_1, b_2, \ldots, b_k)$. Então, o conteúdo da fita j é alterado como se fosse uma MT em que $\delta(p, a_j) = (q, b_j)$.

Dados para uma máquina com k fitas:

1. Fita 1: como a entrada de uma MT ($\triangleright\underline{\sqcup}x$)
2. Outras fitas em branco ($\triangleright\underline{\sqcup}$)

Computação

$\vdash_M$: Considere uma configuração $(p, w_1\underline{a}_1u_1, w_2\underline{a}_2u_2, \ldots, w_k\underline{a}_ku_k)$ e suponha que $\delta(p, a_1, a_2, \ldots, a_k) = (q, b_1, b_2, \ldots, b_k)$. Então, o conteúdo da fita j é alterado como se fosse uma MT em que $\delta(p, a_j) = (q, b_j)$.

Dados para uma máquina com k fitas:

1. Fita 1: como a entrada de uma MT ($\triangleright\underline{\sqcup}x$)
2. Outras fitas em branco ($\triangleright\underline{\sqcup}$)

Decidir e *semidecidir* linguagem, e *computar função* são definidas analogamente a MT.

Exemplo: soma

Dados: $x;y$, representaçoes binárias de naturais
x representa $(x)_2 = \sum_{i=1}^{n} 2^{i-1} x_i$.
Calcular: z tal que $(z)_2 = (x)_2 + (y)_2$.

Exemplo: soma

Dados: $x;y$, representaçoes binárias de naturais
x representa $(x)_2 = \sum_{i=1}^{n} 2^{i-1} x_i$.
Calcular: z tal que $(z)_2 = (x)_2 + (y)_2$.
Máquina de duas fitas:

1. Avançar na fita 1 até achar ; $\quad R_{;}^{1}$

Exemplo: soma

Dados: $x;y$, representaçoes binárias de naturais
x representa $(x)_2 = \sum_{i=1}^{n} 2^{i-1} x_i$.
Calcular: z tal que $(z)_2 = (x)_2 + (y)_2$.
Máquina de duas fitas:

1. Avançar na fita 1 até achar ; R^1
2. Apagar ; e avançar nas duas $\sqcup^1 R^{1,2}$

Exemplo: soma

Dados: $x;y$, representaçoes binárias de naturais

x representa $(x)_2 = \sum_{i=1}^{n} 2^{i-1} x_i$.

Calcular: z tal que $(z)_2 = (x)_2 + (y)_2$.

Máquina de duas fitas:

1. Avançar na fita 1 até achar ; $\quad R^1_{,}$
2. Apagar ; e avançar nas duas $\quad \sqcup^1 R^{1,2}$
3. Loop; cada caractere da 1 é copiado na 2 e apagado; para em $\sqcup$ $\quad \xrightarrow{a \neq \sqcup} \sqcup^1 a^2 R^{1,2} \xrightarrow{a \neq \sqcup}$

Exemplo: soma

Dados: $x;y$, representaçoes binárias de naturais

x representa $(x)_2 = \sum_{i=1}^{n} 2^{i-1} x_i$.

Calcular: z tal que $(z)_2 = (x)_2 + (y)_2$.

Máquina de duas fitas:

1. Avançar na fita 1 até achar ; $\quad R_{;}^{1}$
2. Apagar ; e avançar nas duas $\quad \sqcup^{1} R^{1,2}$
3. Loop; cada caractere da 1 é copiado na 2 e apagado; para em $\sqcup$ $\quad \xrightarrow{a \neq \sqcup} \sqcup^{1} a^{2} R^{1,2} \xrightarrow{a \neq \sqcup} {}^{\curvearrowleft}$
4. Volte para o início das duas fitas $\quad L^{1,2}_{\sqcup^{1,2}}$

Exemplo: soma

Dados: $x;y$, representaçoes binárias de naturais

x representa $(x)_2 = \sum_{i=1}^{n} 2^{i-1} x_i$.

