# 5. Maquinas de Turing

## 5.1 Introdução

As maquinas de Turing são uma proposta de formalização da noção de procedimento efetivo (programa executado num computador que sempre fornece uma resposta)

A maquina de Turing não é nada mais do que um autômato finito determinístico com uma fita,  onde será escrita a palavra a processar, e uma cabeça de leitura que pode se movimentar tanto a direita quanto a esquerda. Com a cabeça de leitura podemos então ler e escrever símbolos do alfabeto de fita.

## 5.2 Definição

Uma maquina de Turing é formalmente descrita pela tupla de 7 valores:

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

- Q é um conjunto finito de estados

- $\Gamma$ é o alfabeto de fita

- $\Sigma \subseteq \Gamma$ é o alfabeto de entrada

- $s \in Q$ é o estado inicial

- $F \subseteq Q$ é o conjunto dos estados de aceitação

- $B \in \Gamma-\Sigma$ é o símbolo branco (geralmente anotado #)

- $\delta: Q\text{x}\Gamma \rightarrow Q\text{x}\Gamma\text{x}\{L,R\}$ é a função de transição (L=Left e R=Right)

A configuração de uma maquina de Turing é um elemento que pertence a:

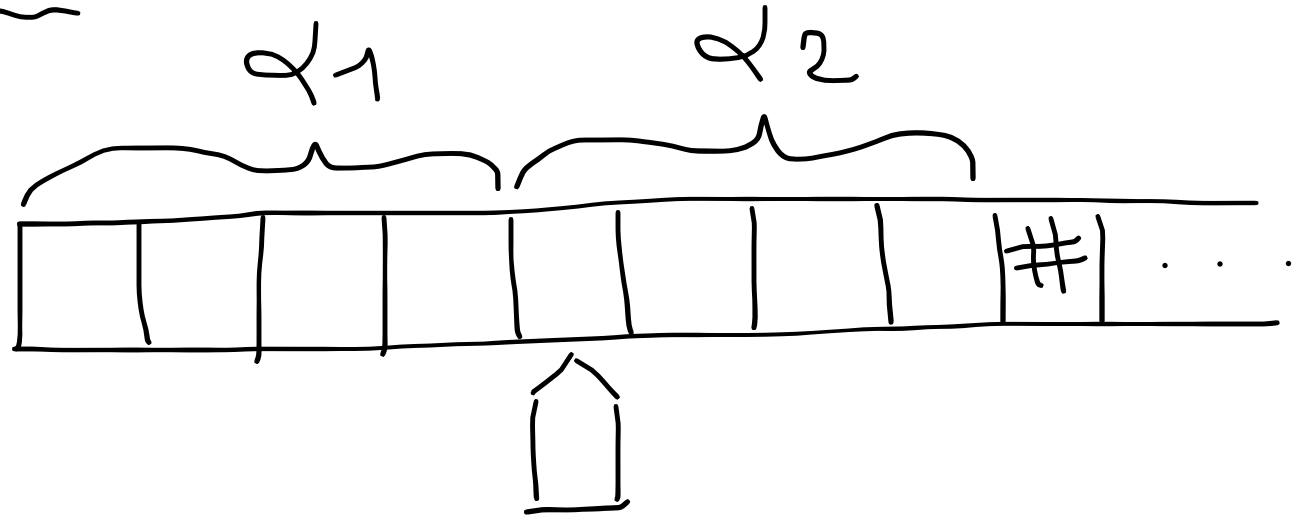
$Q \times \Gamma^* \times (\varepsilon \cup \Gamma^*.(\Gamma - \{B\}))$

onde $\varepsilon \cup \Gamma^*.(\Gamma - \{B\})$ representa a palavra vazia ou o conjunto de todas as palavras que podem ser definidas pelos símbolos do alfabeto $\Gamma$ menos o símbolo branco.

Informalmente, a configuração será dada pelo estado do autômato finito, pela palavra que aparece na fita e pela posição da cabeça de leitura.

