Teoria da Computação - TP1 - part 3

Simulação Máquina de Post

Professor: Walace de Almeida Rodrigues
Aluno: Lucas Gabriel de Almeida - Número: 0035333
Aluno: Lucas Mateus Fernandes - Número: 0035411

Formiga 27 de Fevereiro de 2021.

A seguir, será demonstrado que qualquer programa escrito para a Máquina de Turing (na versão Morphett) pode ser simulado na Máquina de Post.

Para a Máquina de Post Simular a Máquina de Turing um passo fundamental é implementar os elementos básicos da Máquina de Turing portanto para atingir tal objetivo optamos pela seguinte abordagem

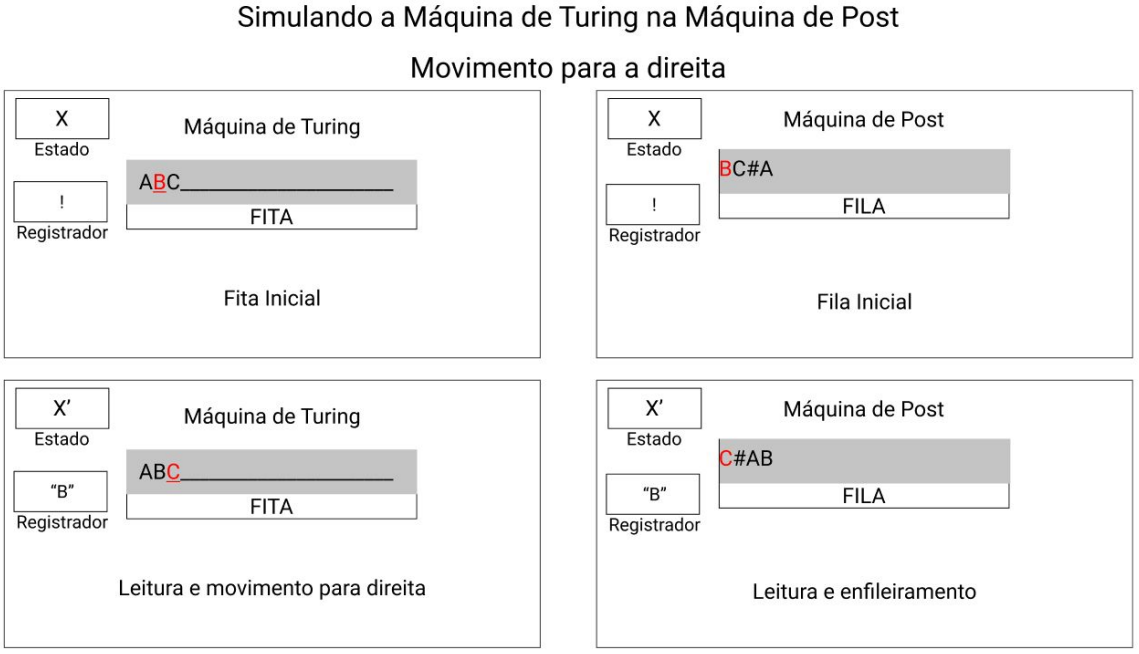
Dada uma Maquina de Post com as seguintes características:

- Alfabeto = { A, B, C } U "conjunto de Caracteres Especiais"
    - "#" = carácter especial que representa final de fila;
    - "$" = carácter especial que representa espaço em Branco;
    - "!" = carácter especial que representa meta carácter Genérico
- estados = {(#,#),(#,!),(#,$),(#,A),(#,B),(#,C),(!,#),(!,!),(!,$),(!,A),(!,B),(!,C), ($,#),($,!),($,$),($,A),($,B),($,C),(A,#),(A,!),(A,$),(A,A),(A,B),(A,C), (B,#),(B,!),(B,$),(B,A),(B,B),(B,C),(C,#),(C,!),(C,$),(C,A),(C,B),(C,C)}

Dada uma Máquina de Post com 'n' caracteres no seu alfabeto, para cada caracter Ci e Cj pertencente ao alfabeto existe um estado representado pelo par ordenado (Ci, Cj).

Fazendo uma pequena observação temos que cada elemento do conjunto de estados representa uma configuração específica da máquina de post, sendo que Ci está relacionado ao penúltimo carácter lido e Cj ao último, desde que Cj seja diferente de "!" pois quando Cj receber o valor '!' Ci começa a representar o último valor lido e Cj fica fixo com o carácter '!' indicando que o significado (Ci,Cj) foi "alterado".