Calcular: z tal que $(z)_2 = (x)_2 + (y)_2$.

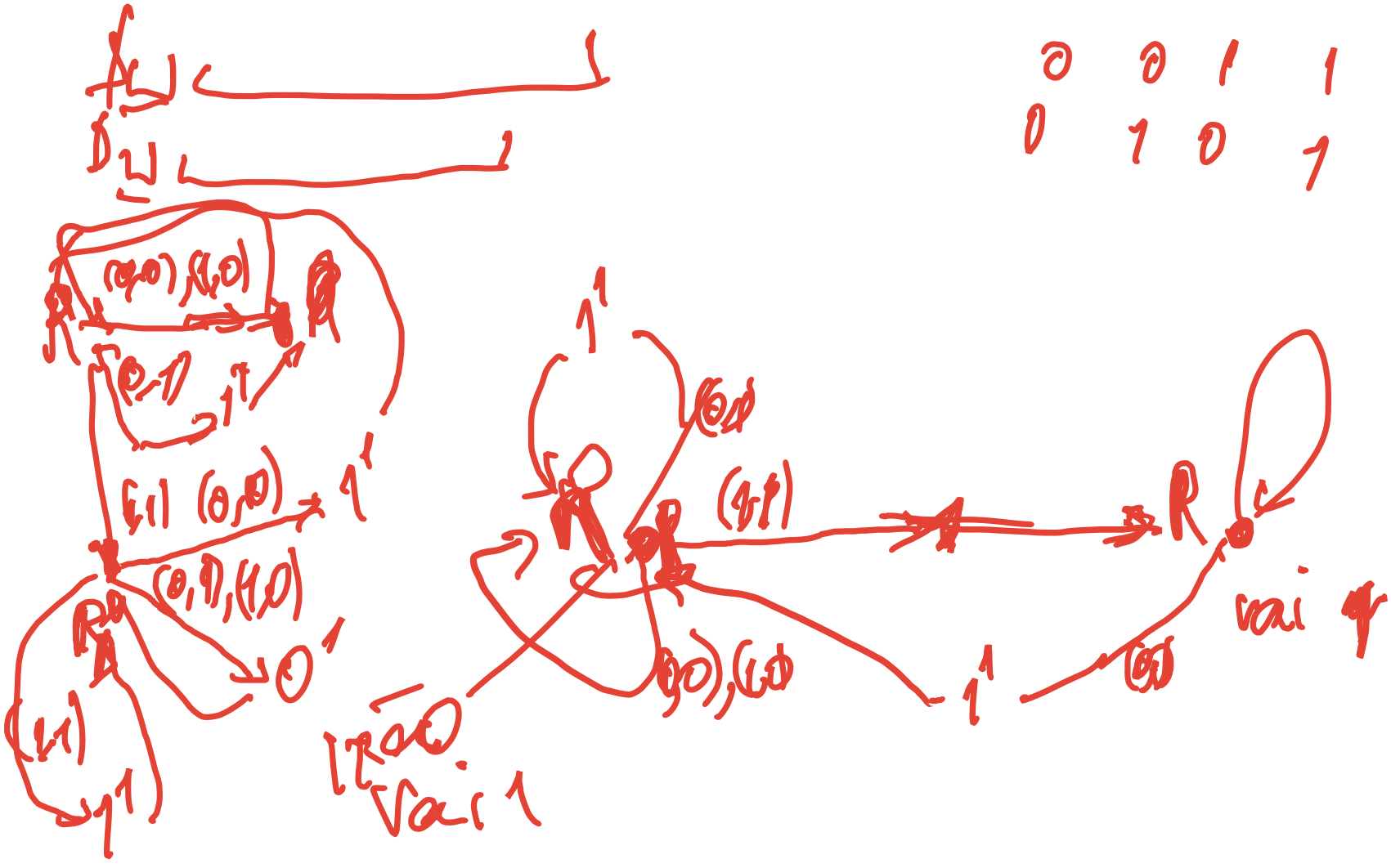
Máquina de duas fitas:

1. Avançar na fita 1 até achar ; $\quad R_{;}^{1}$
2. Apagar ; e avançar nas duas $\quad \sqcup^{1} R^{1,2}$
3. Loop; cada caractere da 1 é copiado na 2 e apagado; para em $\sqcup$ $\quad \overset{a\neq\sqcup}{\longrightarrow} \sqcup^{1} a^{2} R^{1,2} \overset{a\neq\sqcup}{\longrightarrow}{}^{\curvearrowleft}$
4. Volte para o início das duas fitas $\quad L^{1,2}_{\sqcup^{1,2}}$
5. Use o algoritmo escolar para somar os dois números; estado lembra se houve "vai um"

Exemplo: soma

Dados: $x;y$, representaçoes binárias de naturais
x representa $(x)_2 = \sum_{i=1}^{n} 2^{i-1} x_i$.
Calcular: z tal que $(z)_2 = (x)_2 + (y)_2$.
Máquina de duas fitas:

1. Avançar na fita 1 até achar ; $\quad R_{;}^{1}$
2. Apagar ; e avançar nas duas $\quad \sqcup^{1} R^{1,2}$
3. Loop; cada caractere da 1 é copiado na 2 e apagado; para em $\sqcup$ $\quad \xrightarrow{a \neq \sqcup} \sqcup^{1} a^{2} R^{1,2} \xrightarrow{a \neq \sqcup} {}^{\curvearrowleft}$
4. Volte para o início das duas fitas $\quad L^{1,2}_{\sqcup^{1,2}}$
5. Use o algoritmo escolar para somar os dois números; estado lembra se houve "vai um"
6. Volte ao início da fita 1. $\quad L^{1}_{\sqcup}$



A equivalência

Teorema

Com k fixo, seja $M = (K, \Sigma, \delta, s, H)$ uma máquina de Turing com k fitas. Então existe uma MT $M' = (K', \Sigma', \delta', s', H)$, com $\Sigma \subset \Sigma'$, tal que para toda entrada x para M:

A equivalência

Teorema

Com k fixo, seja $M = (K, \Sigma, \delta, s, H)$ uma máquina de Turing com k fitas. Então existe uma MT $M' = (K', \Sigma', \delta', s', H)$, com $\Sigma \subset \Sigma'$, tal que para toda entrada x para M:

1. *Se $(M, x))$ para em t passos, então (M', x) para, em $\mathcal{O}(t(|x| + t))$ passos (no mesmo estado).*

A equivalência

Teorema

Com k fixo, seja $M = (K, \Sigma, \delta, s, H)$ uma máquina de Turing com k fitas. Então existe uma MT $M' = (K', \Sigma', \delta', s', H)$, com $\Sigma \subset \Sigma'$, tal que para toda entrada x para M:

1. *Se $(M, x))$ para em t passos, então (M', x) para, em $\mathcal{O}(t(|x| + t))$ passos (no mesmo estado).*
2. *Se $M(x)$ está definido, então $M'(x)$ também está e $M'(x) = M(x)$.*

A equivalência

Teorema

Com k fixo, seja $M = (K, \Sigma, \delta, s, H)$ uma máquina de Turing com k fitas. Então existe uma MT $M' = (K', \Sigma', \delta', s', H)$, com $\Sigma \subset \Sigma'$, tal que para toda entrada x para M:

1. *Se $(M, x))$ para em t passos, então (M', x) para, em $\mathcal{O}(t(|x| + t))$ passos (no mesmo estado).*
2. *Se $M(x)$ está definido, então $M'(x)$ também está e $M'(x) = M(x)$.*

A equivalência

Teorema

Com k fixo, seja $M = (K, \Sigma, \delta, s, H)$ uma máquina de Turing com k fitas. Então existe uma MT $M' = (K', \Sigma', \delta', s', H)$, com $\Sigma \subset \Sigma'$, tal que para toda entrada x para M:

1. *Se $(M, x))$ para em t passos, então (M', x) para, em $\mathcal{O}(t(|x| + t))$ passos (no mesmo estado).*
2. *Se $M(x)$ está definido, então $M'(x)$ também está e $M'(x) = M(x)$.*

Obs: a constante no $\mathcal{O}$ é exponencial em k.