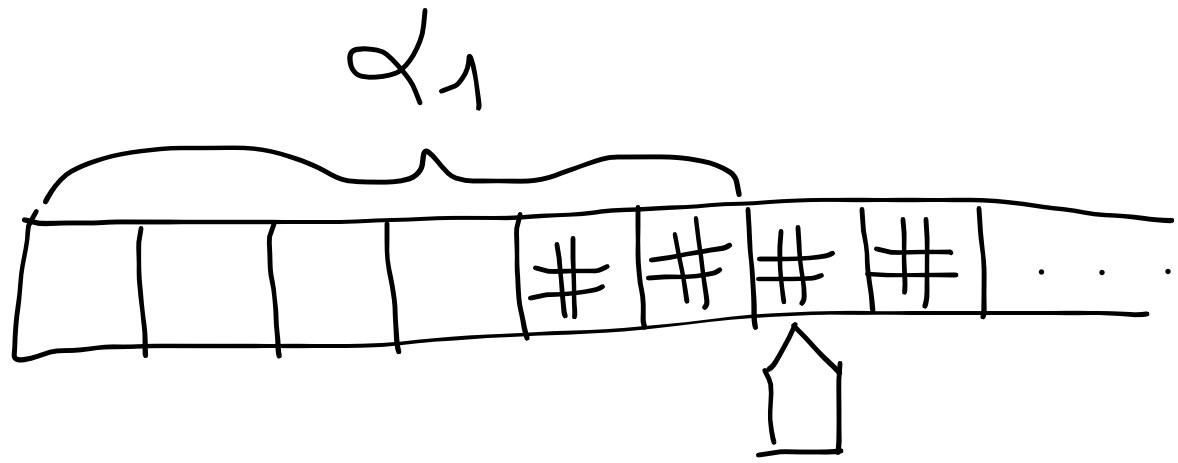
Uma forma de definir a posição da cabeça de leitura é de considerar a palavra que aparece antes da cabeça de leitura e a palavra que se encontra entre a posição da cabeça e o último símbolo não branco na fita.

# Ex 1




$$CONFIG = (q, \alpha_1, \alpha_2)$$

# Ex 2:




$$CONFIG = (q, \alpha_1, \varepsilon)$$

Definição : uma configuração C' é derivável em variais etapas a partir da configuração C pela maquina M se existe $k \geq 0$ e as configurações intermediárias C0, C1, ... , Ck tais que:

. C = C0
. C' = Ck
. Ci |-M Ci+1 para $0 \leq i < k$

Uma palavra é então aceita por uma maquina de Turing se a execução da maquina leva uma configuração cujo estado é de aceitação.

Definição : A linguagem L(M) aceita por uma maquina de Turing é o conjunto de palavras w tais que :

$$(S, \varepsilon, w) \vdash_M^* (p, \alpha_1, \alpha_2)$$

com $p \in F$

Exercício: Construir a maquina que aceita a linguagem a(n)b(n) para n > 0. Testar a maquina com a palavra de entrada aaabbb

Dica: $\Gamma = \{a , b , X , Y , \#\}$ e $\Sigma = \{a , b\}$

## 5.3 Linguagens aceitas e decididas

Consideremos as configurações derivadas a partir de uma configuração qualquer (s,ε,w). Varias situações podem ocorrer:

1) na seqüência de configurações aparece um estado de aceitação e a palavra é então aceita;

2) a seqüência de configuração atinge um estado de parada (função de transição não definida para a configuração atingida) e a palavra não é aceita;

3) a seqüência de configuração não passa por um estado de aceitação e é infinita

Podemos dizer que uma maquina de Turing que aceita uma linguagem não define um procedimento efetivo: teremos a garantia neste caso que a maquina sempre parará para as palavras aceitas. Eventualmente, para certas palavras não aceitas, a maquina poderá produzir uma seqüência de configuração infinita e não existe um procedimento que identifica as sequências infinitas...

Definição: a execução de uma maquina de Turing numa palavra w é a seqüência de configurações :

$$(s, \varepsilon, w) \vdash_m (q_1 \vdash_m \cdots \vdash_m (q_k \vdash_m \cdots$$

máxima tal que:

- ela seja infinita,

- ou ela atinge um estado de aceitação,

- ou ela atinge um estado de parada (diferente de um estado de aceitação)

Definição: uma linguagem L é decidida por uma maquina de Turing M se:

- M aceita L

- M não tem nenhuma execução infinita.

