Teoria da Computação

Máquinas de Turing

Simão Melo de Sousa

Computer Science Department
University of Beira Interior, Portugal

Plano

Aviso Prévio

- **A redacção dos apontamentos da disciplina documento baseou-se fortemente na bibliografia indicada. Parece-nos então óbvio que a leitura e a aprendizagem directa pelas obras originais é recomendada, e mesmo essencial à compreensão profunda das noções aqui apresentadas;**
- **O português não é a língua materna do autor e o presente documento encontra-se em fase (constante) de elaboração/melhoramento pelo que se agradece e até se incentiva qualquer sugestão ou correcção;**

Referencias bibliográficas

- (Principal) C. H. Papadimitriou, H. R. Lewis. *Elements of the Theory of Computation* por Prentice Hall, 1997. Tradução brasileira: *Elementos de Teoria da Computação*, 2a Edição. Bookman, Porto Alegre, 2000.
- (Introdutório, em francês - embora deva existir algo em inglês algures) P. Wolper. *Introduction à la calculabilité*, 3ª edição, Dunod, 2006.
- (introdutório e de leitura agradável) P. Linz. *An introduction to formal languages and automata*. Jones and Bartlett Publisher, 2006.
- (Uma obra de referência e muito completo... um "must") John E. Hopcroft, Rajeev Motwani, Jeffrey D. Ullman. *Introduction to Automata Theory, Languages, and Computation* (3nd Edition). Addison Wesley, 2006 (existe em português do Brasil).
- (Abordagem abrangente) M. Sipser. *Introducton to the Theory of Computation*. PWS Publishing, 2006.

Plano

Introdução

- Os autómatos com pilha tem acesso a uma memória potencialmente infinita, mas mesmo assim não conseguem reconhecer linguagens como $a^n b^n c^n$.
- Mas parece simples reconhecer esta linguagem por um processo "baseado em estados":

```
let rec recognize l n m estado =
  match l with
  [] -> estado = 3 && n = 0
  | 'a' :: li -> if estado = 1 then (recognize li (n+1) m 1)
   else false
  | 'b' :: li -> if estado = 1 then (recognize li n 1 2)
                 else if estado = 2 then (recognize li n (m+1) 2)
                      else false
  | 'c' :: li -> if estado = 2
                  then if  m=n then (recognize li (n-1) m 3)
                               else false
```