A simulação

Idéia: posição r na fita de M' codifica as de M

A simulação

Idéia: posição r na fita de M' codifica as de M

Letras: alfabeto "contém" Σ^k.

A simulação

Idéia: posição r na fita de M' codifica as de M

Letras: alfabeto "contém" Σ^k.

Cursor está ou não: alfabeto "contém" $\{0,1\}^k$.

A simulação

Idéia: posição r na fita de M' codifica as de M

Letras: alfabeto "contém" Σ^k.

Cursor está ou não: alfabeto "contém" $\{0,1\}^k$.

Melhor: $\Sigma' \supseteq \Sigma \cup (\Sigma \times \{0,1\})^k$

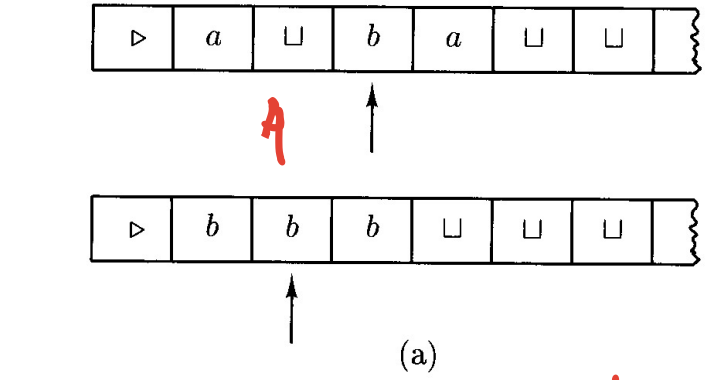
A simulação

Idéia: posição r na fita de M' codifica as de M

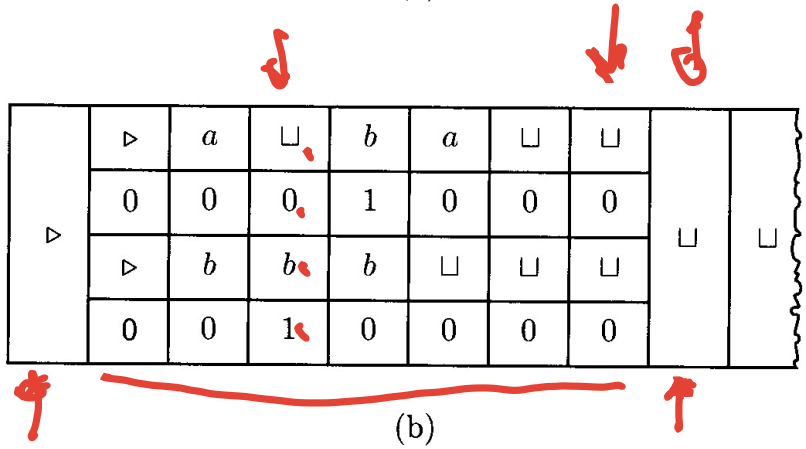
Letras: alfabeto "contém" Σ^k.

Cursor está ou não: alfabeto "contém" $\{0,1\}^k$.

