Utilização de Máquina de Turing aplicada a um problema de comparação de Listas de Palavras

Vinícius Jurinic Cassol [1]
Lucelene Lopes [1]
Aline Duarte Riva [1]

Resumo: Este tutorial apresenta duas abordagens distintas para a construção de duas máquinas de Turing dedicadas a uma mesma aplicação: a comparação de listas de palavras. Ambas as máquinas foram implementadas no software Visual Turing e recebem como entrada duas listas de palavras sobre um alfabeto definido, gerando como saída uma lista com apenas as palavras presentes em ambas. O propósito deste artigo é ilustrar o desenvolvimento de máquinas de Turing de uma forma pedagógica para que estudantes de computação e áreas relacionadas possam ter um exemplo prático e relativamente complexo desta forma de processamento.

Palavras-chave: Comparação de Lista de Palavras, Máquina de Turing, Visual Turing.

Abstract: *This tutorial presents two different approaches to build two distinct Turing machines to a same application: a comparison between two lists of words. Both machines were implemented in Visual Turing software and they receive as input two lists of words over a given alphabet in order to deliver an output list with only the words present in the two input lists. The purpose of this paper is to illustrate the Turing machine development process in a pedagogical way to allow students of computer science and related areas the contact with a rather complex practical example of such processing style.*

Keywords: *Comparison of Word List, Turing Machine, Visual Turing.*

1 Introdução

Alan Mathison Turing nasceu em 23 de junho de 1912 em Londres. Em abril de 1936 concluiu a ideia que hoje é conhecida como "máquina de Turing", sendo publicada no final deste mesmo ano no artigo "On computable numbers, with an application to the Entscheidungsproblem" [1]. Como o título sugere, era apenas uma aplicação da nova ideia de computabilidade matemática, especificando o conjunto de ações disponível em sua máquina.

A ação da máquina de Turing é determinada pela configuração e pelo símbolo em que ela se encontra no momento da verificação [2]. A ação é dividida em três passos: 1) apaga ou imprime um símbolo especificado; 2) move-se para esquerda ou direita; 3) muda para uma configuração nova.

Uma ação completa é definida por Turing como uma tabela de comportamento, ou seja, cada tabela de comportamento é uma máquina de Turing diferente [4]. A tese de Turing é de que, mesmo sendo cada ação restrita em sua forma, um conjunto de ações pode compor todas as operações matemáticas possíveis, sugerindo que todas as operações da mente humana poderiam ser desempenhada por computadores.

Para Turing,com sua máquina era possível fazer o trabalho do calculador humano, indicando haver possibilidade de máquinas computacionais serem construídas da união da matemática e lógica em processadores de símbolos. Durante a II Guerra Mundial, Turing trabalhou no Departamento de Comunicação da Grã-Bretanha na tentativa de quebrar códigos da comunicação alemã, que eram produzidos por um tipo de computador denominado Enigma. Logo após, Turing foi para os EUA para estabelecer códigos seguros para comunicações entre transatlânticos aliados. Quando terminou a guerra, Turing aliou-se ao National Physical Laboratory para desenvolver o

[1]Programa de Pós-Graduação em Ciência da Computação, PUCRS, Porto Alegre, Brasil
{vinicius.cassol, aline.riva}@acad.pucrs.br, lucelene.lopes@pucrs.br

doi: 10.5335/rbca.2010.013

ACE (*automatic computing engine*), um computador totalmente inglês. Contudo, com a demora do processo de construção, decepcionado e deprimido, Turing mudou-se para Manchester e, durante uma crise, cometeu o suicídio comendo uma maçã com cianureto de potássio em 7 junho de 1954.

A máquina de Turing é aceita na comunidade científica para formalização de algoritmo [4]. Apesar de ser um mecanismo simples, formaliza o processo de um humano que realiza cálculos, utilizando para tanto um sistema de escrita e um apagador. Assim, realiza tarefas de qualquer computador com propósito geral. Formalmente, a máquina de Turing é similar aos computadores atuais, tendo como base uma fita, uma unidade de controle e um programa [3].