Definição: uma linguagem é recursiva se ela é decidida por uma maquina de Turing

Definição: uma linguagem é recursivamente enumerável se ela é aceita por uma maquina de Turing

Vocabulário que tem como origem os formalismos computacionais anteriores às Máquinas de Turing (as funções recursivas particularmente)

Podemos de fato definir a noção de procedimento efetivo no contexto das funções recursivas sem utilizar o conceito de Máquina de Turing **!**

## 5.4 Tese de Church-Turing

Tese (hipótese): as linguagem reconhecidas por um procedimento efetivo são as linguagens decididas por uma maquina de Turing.

Uma tese não é um teorema. Ela não pode ser provada!

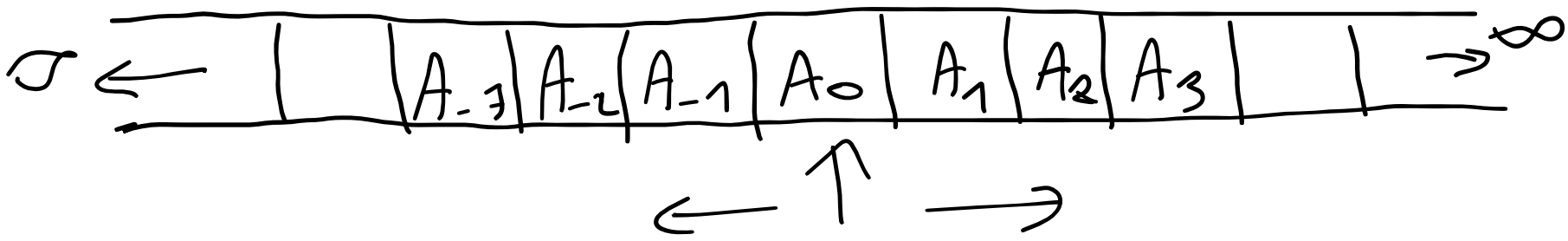
De acordo com a TESE, qualquer programa que pode ser executada num computador pode ser representado então por uma maquina de Turing

É fácil mostrar que qualquer máquina de Turing que decide uma linguagem pode ser simulada num computador (encontrar na web simuladores de Máquinas de Turing)

É mais difícil verificar que qualquer programa pode ser representado por uma máquina de Turing !

Uma abordagem para se convencer que qualquer programa pode ser representado por uma MT é mostrar que qualquer outro modelo computacional sempre será equivalente a uma MT clássica : isso não vai constituir uma prova mas pode ser vista como uma justificativa do uso das MT através de um tipo de experimento.

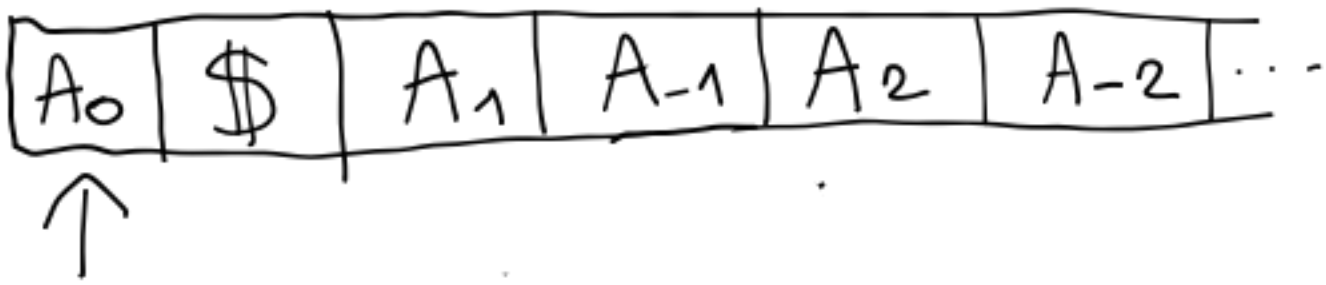
Exemplo 1: Estudo de uma MT de fita infinita a direita e a esquerda.




Parece que se trata de um modelo mais genérico que a MT com finta infinita somente a direita.

Mostrar que toda linguagem aceita ou decidida por este tipo de máquina será também aceita ou decidida por uma MT tradicional !