Introdução

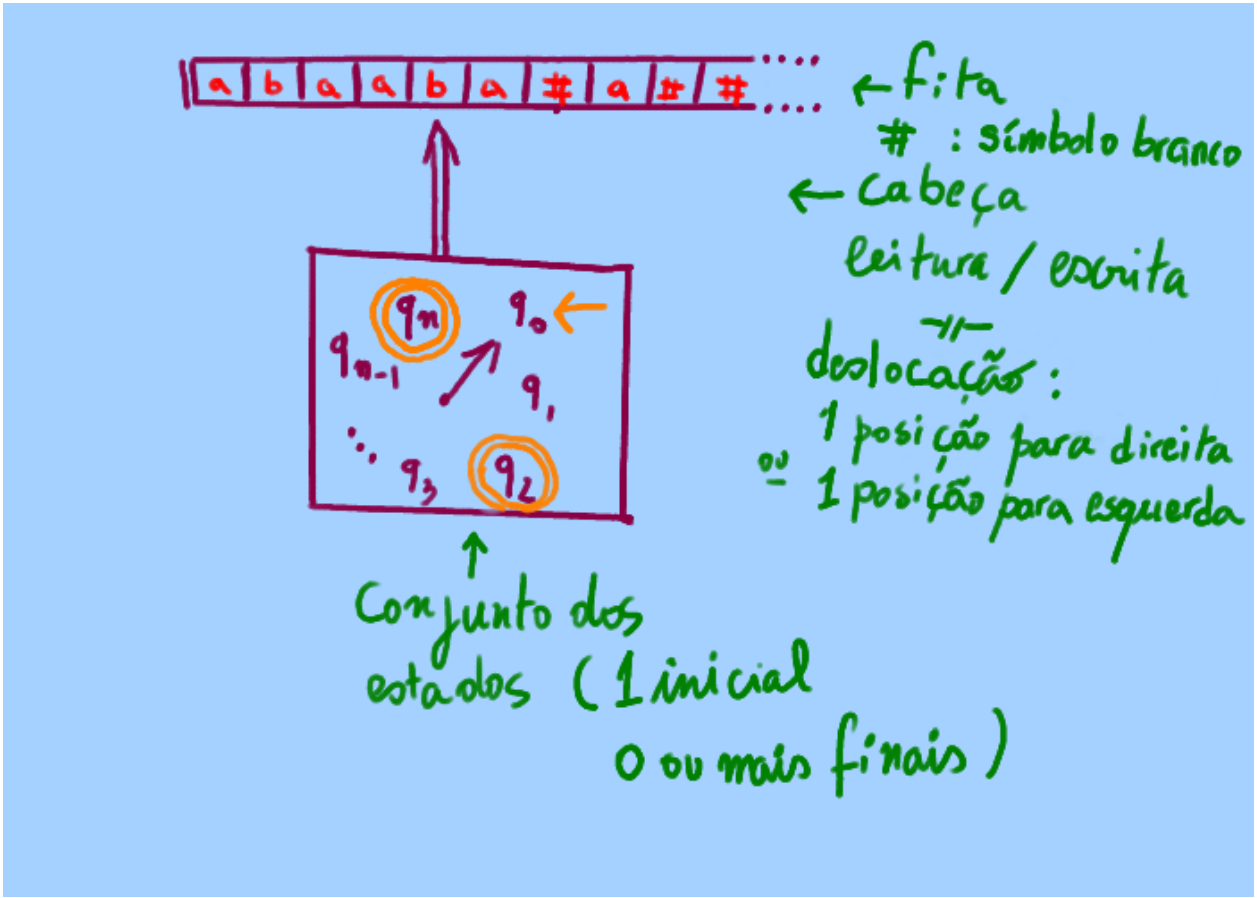
- Limites dos autómatos com pilha: gestão da memória.
- A limitação vem da disciplina LIFO da memória: uma utilização possível por cada memorização.
- O exemplo anterior reutiliza (duas vezes) o valor de 'a's que foram lidos. Um para contabilizar os 'b's e outras para os 'c's.
- As máquinas de Turing permitam libertar-se deste inconveniente.

Plano

Definição

- Vamos aqui nos limitar às máquinas de Turing deterministas (veremos mais adiante que isso não é uma limitação).
- Uma máquina de Turing é uma maquina de estados finitos (alguns deles iniciais ou finais), a semelhança dos autómatos que vimos até aqui.
- A diferença reside no seguinte: a fita de entrada é também de saída. Ou seja dá para ler e para escrever.
- fita = memória. Tem um início (a esquerda), mas não tem fim (infinita a direita).
- Com tal, equipamos a máquina com a possibilidade de se deslocar na fita (para a direita ou para a esquerda, quando possível).

Numa imagem


a b a a b a # a # #
← fita
: símbolo branco
← cabeça
leitura / escrita
deslocação:
1 posição para direita
ou 1 posição para esquerda
q_n
q_0
q_{n-1}
q_1
q_3
q_2
Conjunto dos estados (1 inicial 0 ou mais finais)

As transições

uma transição entre estados (q_1, a, b, d, q_2) denota a seguinte situação:

- Se a máquina se encontra no estado q_1 e
- que a cabeça de leitura aponta para um a
- então escrever b (a é assim substituido por b na fita),
- deslocar a cabeça para a célula seguinte (se d é $\leftarrow$ a célula seguinte é a célula a esquerda, se d é $\rightarrow$ a célula seguinte é a célula a direita)
- e finalmente entrar no estado q_2.

Algumas considerações iniciais

- Vamos diferenciar o alfabeto de entrada Σ do alfabeto da fita Γ (de facto $\Sigma \subset \Gamma$): os elementos de Σ forma a palavra (finita) que inicialmente está no inicio da fita.
- Uma máquina não pode aceder a posições que estão a esquerda do início da fita.
- Como a fita é infinita (a direita) as restante posições contém um *símbolo branco* notado $\sharp$.
- A máquina tem a possibilidade de escrever na fita os símbolos de Σ mais, eventualmente, outros. Estes símbolos formam Γ.
- Neste sentido uma máquina de Turing é uma máquina com output (a palavra contida na fita).