Como referido anteriormente, a máquina é constituída de três partes:

- Fita: Utilizada como compartimento de entrada, de saída e como dispositivo de memória de trabalho;
- Unidade de Controle: Mostra o estado em que a máquina se encontra. Composta por um agente de leitura e gravação (cabeça da fita), o qual percorre uma célula da fita por vez, movimentando-se em dois sentidos (esquerda ou direita);
- Programa ou Função de Transição: Função que define e comanda as leituras, as gravações e em qual sentido se movimenta a unidade de controle (cabeça da máquina).

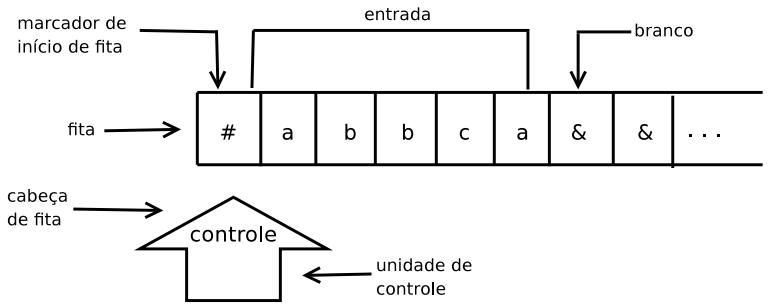


Figura 1. Descrição genérica de máquina de Turing [5].

A Figura 1 apresenta o exemplo de uma fita da máquina de Turing, sendo, à esquerda, seu movimento finito e, à direita, pode ser infinito. A fita é dividida em células, cada contendo um símbolo, que geralmente pertence a um alfabeto de entrada, a um alfabeto auxiliar, ou pode ser "branco" ou "marcador de início de fita" [5].

Este trabalho tem por objetivo apresentar a solução para um determinado problema computacional pelo do uso do formalismo de máquina de Turing. O problema escolhido é o de análise de duas listas de palavras para determinar quais estão presentes em ambas as listas, ou seja, a intersecção dessas listas. Adicionalmente, após determinar quais são as palavras presentes em ambas as listas, a máquina conta o número de palavras presentes em ambas, ou seja, calcula a cardinalidade da intersecção entre as listas.

O presente artigo está organizado da seguinte forma: a próxima seção apresenta uma breve definição do software Visual Turing 2.0 utilizado para implementar a máquina de Turing proposta; a seção 3 detalha o problema e apresenta uma tentativa inicial de implementação da máquina proposta através de uma abordagem simplista; em sequência, a seção 4 apresenta a verdadeira implementação que soluciona o problema proposto e a seção 5 apresenta a execução detalhada de um exemplo prático de aplicação da máquina descrita na seção 4; finalmente, a conclusão sumariza os resultados deste trabalho e sugere possíveis extensões ao mesmo.

2 Visual Turing 2.0

Para o desenvolvimento do presente trabalho utilizou-se o Visual Turing 2.0 [6]. Trata-se de uma ferramenta *open-source* que permite a criação de máquinas de Turing. Para sua escolha considerou-se o fato de se tratar de uma ferramenta gratuita e de fácil manejo, o que facilita a implementação do trabalho. Destaca-se, porém, que

se encontram disponíveis outras ferramentas para desenvolvimento de máquinas de Turing, como, por exemplo, o JFLAP[2].

Considerando-se o Visual Turing, observa-se que a ferramenta permite a implementação da máquina de Turing, em que é possível a definição de símbolos, relacionamentos e instruções entre os símbolos, bem como outras máquinas já implementadas. Além disso, esta ferramenta permite que seja definida uma fita, de tamanho infinito, onde cada posição contém um símbolo a ser verificado durante a execução da máquina de Turing. A máquina de Turing, por sua vez, durante a execução, percorre a fita para a esquerda ou direita, conforme definição, e pode ou não escrever na posição da fita em que se encontra.

2.1 Edição de Máquinas de Turing

A área de trabalho do Visual Turing pode ser observada na Figura 2, onde se apresenta uma máquina simples que apenas lê uma letra do alfabeto $\{a, b\}$ e escreve a letra contrária (escreve a se ler b, ou escreve b se ler a).