Melhor: $\Sigma' \supseteq \Sigma \cup (\Sigma \times \{0,1\})^k$



(a)



(b)

Fase 1: recodificação

Exemplo com $(M', abaa)$:

1. Aplica $S_{\rightarrow}$: $\triangleright\underline{\sqcup}abaa \mapsto \triangleright\sqcup\underline{\sqcup}abaa$.

Fase 1: recodificação

Exemplo com $(M', abaa)$:

1. Aplica $S_{\rightarrow}$: $\triangleright\sqcup abaa \mapsto \triangleright\sqcup\sqcup abaa$.
2. $L\,(\triangleright, 0, \triangleright, 0, \ldots, \triangleright, 0)\,R\,(\sqcup, 1, \sqcup, 1, \ldots, \sqcup, 1)$

Fase 1: recodificação

Exemplo com $(M', abaa)$:

1. Aplica $S_{\rightarrow}$: $\triangleright\underline{\sqcup}abaa \mapsto \triangleright\sqcup\underline{\sqcup}abaa$.
2. $L\,(\triangleright, 0, \triangleright, 0, \ldots, \triangleright, 0)\,R\,(\sqcup, 1, \sqcup, 1, \ldots, \sqcup, 1)$
3. Loop: $R \xrightarrow{a \neq \sqcup} (a, 0, \sqcup, 0, \ldots, \sqcup, 0)$

Fase 1: recodificação

Exemplo com $(M', abaa)$:

1. Aplica $S_{\rightarrow}$: $\triangleright\underline{\sqcup}abaa \mapsto \triangleright\sqcup\underline{\sqcup}abaa$.
2. $L\,(\triangleright, 0, \triangleright, 0, \ldots, \triangleright, 0)\,R\,(\sqcup, 1, \sqcup, 1, \ldots, \sqcup, 1)$
3. Loop: $R \xrightarrow{a \neq \sqcup} (a, 0, \sqcup, 0, \ldots, \sqcup, 0)$

Fase 1: recodificação

Exemplo com $(M', abaa)$:

1. Aplica $S_{\rightarrow}$: $\triangleright\underline{\sqcup}abaa \mapsto \triangleright\sqcup\underline{\sqcup}abaa$.
2. $L\,(\triangleright, 0, \triangleright, 0, \ldots, \triangleright, 0)\,R\,(\sqcup, 1, \sqcup, 1, \ldots, \sqcup, 1)$
3. Loop: $R \xrightarrow{a \neq \sqcup} (a, 0, \sqcup, 0, \ldots, \sqcup, 0)$

Agora a fita de M' codifica as de M
e o cursor está no fim da fita.

Fase 2: o passo

Os nomes dos estados aqui contem um estado de M, uma k-upla de $\Sigma \cup \{\leftarrow, \rightarrow\}$, e mais alguma coisa.

1. Começa com o cursor à direita da fita, lembrando de um estado q de M.

Fase 2: o passo

Os nomes dos estados aqui contem um estado de M, uma k-upla de $\Sigma \cup \{\leftarrow, \rightarrow\}$, e mais alguma coisa.

1. Começa com o cursor à direita da fita, lembrando de um estado q de M.

2. Vá para a esquerda, capturando no estado as letras em cima de cada cursor; aí volte para a direita. O estado lembra $q, a_1, a_2, \ldots, a_k$.

Fase 2: o passo

Os nomes dos estados aqui contem um estado de M, uma k-upla de $\Sigma \cup \{\leftarrow, \rightarrow\}$, e mais alguma coisa.

1. Começa com o cursor à direita da fita, lembrando de um estado q de M.

2. Vá para a esquerda, capturando no estado as letras em cima de cada cursor; aí volte para a direita. O estado lembra $q, a_1, a_2, \ldots, a_k$.
3. Lembre $\delta(q, a_1, a_2, \ldots, a_k) = (p, b_1, b_2, \ldots, b_k)$.

Fase 2: o passo

Os nomes dos estados aqui contem um estado de M, uma k-upla de $\Sigma \cup \{\leftarrow, \rightarrow\}$, e mais alguma coisa.