Algumas considerações iniciais

- Dado um momento da execução da máquina só uma quantidade finita de células da fita foram exploradas.
- Ou seja, há sempre uma quantidade finita de fita ocupada por letras. Dada uma posição da cabeça de leitura na fita há uma quantidade finita de letras a esquerda e uma quantidade finita de letras a direita até uma posição em que a direita só há símbolos $\sharp$.
- As deslocações na fita são representadas pelos símbolos (L, R) ou $(\leftarrow, \rightarrow)$
- Vamos considerar que não há transições que saem dos estados finais. Embora possa parecer uma restrição, é possível transformar qualquer máquina de Turing em máquinas de Turing com esta propriedade.
- Uma palavra é aceite quando a execução levar o seu reconhecimento pára um estado final.

Definição Formal

Definition (Máquina de Turing)

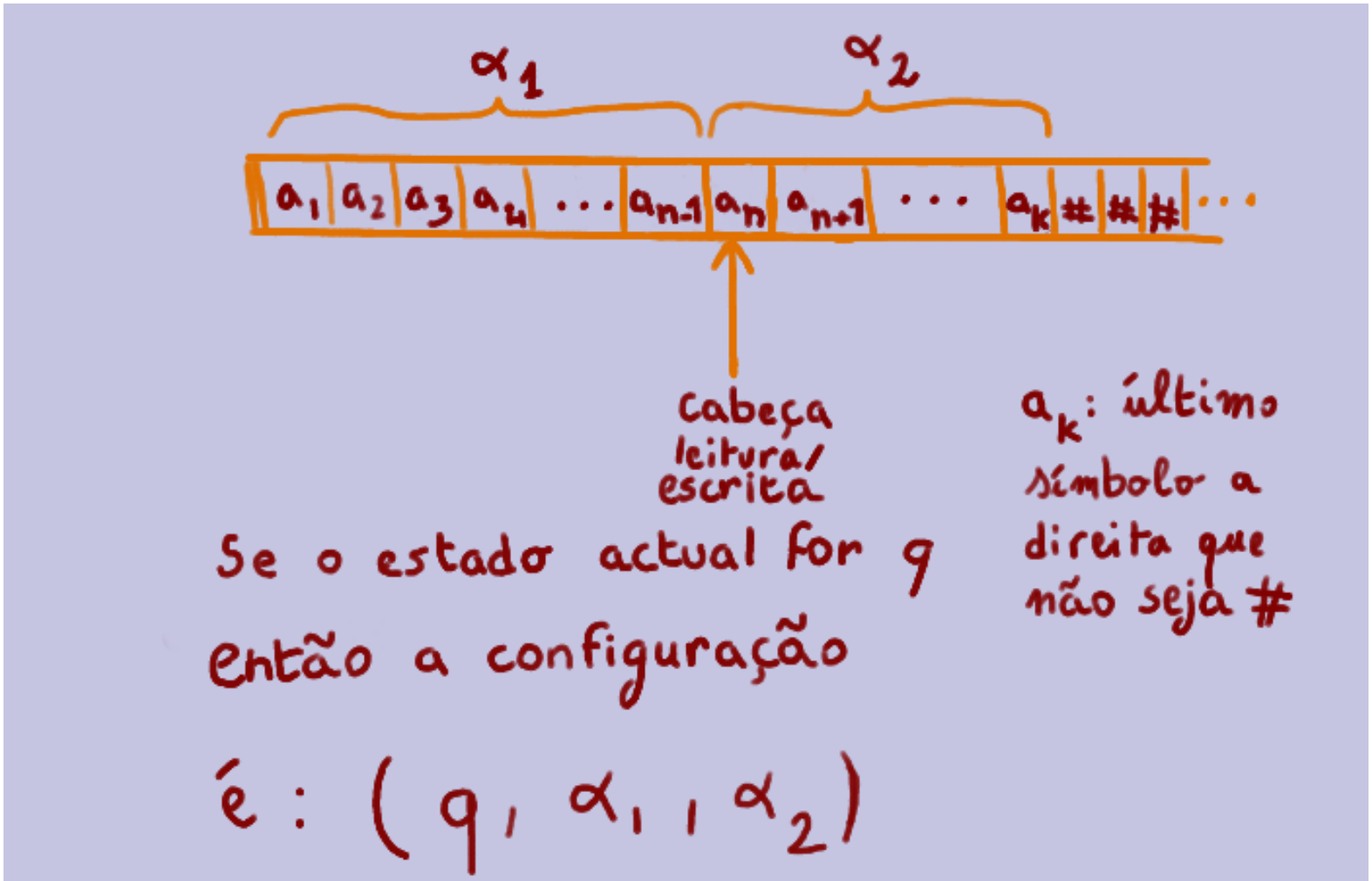
Uma máquina de Turing é um 7-tuplo $(Q, \Gamma, \Sigma, \delta, s, B, F)$ onde