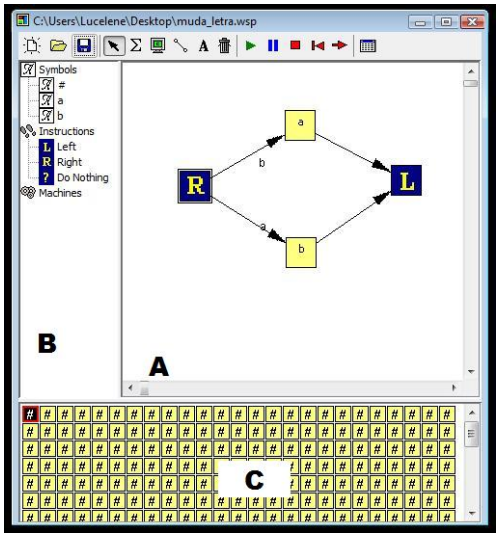


Figura 2. Exemplo do Visual Turing 2.0 que muda uma única letra.

Na Figura 2, a janela marcada com a letra **B** apresenta as primitivas da máquina de Turing, contendo todos os símbolos, componentes, instruções e outras máquinas que podem compor a máquina que está sendo desenvolvida. Na janela marcada com a letra **A**, encontra-se o painel de edição, onde os componentes escolhidos em **B** podem ser trabalhados e conectados a fim de formar a máquina desejada.

Nesse painel, o estado inicial da máquina (primeira instrução) é indicado por uma instrução com borda dupla (neste caso a instrução move-se à direita: **R**). De cada instrução um conjunto de arcos indica as possíveis próximas instruções, podendo ser anotados por letras que indicam quando serão o caminho adotado. De acordo com a letra apontada pela cabeça da fita, um dos arcos será escolhido. No exemplo apresentado, a máquina inicia na instrução vá para a direita. Caso a cabeça da fita aponte para a letra a, a máquina executa a instrução **b**, que escreve a letra b nesta posição. Caso a cabeça da fita aponte para a letra b, a máquina vai para a instrução **a**, que escreve a letra a. Independentemente do caminho ou da letra apontada pela cabeça da fita, a máquina segue após para a instrução **L**, que vai para a esquerda e retorna a cabeça da fita para a posição original.

Completando a identificação das janelas na Figura 2, a janela marcada com a letra **C** contém a fita com os símbolos que serão utilizados durante a execução da máquina.

Como mencionado anteriormente, uma máquina de Turing previamente definida pode ser usada como parte de outra máquina. Esta possibilidade visa atender à modelagem de problemas complexos, na qual se faz necessária

[2] http://www.jflap.org.

a divisão do problema em módulos, sendo que o resultado de um processo executado por uma máquina pode ser utilizado nas condições da máquina atual.

Outras inovações no Visual Turing 2.0 são a existência de dois recursos inexistentes na definição original de máquina de Turing, que facilitam a utilização sem criar qualquer restrição ou aumento do poder computacional do formalismo. Trata-se da instrução *não faz nada* [?], que serve para centralizar fluxos de uma máquina desenhada, ou apenas para ler a letra apontada pela cabeça da fita; e da definição da aplicação repetida de diversas instruções idênticas. Por exemplo, a Figura 3 apresenta uma máquina simples que utiliza estes recursos adicionais. Trata-se de uma máquina que troca as letras em posições ímpares de uma palavra até encontrar o final desta palavra, indicado por uma sequência das letras que representam como "branco" (em Visual Turing 2.0: #).

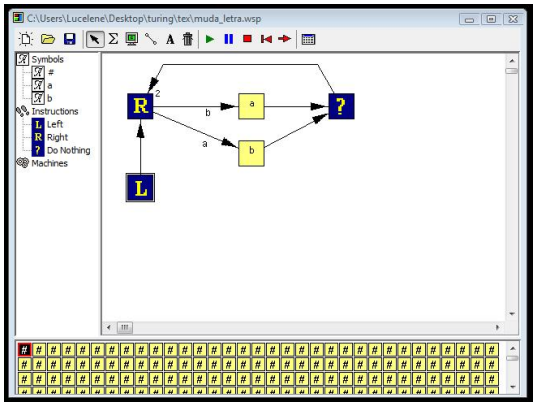


Figura 3. Exemplo do Visual Turing 2.0 que muda letras ímpares de uma palavra.