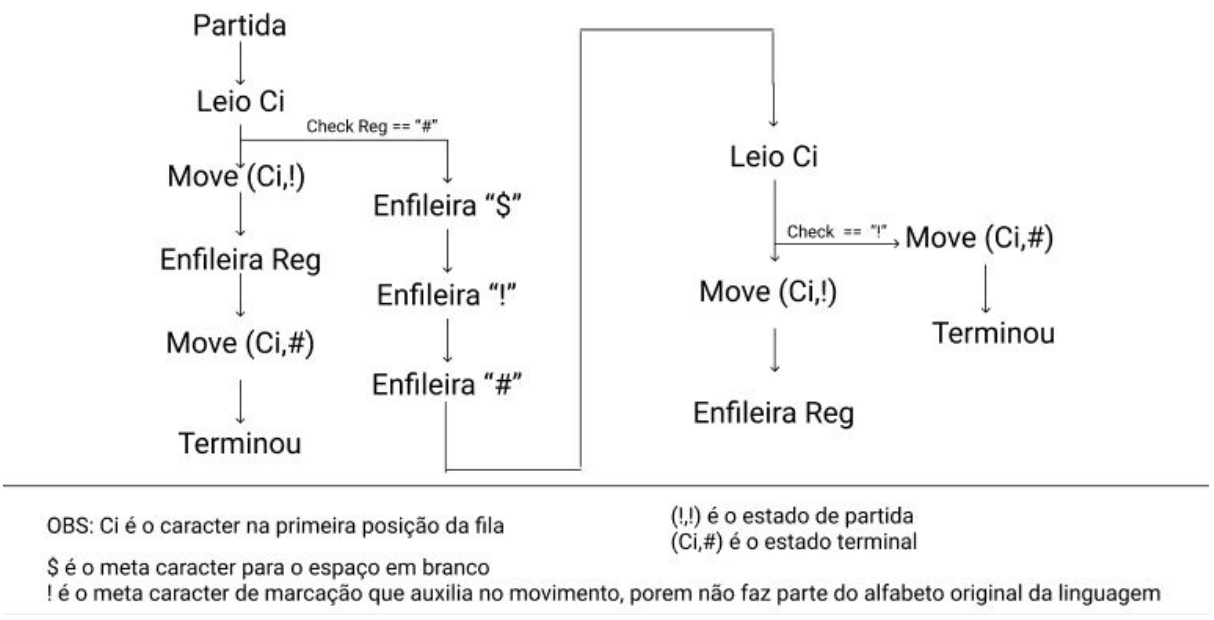
A máquina começa pelo estado (!,!) pois ainda não houve leitura e o "!" é um meta carácter genérico..



Temos a MT, com a fita “ABC” com o cabeçote apontando para “B” e sua respectiva representação na MP com a fila “BC#A”, sendo que ambas possuem o valor “!” em seu registrado e um estado arbitrário ‘X”.

Na MT ao realizar uma ação de movimento para direita está implícito a leitura do caracter a escrita de algo na fita e o movimento do cabeçote.

Para que a MP realize tais procedimentos foi criado o diagrama de fluxo presente a seguir:



Seguindo tal diagrama e partido da configuração da máquina previamente expressa chegamos na efetiva simulação do movimento a direita o que pode ser visto através da tabela de computação abaixo:

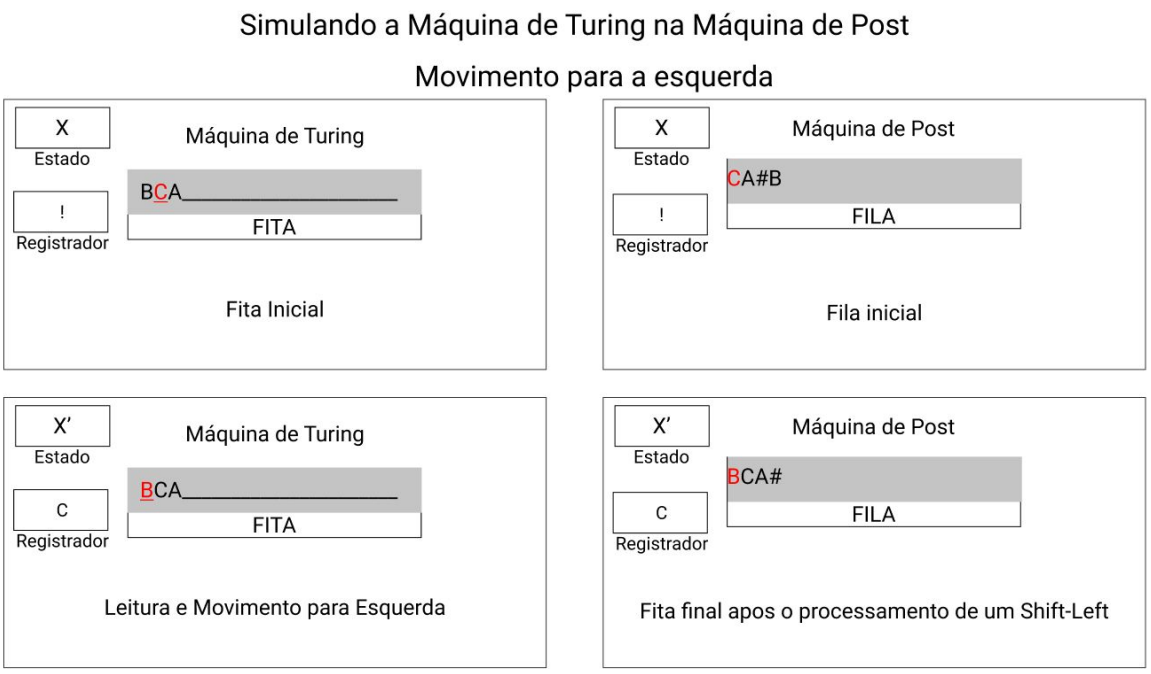
Tabela de Computação
(movimento para direita)