- Q é um conjunto de finito de estados
- Γ é o alfabeto de fita
- Σ ($\subseteq \Gamma$) é o alfabeto de entrada (alfabeto utilizada para as palavras inicialmente colocados na fita)
- $s \in Q$ é o estado inicial
- F ($\subseteq Q$) é o conjunto dos estado finais
- $B \in \Gamma - \Sigma$ é o *símbolo branco* (notação habitual: $\sharp$)
- $\delta : (Q \times \Gamma) \rightarrow (Q \times \Gamma \times \{L, R\})$ é a função de de transição.

Configurações

- Para formalizar a noção de execução precisamos, como já é habitual, de precisar o conceito de configuração.
- lembrete: configuração = estado preciso e completo da máquina de Turing.
- Aqui: precisamos de saber o estado da fita, o estado em que a máquina se encontra, a posição da cabeça de leitura/escrita.
- Como vimos nos acetatos anteriores num determinado momento temos sempre uma situação semelhante ao gráfico seguinte.

Configurações



Configurações

Definition (Configuração duma máquina de Turing)

Uma configuração é um elemento do conjunto

$$Q \times \Gamma^* \times (\{\epsilon\} \cup \Gamma^*.(\Gamma - \{B\}))$$

Derivações

Definition (Derivação num passo e Derivação generalizada)

- Seja $C = (q, \alpha_1, \alpha_2)$ uma configuração duma máquina M. Vamos assumir que temos sempre a situação $\alpha_2 = b\alpha_2'$. No caso de $\alpha_2 = \epsilon$, tomamos $b = \sharp$. As configurações deriváveis de C são definidas da seguinte forma:
  - Se $\delta(q, b) = (q', b', R)$ então temos $(q, \alpha_1, b\alpha_2') \vdash_M (q', \alpha_1 b', \alpha_2')$
  - Se $\delta(q, b) = (q', b', L)$ (neste caso $\alpha_1 \neq \epsilon$, ou seja $\alpha_1 = \alpha_1' a$) então temos $(q, \alpha_1' a, b\alpha_2') \vdash_M (q', \alpha_1', ab'\alpha_2')$.
  - Uma configuração C' é derivável em vários passos de C pela máquina M (notação $C \vdash_M^* C'$) se existe um $k \in \mathbb{N}, k \geq 0$ e configurações intermédias $C_0, C_1, C_2, \ldots C_k$ tais que
    - $C = C_0$
    - $C' = C_k$
    - $C_i \vdash C_{i+1}$ para $0 \leq i < k$

Palavras e Linguagens Aceites

Uma palavra w é aceite por uma máquina de Turing M se a execução por M começando com fita inicializada com esta palavra e partindo do estado inicial de M termina num estado final.