Esta máquina inicia movendo a cabeça da fita para a esquerda e, em seguida, move a cabeça da fita duas vezes para a direita; dessa forma inicia o processo na primeira letra da palavra. Mais uma vez, caso encontre a letra a, muda-a para b, e vice-versa. Realizada esta troca, vai para a instrução *não faz nada*, que serve apenas para reunir os fluxos e voltar para a instrução que avança duplamente para a direita. Esta máquina encerra naturalmente seu processamento quando a palavra terminar, pois, caso encontre na cabeça da fita a letra # após a instrução que vai duplamente à direita não existe arco com esta anotação; portanto, a máquina encerra sua execução.

2.2 Execução de Máquina de Turing

Após o desenvolvimento da máquina desejada, também é possível executá-la através do Visual Turing 2.0. A máquina será executada com a velocidade padrão de 100 instruções por segundo, a qual pode ser alterada, caso desejado, pressionando-se F7 durante a execução. Dentre outras características, é possível pausar e reiniciar a execução, bem como executar a máquina manualmente, acompanhando a execução passo a passo.

3 Problema de Comparação de Lista e Solução Simplista

O problema a ser resolvido por este trabalho, através de uma Máquina de Turing, é a comparação de duas listas com o objetivo de manter as palavras escritas na primeira lista que se encontram presentes na segunda lista. As palavras são compostas pelo alfabeto a e b, não ultrapassando a quantidade de três caracteres para cada palavra. Para a marcação de final das duas listas na fita utilizaram-se os caracteres #z#. Para esta comparação, a máquina lê cada um dos elementos da primeira lista e compara com todos os elementos da segunda lista com o objetivo de apagar, ou seja, preencher com # os elementos da primeira lista que não aparecem na segunda.

3.1 Formato de Entrada

O formato de entrada são duas listas de palavras. Nesta versão, as palavras são, necessariamente, de 1, 2 ou 3 letras sobre um alfabeto `a,b`.

Os caracteres #z# (símbolos de controle) são utilizados para sinalizar o final de uma lista. Logo, na fita da máquina fica a #<primeira lista de palavras separada por #>#z#<segunda lista de palavras>#z#. Por exemplo, dado as duas listas a seguir, a comparação dessas listas tem como intersecção as palavras aba, bbb e a.

$$aba \;\; ba \;\; bbb \;\; a \qquad \text{e} \qquad aa \;\; bbb \;\; b \;\; aba \;\; a \;\; bab$$

3.2 Formato de Saída

No formato de saída, somente a intersecção das duas listas (palavras presentes em ambas listas) permanecem na primeira lista, sendo as demais palavras apagadas com #. Por exemplo, dadas como entrada duas listas de palavras, a saída também serão duas listas de palavras, sendo que a primeira lista possui a cardinalidade do conjunto intersecção.

Entra: #aba#ba#bbb#a#z#aa#bbb#b#aba#a#bab#z#
Sai: #aba####bbb#a#z#aa#bbb#b#aba#a#bab#z#

3.3 Algoritmo de Solução Simplista

Algoritmo 1: Processo Geral da Abordagem Simplista

```
1  Para todas as palavras p da primeira lista
2      Reconhece a palavra p
3      Marca o fim da palavra p e vai para o início da próxima lista
4      Para todas palavras q da segunda lista
5          Se p = q então
6              Aceita a palavra p (desmarca o fim da palavra p)
7              Interrompe a busca na segunda lista
8      Se chegou até ao fim da segunda lista sem que p fosse igual a q
9          Recusa a palavra p (retorna na primeira lista e "apaga" a palavra p)
```

A implementação prática do processo descrito no algoritmo 1 é feita por meio de uma máquina de Turing apresentada na Figura 4. Nesta máquina, o alfabeto é composto pelas letras a e b, além dos símbolos de controle # e z.

Esta máquina tem por função reconhecer repetidamente as palavras da primeira lista (linhas 1 e 2 do Algoritmo 1) e chamar outras máquinas para executar as demais tarefas. Uma vez reconhecida a palavra da primeira lista, são chamadas duas máquinas de Turing:

- uma submáquina genérica *marca_e_vai*, que marca a palavra reconhecida e vai para o início da segunda lista; e
- uma submáquina específica *busca_**, que busca encontrar na segunda lista a palavra reconhecida na primeira.