Fila	Estado	Registrador	Função
BC#A	(!,X)	$	Partida
C#A	(!,X)	B	Leitura
C#A	(B,X)	B	Move
C#AB	(B,X)	B	Enfileira
C#AB	(B,X')	B	Move
C#AB	(B,X')	B	Terminou

Para fins de verbosidade, criamos label para algumas funções
Leitura equivale ao comando " *X ler Y'* "
Check Reg = 'argumento' equivale ao comando " *Y' if argumento X'* "
Move equivale ao comando " *Y' if arg Y''* "
Enfileira equivale ao comando " *Y'' atrib arg X'* "
Terminou significa que o movimento foi Simulado.

Vale ressaltar que durante o movimento para a direita o par ordenado de estados (Ci,Cj) recebe o seguinte significado

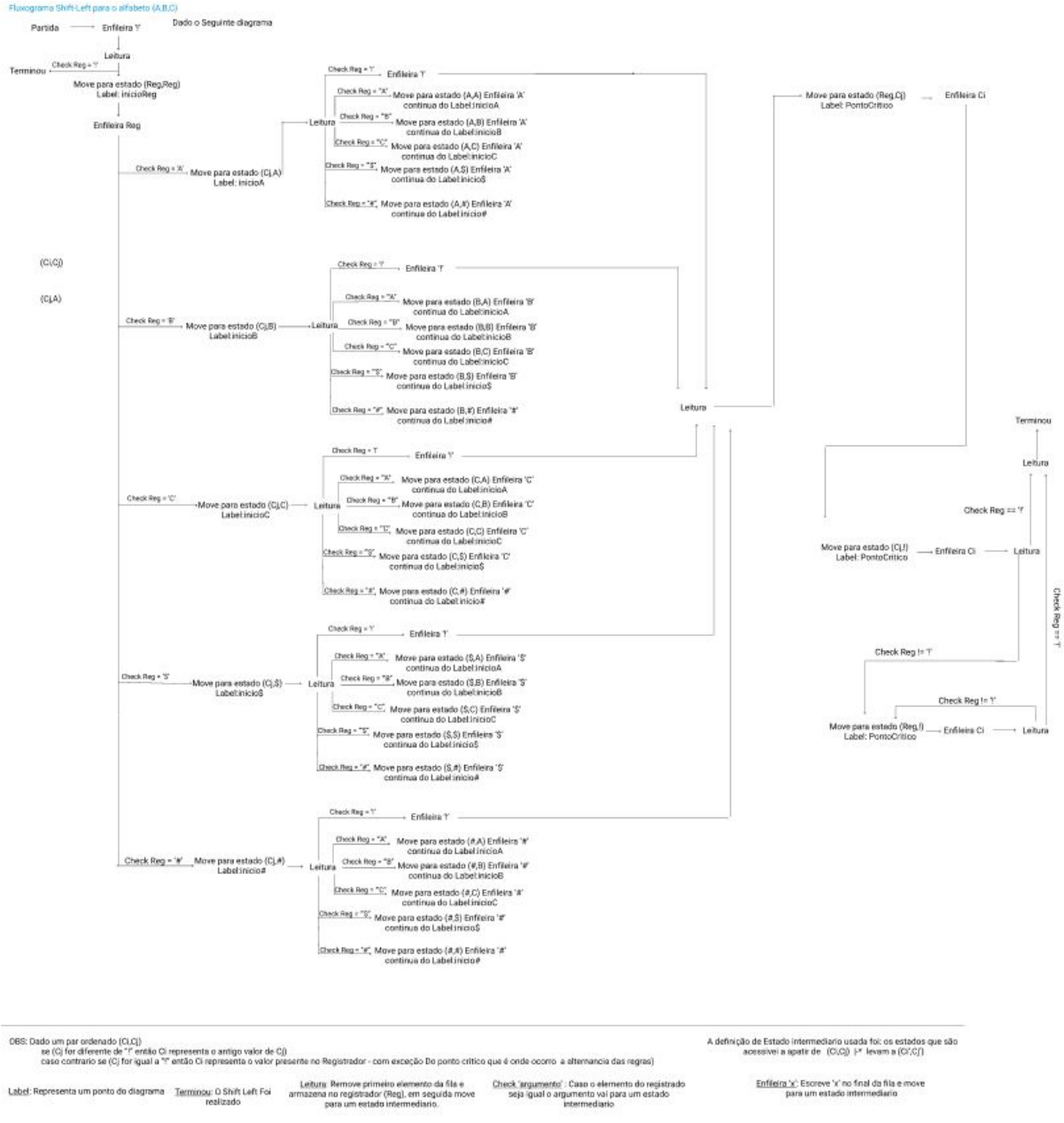
- Ci = Valor do Registrador
- Cj = Estado