Definition (Linguagem aceite por uma máquina M)

A linguagem $L(M)$ aceite por uma máquina de Turing M é o conjunto das palavras das palavras aceites w, ou seja, tais que

$$(s, \epsilon, w) \vdash_M^* (p, \alpha_1, \alpha_2) \quad (com\ p \in F)$$

Exemplos

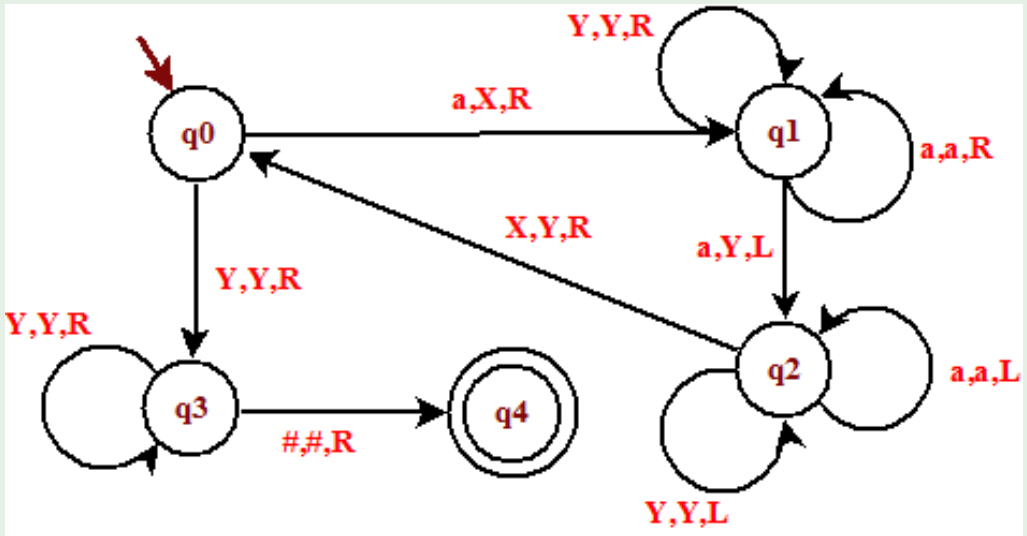
uma máquina

$$M = (Q, \Gamma, \Sigma, \delta, s, B, F)$$

com

- $Q = \{q_0, q_1, q_2, q_3, q_4\}$
- $\Gamma = \{a, b, X, Y, \sharp\}$
- $\Sigma = \{a, b\}$
- $s = q_0$
- $B = \sharp$
- $F = \{q_4\}$
- δ é definido da seguinte forma:

o seu grafo



	a	b	X	Y	$\sharp$
q_0	(q_1, X, R)			(q_3, Y, R)	
q_1	(q_1, a, R)	(q_2, Y, L)	(q_0, X, R)	(q_1, Y, R)	
q_2	(q_2, a, L)			(q_2, Y, L)	
q_3				(q_3, Y, R)	$(q_4, \sharp, R)$
q_4					

Exemplos

uma máquina

$$M = (Q, \Gamma, \Sigma, \delta, s, B, F)$$

com

- $Q = \{q_0, q_1, q_2, q_3, q_4\}$
- $\Gamma = \{a, b, X, Y, \sharp\}$
- $\Sigma = \{a, b\}$
- $s = q_0$
- $B = \sharp$
- $F = \{q_4\}$
- δ é definido da seguinte forma

	a	b	X	Y	$\sharp$
q_0	(q_1, X, R)			(q_3, Y, R)	
q_1	(q_1, a, R)	(q_2, Y, L)		(q_1, Y, R)	
q_2	(q_2, a, L)		(q_0, X, R)	(q_2, Y, L)	
q_3				(q_3, Y, R)	$(q_4, \sharp, R)$
q_4					

a sua execução

$(q_0, \epsilon, aaabbb)$
$(q_1, X, aabbb)$
$(q_1, Xa, abbb)$
(q_1, Xaa, bbb)
$(q_2, Xa, aYbb)$
$(q_2, X, aaYbb)$
$(q_2, \epsilon, XaaYbb)$
$(q_0, X, aaYbb)$
$(q_1, XX, aYbb)$

$\vdots$

$(q_1, XXXYY, b)$
$(q_2, XXXY, YY)$
(q_2, XXX, YYY)
$(q_2, XX, XYYY)$
(q_0, XXX, YYY)
$(q_3, XXXY, YY)$
$(q_3, XXXYY, Y)$
$(q_3, XXXYYY, \epsilon)$
$(q_4, XXXYYY\sharp, \epsilon)$