3.3.1 Submáquina genérica *marca_e_vai*

A linha 3 do algoritmo 1 é implementada pela máquina de Turing genérica representada na Figura 5, onde a palavra reconhecida é marcada com o símbolo z no final e o leitor da máquina é posicionado no início da segunda lista, ou seja, após a sequência de símbolos #z#. Esta máquina escreve o símbolo z e depois avança para a direita até encontrar o símbolo z (que marca o fim da primeira lista) e, em seguida, anda mais uma posição para a direita, lendo o símbolo #, posicionando o leitor no início da segunda lista.

3.3.2 Máquinas específicas de reconhecimento de palavras

De acordo com a palavra reconhecida no processo geral (máquina da Figura 4), após chamar a submáquina *marca_e_vai*, uma submáquina de Turing específica é chamada para verificar as palavras da segunda lista, até

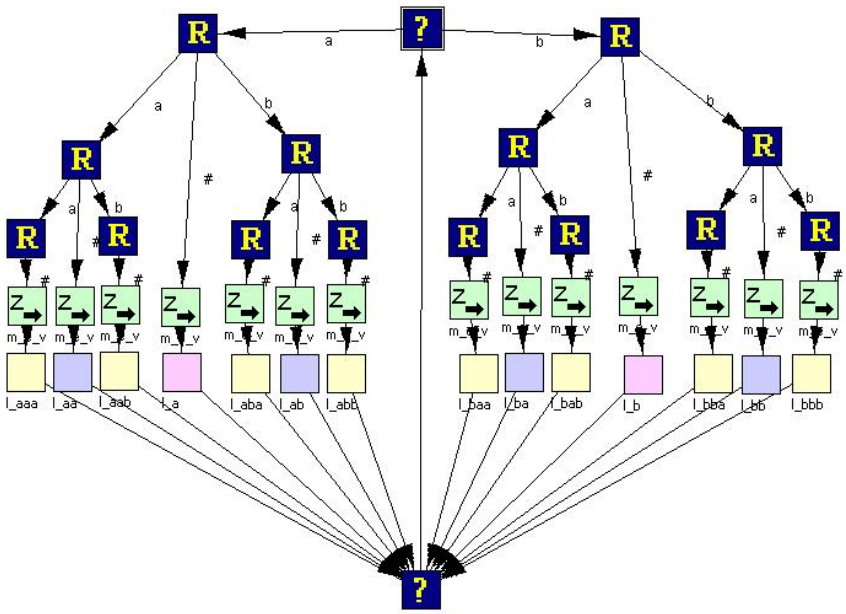


Figura 4. Principal máquina de Turing responsável por reconhecer repetidamente as palavras da primeira lista e chamar outras máquinas de Turing auxiliares ao processo.

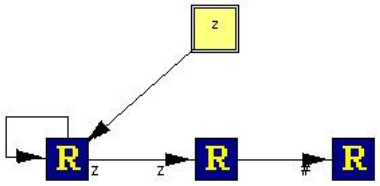


Figura 5. Máquina de Turing responsável por marcar o final da palavra reconhecida na primeira lista e posicionar o leitor da máquina após a sequência `#z#`.

encontrar a palavra reconhecida (neste caso aceita esta palavra), ou encontrar o fim da lista (neste caso recusa a palavra).

A Figura 6 apresenta todas as máquinas que buscam as possíveis 14 palavras que podem ser reconhecidas. Basicamente, essas máquinas buscam a sequência com a palavra encontrada seguida do símbolo `#` e, caso não encontrem, avançam para a próxima palavra da lista (até o símbolo `#`), onde tentam novamente encontrar esta sequência. Caso encontre a sequência, a máquina "*aceita palavra*" é chamada e a busca é finalizada. Caso a sequência não seja encontrada até o final da segunda lista, ou seja, até encontrar a sequência `#z#`, a máquina "*rejeita palavra*" é acionada.