Temos a MT, com a fita “BCA” com o cabeçote apontando para “C” e sua respectiva representação na MP com a fila “CA#B”, sendo que ambas possuem o valor “!” em seu registrado e um estado arbitrário ‘X”.

Na MT ao realizar uma ação de movimento para esquerda está implícito a leitura do caracter a escrita de algo na fita e o movimento do cabeçote.

Para que a MP realize tais procedimentos foi criado o diagrama de fluxo presente a seguir:



Seguindo tal diagrama e partido da configuração da máquina previamente expressa chegamos na efetiva simulação do movimento a direita o que pode ser visto através da tabela de computação abaixo:

Tabela de Computação

Fila	Estado	Registrador	Função
CA#B	(!,!)	!	Partida
CA#B	(!,!)	!	Partida
CA#B!	(!,!)	!	Enfileira
CA#B!	(!,!)	!	Enfileira
CA#B!	(!,!)	!	Enfileira
CA#B!	(!,!)	!	Enfileira
A#B!	(!,!)	C	Leitura
A#B!	(C,C)	C	Move
A#B!C	(C,C)	C	Enfileira
A#B!C	(C,C)	C	Move
#B!C	(C,C)	A	Leitura
#B!C	(C,A)	A	Move
#B!CC	(C,A)	A	Enfileira
B!CC	(C,A)	#	Leitura
B!CC	(A,#)	#	Move
B!CCA	(A,#)	#	Enfileira
!CCA	(A,#)	B	Leitura
!CCA	(#,B)	B	Move
!CCA#	(#,B)	B	Enfileira
CCA#	(#,B)	!	Leitura(Começo transição)
CCA#!	(#,B)	!	Enfilleira(Transição)
CA#!	(#,B)	C	Leitura(Transição)
CA#!	(C,B)	C	Move(Transição)
CA#!C	(C,B)	C	Enfileira(reg)
CA#!C	(B,!)	C	Move (reg)
CA#!CB	(B,!)	C	Enfileira (reg)
A#!CB	(B,!)	C	Leitura
A#!CB	(C,!)	C	Move
A#!CBC	(C,!)	C	Enfileira
#!CBC	(C,!)	A	Leitura
#!CBC	(A,!)	A	Move
#!CBCA	(A,!)	A	Enfileira
!CBCA	(A,!)	#	Leitura
!CBCA	(#,!)	#	Move
!CBCA#	(#,!)	#	Enfileira
CBCA#	(#,!)	!	Leitura
BCA#	(#,!)	C	Leitura
BCA#	(#,!)	C	Terminou

Para fins de verbosidade, criamos apelidos para algumas funções
Leitura equivale ao comando “ *X ler Y’* ”
Check Reg = ‘argumento’ equivale ao comando “ *Y’ if argumento X’ ”*
Move equivale ao comando “ *Y’ if arg Y’’* "
Enfileira equivale ao comando “ *Y’’ atrib arg X’* ”
Terminou significando que o movimento foi Simulado.
Label representa um ponto de referência dentro do diagrama

Vale ressaltar que durante o movimento para a esquerda o par ordenado de estados (Ci,Cj) recebe o seguinte significado

- Se (Cj for diferente de “!” então Ci representa o antigo valor de Cj que equivale ao antigo valor do registrando, podendo assim simular um pseudo registrador adicional)
- Caso contrário se (Cj for igual a “!” então Ci representa o valor presente no Registrador)