Prova ? Mostrar que existe um TM tradicional (infinita a direita somente) que consegue simular uma TM 2 $\infty$

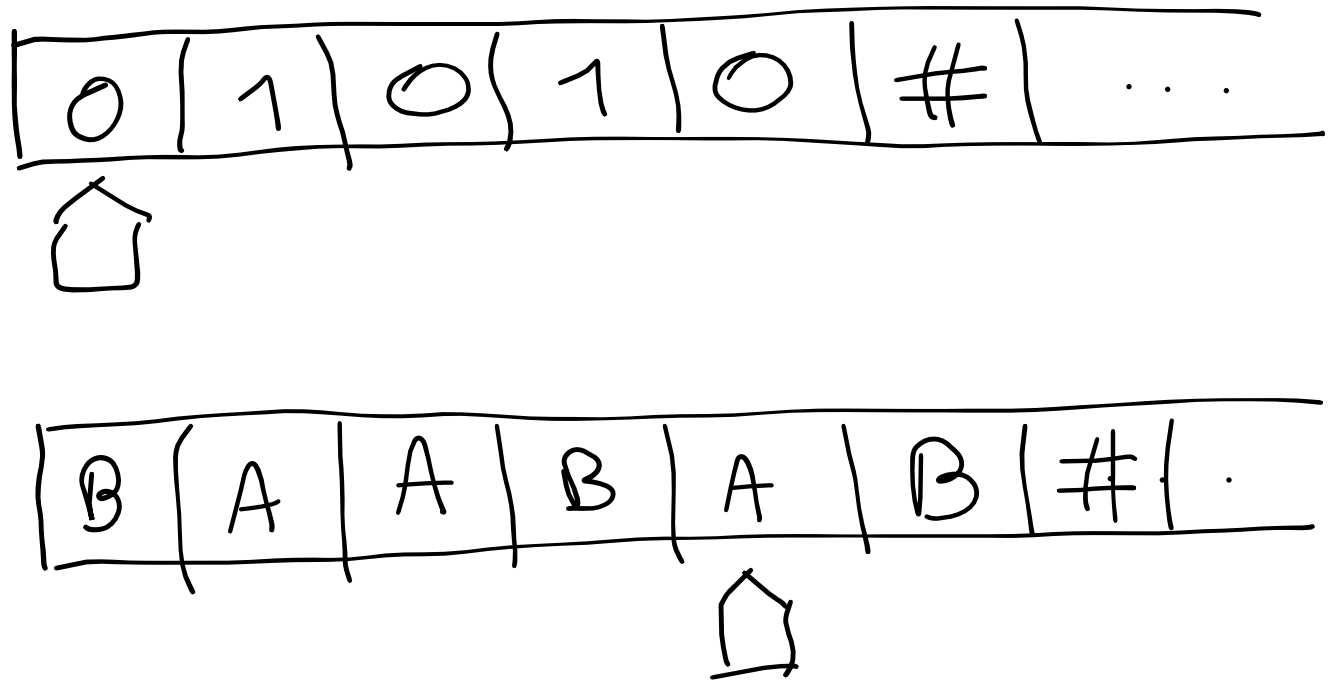



Para simular uma etapa de execução da máquina de fita infinita dupla, a máquina de uma fita infinita a direita deve:

- a partir da primeira leitura, se a função de transição indica um movimento para direita na TM 2 ∞, aplicar 2 movimentos seguidos para a direita; em seguido toda movimentação a direita ou à esquerda será módulo 2;
- a partir da primeira leitura, se a função de transição indica um movimento para a esquerda na TM 2 ∞, aplicar 3 movimentos seguidos para a direita; em seguida toda movimentação a direita ou à esquerda será módulo 2
- o símbolo $ será importante para indicar o momento em que a cabeça volta para o centro da fita (ele será detectado somente nos movimentos à esquerda)