1. Começa com o cursor à direita da fita, lembrando de um estado q de M.

→ $\delta(q, \sqcup) = (q, (\sqcup, 0)^k)$

2. Vá para a esquerda, capturando no estado as letras em cima de cada cursor; aí volte para a direita. O estado lembra $q, a_1, a_2, \ldots, a_k$.
3. Lembre $\delta(q, a_1, a_2, \ldots, a_k) = (p, b_1, b_2, \ldots, b_k)$.
4. Vá para a esquerda, aplicando as instruções b_i. Cuidado se tiver que aumentar alguma fita.

Fase 2: o passo

Os nomes dos estados aqui contem um estado de M, uma k-upla de $\Sigma \cup \{\leftarrow, \rightarrow\}$, e mais alguma coisa.

1. Começa com o cursor à direita da fita, lembrando de um estado q de M.
   $\delta(q, \sqcup) = (q, (\sqcup, 0)^k)$
2. Vá para a esquerda, capturando no estado as letras em cima de cada cursor; aí volte para a direita. O estado lembra $q, a_1, a_2, \ldots, a_k$.
3. Lembre $\delta(q, a_1, a_2, \ldots, a_k) = (p, b_1, b_2, \ldots, b_k)$.
4. Vá para a esquerda, aplicando as instruções b_i. Cuidado se tiver que aumentar alguma fita.
5. Volte para a direita.

Fase 3: saída

Quando M entra num estado de parada, volte e recodifique, mantendo só o conteúdo e cursor da fita 1. Aí para.

O tempo

- Fase 1: $\mathcal{O}(|x|)$

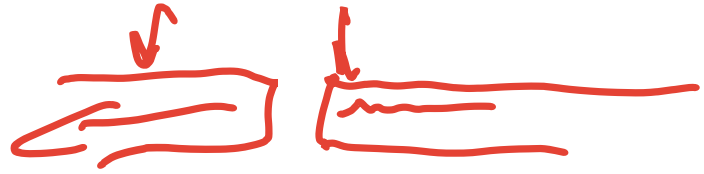
O tempo

- Fase 1: $\mathcal{O}(|x|)$
- Fase 2: percorre duas vezes a fita. Com m passos, $\mathit{comprimento} \leq |x| + m \leq |x| + t$.
  t passos $\mapsto \mathcal{O}(t(|x| + t))$

O tempo

- Fase 1: $\mathcal{O}(|x|)$
- Fase 2: percorre duas vezes a fita. Com m passos, $\mathit{comprimento} \leq |x| + m \leq |x| + t$.
  t passos $\mapsto \mathcal{O}(t(|x| + t))$
- Fase 3: menos que um passo da fase 2.

Outras doidices

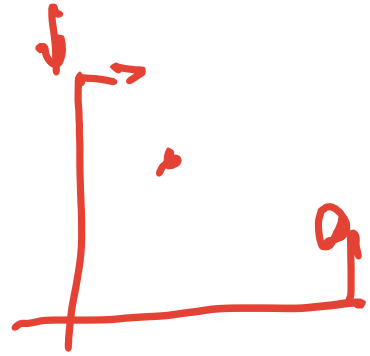


- Fita duplamente infinita (Turing original).

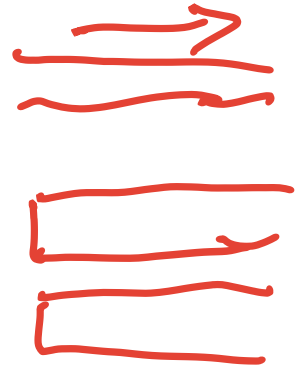
Outras doidices

- Fita duplamente infinita (Turing original).
- Várias cabeças de leitura.

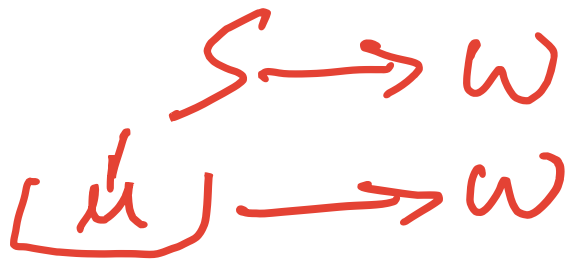
Outras doidices



- Fita duplamente infinita (Turing original).
- Várias cabeças de leitura.
- Fita bidimensional.