3.4 Problemas da solução simplista

Ainda que funcione de forma adequada, a solução simplista tem um grave problema para ser generalizada e tratar palavras em um alfabeto maior do que apenas duas letras (a e b) e com um número qualquer de letras (não apenas 1, 2 ou 3 letras em cada palavra). Na verdade, o funcionamento desta máquina simplista está baseado em fazer um reconhecedor para cada palavra possível e, portanto, dificilmente pode ser generalizado.

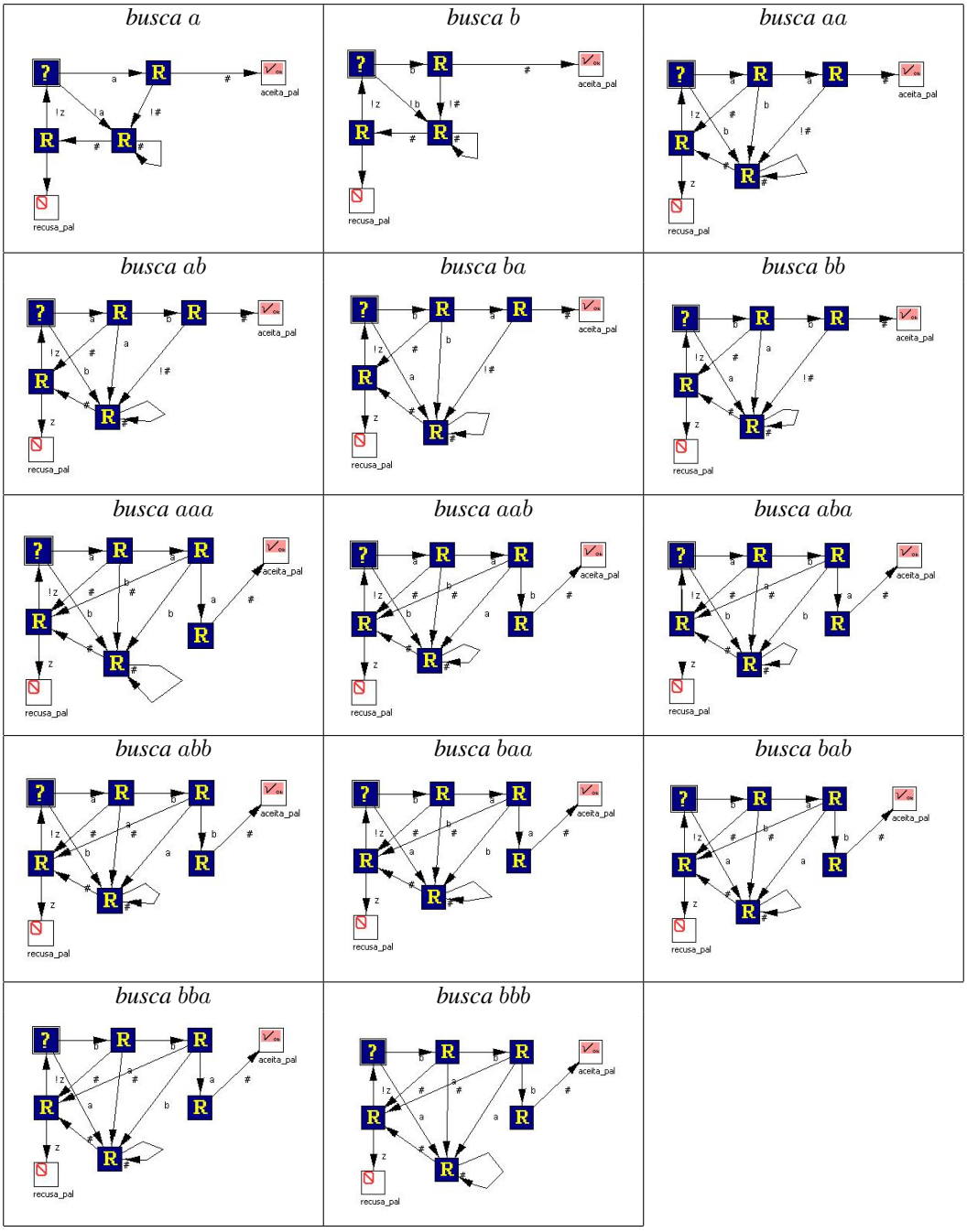


Figura 6. Máquinas de Turing responsáveis por buscar na segunda lista cada uma das possíveis palavras reconhecidas na primeira lista até encontrá-la ou até encontrar o fim da segunda lista.