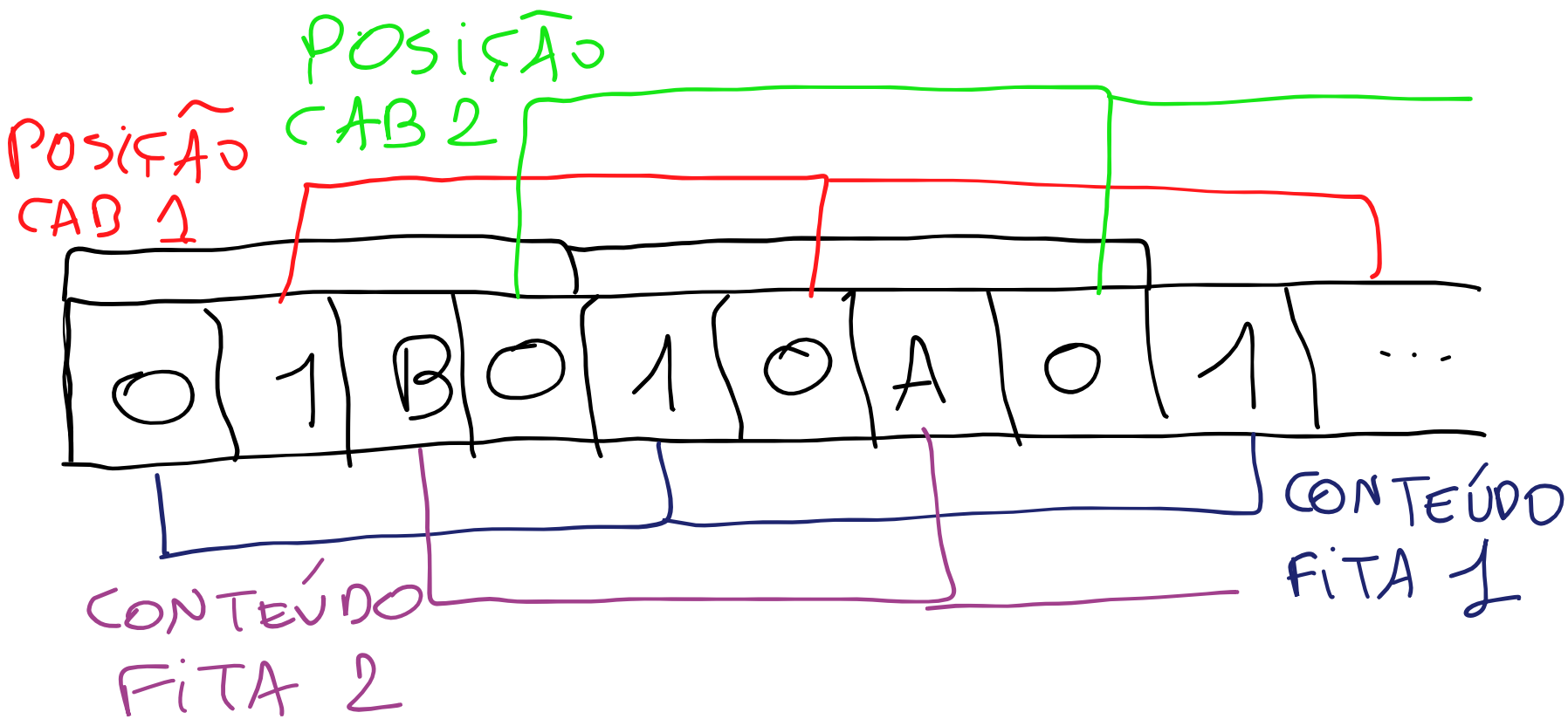
A definição formal da TM que simula a TM 2 ∞ é complexo, mas o importante é de se convencer que tal máquina pode ser construída e simular o comportamento da TM 2 ∞ !

# Exemplo 2: Estudo de uma MT de 2 fitas




A configuração da máquina é caracterizada por um um estado discreto (de um grafo dos estados), pelo conteúdo de cada fita, e pela posição das cabeças de leitura.