Plano

Considerações sobre palavras aceites

- Uma máquina de Turing pode entrar em ciclo
- Basta considerar uma máquina com dois estados e as transições seguintes: $q_0 \xrightarrow{a,a,R} q_0$, $q_0 \xrightarrow{\sharp,\sharp,R} q_0$ e $q_0 \xrightarrow{b,b,R} q_1$ onde q_1 é final.
- De facto vários casos podem ocorrer. Consideramos assim a execução a partir da configuração (s, ϵ, w). Assim, a execução gera uma sequência de configurações tal que:
  - Esta termina numa configuração que contém um estado final. Neste caso a palavra w é aceite.
  - Esta termina numa configuração a partir da qual não é possível continuar a derivar
    - porque a função de transição não está definida para esta configuração ou
    - porque a função de transição define uma deslocação para a esquerda e a cabeça de leitura se encontra na posição inicial.
  - Esta nunca passa por um estado final e é infinita.

Considerações sobre palavras aceites

- Nos dois primeiros casos temos, a máquina nos permite determinar com exactidão se uma palavra é ou não aceite.
- no último caso não se pode decidir com toda a certeza. Não é porque a máquina ainda não conseguiu chegar a um estado final num tempo considerado razoável que isto não irá acontecer mais além.
- Assim as máquinas de Turing parecem ir além do que se entende ser uma algoritmo (ver aula de introdução)
- Vamos assim revisitar e precisar a noção de execução e introduzir o conceito de linguagem decidia (diferente da noção de linguagem aceite).

Execução

Definition (Execução)

A execução duma máquina de Turing M numa palavra w é a sequência de configuração $(s, \epsilon, w) \vdash_M C_1 \vdash_M C_2 \vdash_M \ldots \vdash_M C_k \vdash_M \ldots$ máxima, isto é:

- é infinita, ou
- termina numa configuração em que o estado é final ou
- termina numa configuração em que o estado não é final.

Linguagem aceite , linguagem decidida

Estamos assim em medida de revisitar e refinar a noção de linguagem associada a uma máquina de Turing.

Definition (linguagem aceite)

A linguagem $L(M)$ aceite por uma máquina de Turing M é o conjunto das palavras das palavras aceites w, ou seja, tais que

$$(s, \epsilon, w) \vdash_M^* (p, \alpha_1, \alpha_2) \quad (com\ p \in F)$$

Definition (linguagem decidida)

Uma linguagem L é decidida por uma máquina de Turing M se

- M aceita L
- M não tem execução infinitas

Máquinas de Turing sem execuções infinitas (que decidam linguagens) são, finalmente, a nossa formalização de procedimento efectivo!

Linguagem Decidida e Autómatos

- DFA: linguagem aceite=linguagem decidida
- NDFA: qualquer linguagem aceite pode ser aceite por um DFA.
- Autómatos pushdown: os PDAs podem ter execuções infinitas, mas o algoritmo CYK demonstra que é possível decidir linguagens livre de contexto (com um mecanismo mais expressivo que os PDAs: as MTs).
- PDA deterministas (DPDA): não apresentam execuções infinitas, ou seja, linguagem aceite = linguagem decidida.

Linguagens Recursivas e Linguagens Recursivamente Numeráveis

Definition (Linguagens Recursivas e Linguagens Recursivamente Numeráveis)

- Uma linguagem é dita **recursiva** se é decidida por uma máquina de Turing
- Uma linguagem é dita **recursivamente enumerável** se é aceite por uma máquina de Turing

Plano

Uma equação para a posteridade: MT=Algoritmo

- Retomemos agora e finalmente a problemática fundamental a teoria da computação: como podemos definir formalmente a noção de algoritmo? Por uma máquina de Turing.
- (ver aula de apresentação para uma discussão completa (mas informal) sobre esta problemática)

Definição

As linguagens reconhecidas por um procedimento efectivo são exactamente as linguagens decididas por uma máquina de Turing.

ou ainda:

As linguagens reconhecidas por um algoritmo são exactamente as linguagens decididas por X

(em que X denota qualquer modelo da computação equivalente às máquinas de Turing, como por exemplo funções recursivas totais de Kleene, os λ-termos fortemente nomalizáveis do λ-calculo etc...)

Plano

Plano

Plano