Outras doidices



- Fita duplamente infinita (Turing original).
- Várias cabeças de leitura.
- Fita bidimensional.
- Máquina de acesso aleatório.

MT não determinística

MT não determinística

$M = (K, \Sigma, \Delta, s, H)$. A diferença com MT é que $\Delta \subseteq (K \setminus H \times \Sigma \cup \{\triangleright\}) \times (K \times \Sigma \cup \{\leftarrow, \rightarrow\})$.

Configuraçoes, $\vdash_M$, $\vdash_M^*$ como antes.

MT não determinística

$M = (K, \Sigma, \Delta, s, H)$. A diferença com MT é que $\Delta \subseteq (K \setminus H \times \Sigma \cup \{\triangleright\}) \times (K \times \Sigma \cup \{\leftarrow, \rightarrow\})$.

Configuraçoes, $\vdash_M$, $\vdash_M^*$ como antes.

A partir de uma configuração, existe uma árvore de computações, que pode ter caminhos infinitos.

Nós: configurações.

Filho dada por $\vdash_M$ (transição pode rotular aresta).

Folhas: ou o estado está em H, ou simplesmente não tem o que fazer.

MT não determinística

$M = (K, \Sigma, \Delta, s, H)$. A diferença com MT é que $\Delta \subseteq (K \setminus H \times \Sigma \cup \{\triangleright\}) \times (K \times \Sigma \cup \{\leftarrow, \rightarrow\})$.
Configuraçoes, $\vdash_M$, $\vdash_M^*$ como antes.

A partir de uma configuração, existe uma árvore de computações, que pode ter caminhos infinitos.
Nós: configurações.
Filho dada por $\vdash_M$ (transição pode rotular aresta).
Folhas: ou o estado está em H, ou simplesmente não tem o que fazer.

M aceita w se alguma computação de (M, w) para em algum estado de H.

MT não determinística

$M = (K, \Sigma, \Delta, s, H)$. A diferença com MT é que $\Delta \subseteq (K \setminus H \times \Sigma \cup \{\triangleright\}) \times (K \times \Sigma \cup \{\leftarrow, \rightarrow\})$.
Configuraçoes, $\vdash_M$, $\vdash_M^*$ como antes.

A partir de uma configuração, existe uma árvore de computações, que pode ter caminhos infinitos.
Nós: configurações.
Filho dada por $\vdash_M$ (transição pode rotular aresta).
Folhas: ou o estado está em H, ou simplesmente não tem o que fazer.

M aceita w se alguma computação de (M, w) para em algum estado de H.

M semidecide a linguagem $\{w \mid M \text{ aceita } w\}$.

Computando funções e linguagens

Computando funções e linguagens

Uma MTND M termina sempre se para qualquer entrada x, a árvore de computação de (M, x) é finita.

Computando funções e linguagens

Uma MTND M termina sempre se para qualquer entrada x, a árvore de computação de (M, x) é finita.

Uma MTND decide uma linguagem L se ela termina sempre e semidecide L.

Computando funções e linguagens

Uma MTND M termina sempre se para qualquer entrada x, a árvore de computação de (M, x) é finita.

Uma MTND decide uma linguagem L se ela termina sempre e semidecide L.

Uma MTND M computa uma função f sse ela termina sempre e

$$(s, \triangleright \underline{\sqcup} w) \vdash_M^* (h, u \underline{\sigma} v) \text{ sse } u\sigma = \triangleright\sqcup \text{ e } v = f(w).$$

MTND ≡ MT

Teorema

Se uma MTND M decide ou semidecide uma linguagem, ou computa uma função, então existe uma MT M' que faz o mesmo.