Prova ❓ Mostrar que existe um TM tradicional (infinita a direita somente) que consegue simular uma TM 2 fitas

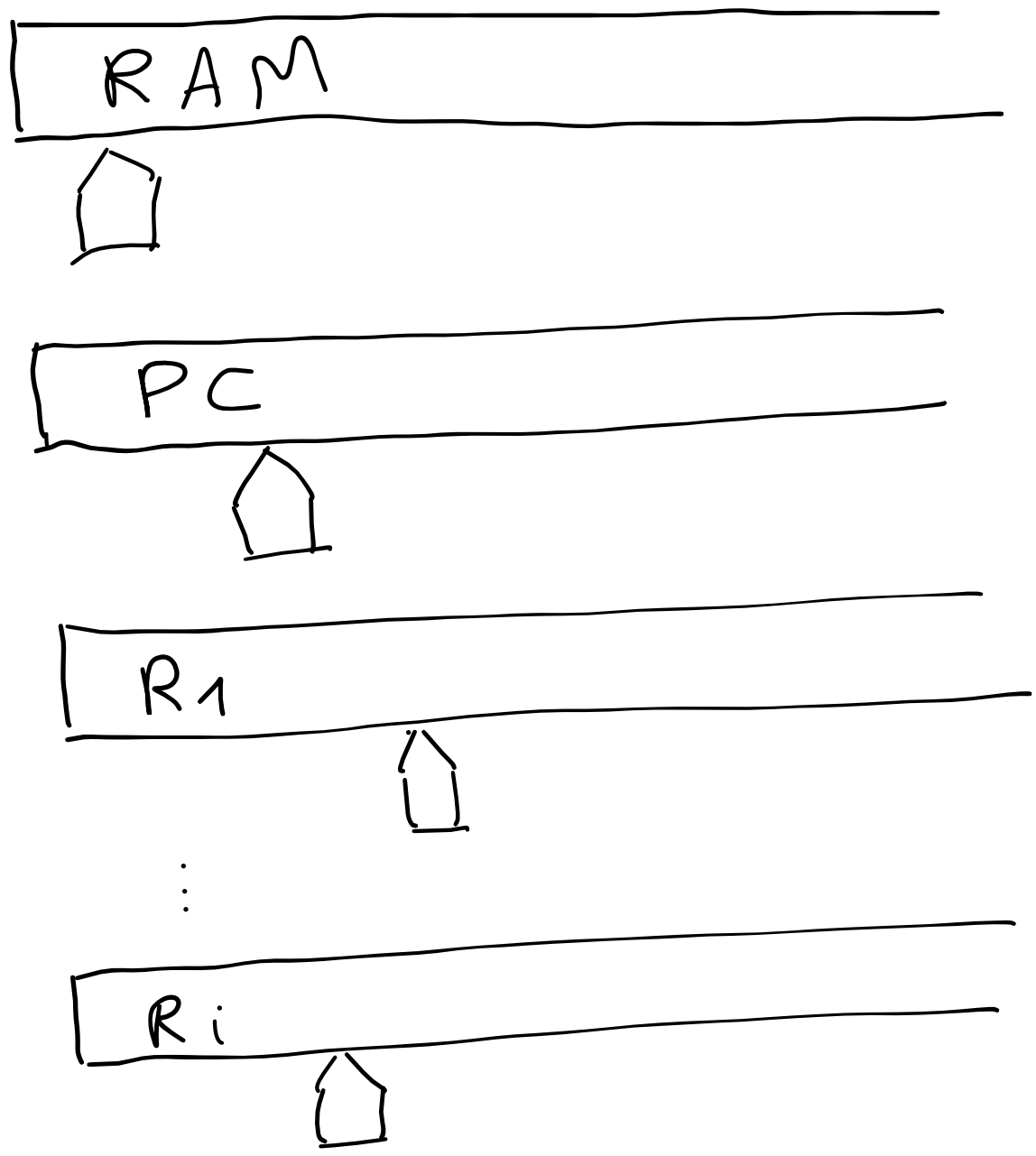



Para simular uma etapa de execução da máquina de 2 fitas, a máquina de 1 fita deve:
- encontrar as posições das cabeças e determinar os símbolos lidos;
- modificar os símbolos, movimentar as cabeças de acordo com o movimento previsto na função de transição, e modificar o estado.

A definição formal da máquina de fita única que simula a máquina de 2 fitas é bastante complexa mas é fácil de se convencer que ela pode ser construída.

Exemplo 3: PC (máquina a memória de acesso direto)

Um computador é composto por uma memória de acesso direto (RAM- Random Access Memory) e um certo número de registradores (em particular um contador de instrução - PC : Program Counter)

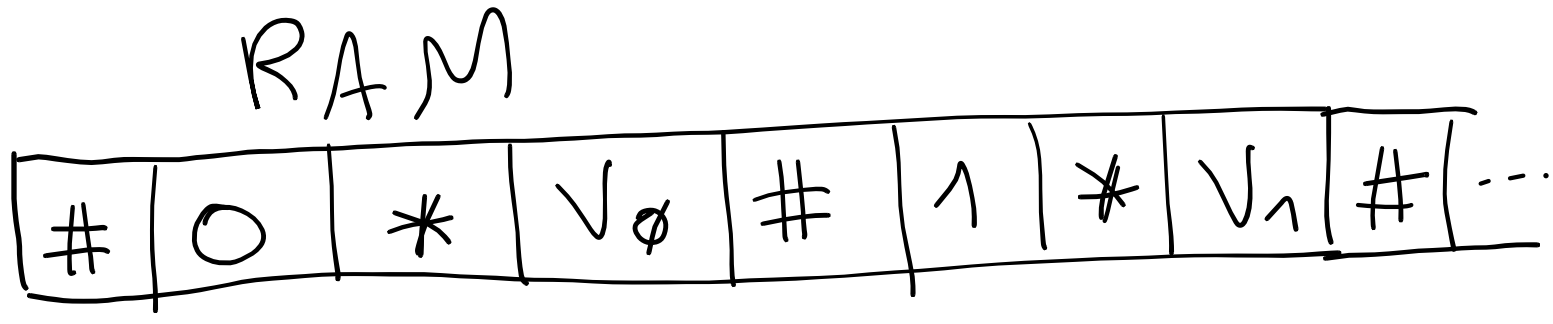


Prova ❓ Mostrar que existe um TM tradicional (infinita a direita somente) que consegue simular um PC

Um PC pode ser simulado por uma TM de várias fitas: 1 fita para a memória e 1 fita para cada registrador.

O conteúdo da memória é representado por pares de palavras (endereço + conteúdo).

Os elementos do par são separados por * e os pares são separados por #



Para simular o PC, a TM de fita simples faz:
1) Percorrer a fita da RAM até encontrar o endereço que se encontra na cabeça da fita PC
2) Ler e decodificar (com o autômato correspondente) a instrução que se encontra no endereço lido.
3)Eventualmente encontrar na RAM os argumentos da instrução.
4) Executar a instrução, o que implica eventualmente a modificação da RAM e/ou dos registradores.
5) Incrementar o contador de programa (ao menos de encontrar uma instrução do tipo GOTO que manda a cabeça da fita PC no endereço correspondente.
6) Repetir o ciclo.

É o óbvio que a formalização de um PC por uma TM de fita simples é extremamente complexa; mas é evidente que com muita paciência poderíamos construir tal máquina.

Observação sobre a complexidade da TM:

Poderíamos mostrar que para simular n movimentos de uma máquina de várias fitas, no máximo, precisaríamos de n^2 movimentos com uma TM de 1 fita, o que corresponde a uma complexidade de tipo polinomial e não exponencial (exponencial seria por exemplo 2^n)

Se n=10 então n^2=100, enquanto que 2^10= ?

Veremos na parte sobre a complexidade que em particular, os processamentos em paralelo só permitem reduzir em tempo problemas polinomiais mas não exponenciais !

Na pratica isso significa que um problema exponencial com n grande não será tratável mesmo usando em paralelo vários processadores.

## 5.5 Máquinas de Turing não deterministicas

Diferença na definição da função de transição.

Para uma configuração dada, a função de transição define um conjunto de triplas ($Q \times \Gamma \times \{L,R\}$). Cada elemento do conjunto deriva então uma configuração distinta.

Na pratica, significa que a partir de uma configuração dada, várias evoluções são possíveis. No caso de uma TM determinístico, considerando a função de transição, uma única configuração será derivada **!**

Se uma das execuções possíveis da TM não determinístico contém uma configuração onde aparece um estado de aceitação, então a máquina aceita a palavra de entrada da fita. Eventualmente outras sequências de configurações para a mesma palavra de entrada poderão levar a máquina a estados que não são de aceitação ou a sequências de configuração infinitas.

No caso de TM não deterministas, que não correspondem à noção clássica de procedimento efetivo (existem programas de computadores não determinístico ❓), só se fala em linguagens aceitas.