MTND ≡ MT

Teorema

Se uma MTND M decide ou semidecide uma linguagem, ou computa uma função, então existe uma MT M' que faz o mesmo.

MTND ≡ MT

Teorema

Se uma MTND M decide ou semidecide uma linguagem, ou computa uma função, então existe uma MT M' que faz o mesmo.

Dem: Para cada entrada w, M' percorre a árvore de computação de (M, w).

MTND ≡ MT

Teorema

Se uma MTND M decide ou semidecide uma linguagem, ou computa uma função, então existe uma MT M' que faz o mesmo.

Dem: Para cada entrada w, M' percorre a árvore de computação de (M, w). Como podem haver caminhos infinitos, a busca é em largura.

MTND ≡ MT

Teorema

Se uma MTND M decide ou semidecide uma linguagem, ou computa uma função, então existe uma MT M' que faz o mesmo.

Dem: Para cada entrada w, M' percorre a árvore de computação de (M, w). Como podem haver caminhos infinitos, a busca é em largura.

Se encontra um estado de parada de M, para e dá a resposta.

MTND ≡ MT

Teorema

Se uma MTND M decide ou semidecide uma linguagem, ou computa uma função, então existe uma MT M' que faz o mesmo.

Dem: Para cada entrada w, M' percorre a árvore de computação de (M, w). Como podem haver caminhos infinitos, a busca é em largura.

Se encontra um estado de parada de M, para e dá a resposta. Se não:

- Se M computa função, isso não pode acontecer.
- Se M decide, alguma hora a árvore acaba.
- Se (M, w) não para, (M', w) não para.

Detalhes

Detalhes

Numere os elementos de Δ, como $\delta_1, \ldots, \delta_m$. Cada caminho na árvore de computação de (M, w) pode ser representada por uma sequência finita de δ_i's.

Detalhes

Numere os elementos de Δ, como $\delta_1, \ldots, \delta_m$. Cada caminho na árvore de computação de (M, w) pode ser representada por uma sequência finita de δ_i's.

M' tem três fitas

1. Recebe w de entrada, e só é alterada no caso de função, para ter o resultado.

Detalhes

Numere os elementos de Δ, como $\delta_1, \ldots, \delta_m$. Cada caminho na árvore de computação de (M, w) pode ser representada por uma sequência finita de δ_i's.

M' tem três fitas

1. Recebe w de entrada, e só é alterada no caso de função, para ter o resultado.
2. Fita de trabalho, que simula a de M.

Detalhes

Numere os elementos de Δ, como $\delta_1, \ldots, \delta_m$. Cada caminho na árvore de computação de (M, w) pode ser representada por uma sequência finita de δ_i's.

M' tem três fitas

1. Recebe w de entrada, e só é alterada no caso de função, para ter o resultado.
2. Fita de trabalho, que simula a de M.
3. Fita de geração de caminhos - vai conter uma sequência de $\{1, 2, \ldots, m\}$.

Simulação

Enquanto não satisfizer algum critério de parada

Copie a fita 1 na fita 2

Gere a próxima sequência na fita 3

`/% ordem comprimento lex %/`

Efetue na fita 2 a sequência de transições dada na fita 3 até onde der

Massageie a saída, se necessário.

Simulação

Enquanto não satisfizer algum critério de parada

Copie a fita 1 na fita 2

Gere a próxima sequência na fita 3

`/% ordem comprimento lex %/`

Efetue na fita 2 a sequência de transições dada na fita 3 até onde der

Massageie a saída, se necessário.

Isso é claramente exponencial!

Nada a ver, mas...

Nada a ver, mas...

Prop: Se uma árvore de computação é infinita, então existe uma computação infinita.

Nada a ver, mas...

Prop: Se uma árvore de computação é infinita, então existe uma computação infinita.

Lema de König: Se uma árvore infinita enraizada é tal que cada nó tem um número finito de filhos, então ela contém um caminho infinito.