O conceito de linguagem decidida não faz sentido no caso não determinístico : para uma mesma palavra de entrada, algumas configurações aceitam a palavra e outras não, então o que se pode deduzir sobre a decidibilidade da máquina em relação à palavra ❓

Só pode se dizer então que a TM não determinista aceita a palavra através de uma das sequências de configurações possível ❗

Teorema: Toda linguagem aceita por uma TM não determinista é também aceita por uma TM determinista.

Isso significa que o não determinismo não aumenta o poder computacional das TM : mais uma comprovação da tese de Church-Turing ❗

O estudo das TM não deterministas parece não ter interesso no caso da formalização da noção de procedimento efetivo !

Será que tais Máquinas existem (ou existirão ) ?

Mas elas são necessárias para definir o conceito de problema NP (e NP-completo / NP-Hard) !

Prova do teorema ?

Devemos provar que a TM det simula todas as execução da TM não det que levam à aceitação das palavras.

Uma solução seria de simular cada execução possível da TM não det: o problema é que certas execuções possíveis podem ser infinitas; entrando numa destas execuções, a gente não vai poder encontrar execuções possíveis que aceitam a palavra !

Uma possível representação do funcionamento da TM não det é através de uma árvore: cada ramo da árvore representa então uma sequência de configuração possível. Caso um dos ramos leva a um estado de aceitação, então a máquina aceita a palavra.

Como garantir que a simulação da TM det vai encontrar todos os ramos de aceitação ?

Uma ideia tentadora seria de explorar a árvore usando um mecanismo de busca em profundidade: o problema é a possibilidade de explorar um ramo infinito.

A solução para não perder nenhum ramo de aceitação é utilizar um percurso em largura **!**
Porque **?**

A realização da TM det seria dada por uma máquina de 3 fitas (equivalente também a uma máquina de fita simples):
- 1 fita para receber a palavra (o conteúdo desta fita não sera alterado);
- 1 fita que simula uma execução possível;
- 1 fita que mantém o registo das posições na árvore (fita da busca em largura que contém o conjunto das configurações a serem exploradas ainda).

## 5.6 Máquina de Turing universal

Existe uma TM que pode simular qualquer TM !

Tal máquina M recebe como entrada na fita a palavra w que corresponde à descrição de uma TM M' qualquer assim como a palavra de entrada w' que M' deve processar.

Tal máquina será então um interpretador de máquina de Turing e é chamada de TM universal.

Na pratica, podemos encontrar na internet numerosos simuladores de TM. Tais simuladores executam qualquer máquina de Turing e são executados num PC que pode ser representado também por uma máquina de 1 fita !

É claro que na pratica tal máquina será complexa demais para ser modelada.

## 5.7 Função calculável por uma TM

Até agora, as máquinas apresentadas só servem para tratar problemas de tipo booleano (uma palavra que pertence ou não a uma certa linguagem)

Na pratica, em computação, calculamos, na maior parte do tempo, funções que dependem de argumentos que seriam as palavras de entrada das TM:

Definição: uma TM que calcula uma função

$f: \Sigma^* \rightarrow \Sigma^*$

se para toda palavra de entrada w, ela para numa configuração onde o resultado f(w) se encontra na fita no lugar de w.

Versão alternativa da tese de Church-Turing (não é um teorema ❗ ):

Uma função é calculável se e somente se existe uma TM que a calcula.

Enunciado alternativo da tese de Church-Turing: As funções calculáveis por um procedimento efetivo (algoritmo) são as funções calculáveis por uma TM.

Decidir uma linguagem é um tipo de calculo de função cujo domínio de resposta é : {0,1}

Existem versões alternativas das TM sem a noção de estado de aceitação mas onde os valores 0 ou 1 aparecem na fita quando é atingido um estado de parada.

Na pratica, o calculo de funções por TM não constitua a forma mais adequada de formalização do conceito de procedimento efetivo.

Outros modelos computacionais baseados na definição de funções recursivas (como as funções μ-recursivas ou o λ-calculo) são mais adaptados ao calculo de funções.

Mas para todos estes formalismos computacionais, existem teoremas que mostram a equivalência com as TM, mantendo válida a tese de Church-Turing.