Nesse sentido, esta abordagem foi abandonada. Em vista disso, uma solução mais genérica e que possa ser generalizada para outros alfabetos com um número virtualmente infinito de palavras é proposta na próxima seção.

4 Solução Sofisticada

O algoritmo básico da solução proposta segue a mesma linha de raciocínio do algoritmo anterior, no sentido de que repete as tarefas de analisar cada uma das palavras da primeira lista, tentando encontrá-las por passagens repetidas pelas palavras da segunda lista. No entanto, as semelhanças das duas abordagens terminam neste ponto. Nesta nova abordagem a comparação das palavras se faz letra a letra, com a cabeça da fita indo e voltando repetidas vezes de uma lista para a outra. O novo processo geral de funcionamento é descrito pelo algoritmo 2.

Algoritmo 2: Processo Geral da Solução Proposta

```
1  Para todas as palavras p da primeira lista
2      Para todas as palavras q da segunda lista
3          Para todas as letras m de p e n de q
4              Se m de p é igual a n de q
5                  Continua (avança as letras m e n)
6              Senão
7                  Vai para a próxima palavra da segunda lista
8          Se chegou até ao fim das letras de p e q
9              Aceita a palavra p e passa para a próxima palavra da primeira lista
10     Se chegou ao fim da segunda lista sem que p fosse igual a q
11         Rejeita a palavra p e passa para a próxima palavra da primeira lista
12 Apaga a segunda lista
13 Contabiliza o número de palavras
```

4.1 Diferenças da solução simplista

A primeira diferença nesta nova solução é uma mudança no formato de entrada que agora marca com a sequência #z# também o início da primeira lista. Com isso, a mesma lista exemplo, que antes era: #aab#bbb#bba#z#bba#aab#z#, passa a ser representada por: #z#aab#bbb#bba#z#bba#aab#z#.

Também é possível com esta segunda solução tratar palavras que tenham mais do que três letras, pois faz a comparação de palavras letra a letra conforme descrito no algoritmo 2. Outra diferença é que agora foram incluídos dois novos símbolos para "representar" versões alteradas das letras a e b, que são os símbolos g, que representa um a reconhecido, e h, que representa um b reconhecido.

A última diferença é que esta nova solução completa o objetivo do trabalho, pois, após identificar as palavras presentes nas duas listas, calcula a cardinalidade da intersecção das listas. Essa alteração muda um pouco o formato de saída, que agora traz, antes das palavras encontradas em ambas as listas, dois novos símbolos, resultantes do cálculo da intersecção (em binário do tamanho desta lista). Os dois novos símbolos (g e h) são reaproveitados para representar em binário o número de palavras restantes na intersecção das listas. Para isso utiliza-se g como o valor zero e h como o valor um. Por exemplo, a sequência de símbolos hgh equivale ao número binário 101, ou seja, 5 em decimal.

O resultado final é que para uma entrada:

#z#aa#bbbb#bba#z#bba#aab#bbbb#b#aaa#bb#z#

teremos como saída:

hg#z####bbbb#bba#z#

onde, hg representa o número de palavras na intersecção (10 em binário, 2 em decimal), seguido das palavras $bbbb$ e bba, que estavam presentes nas duas listas.

4.2 Implementação

A implementação do algoritmo 2 (a solução proposta) é feita pela máquina de Turing apresentada na Figura 7. Nesta máquina, faz-se uso das seguintes submáquinas:

- *inicial*, que inicializa as listas marcando a primeira palavra de cada uma das listas;
- *prox_let*, que posiciona a cabeça da fita na próxima letra da palavra que está sendo analisada na segunda lista;
- *prox_pal*, que pula para a próxima palavra da segunda lista;
- *volta_pal*, que retorna para a letra a ser verificada da palavra da primeira lista que está sendo analisada;
- *reinicia_pal*, que retorna para a primeira letra da palavra da primeira lista que está sendo analisada;
- *aceita_pal*, que marca como aceita a palavra da primeira lista encontrada na segunda lista e marca a próxima palavra da primeira lista;
- *recusa_pal*, que apaga a palavra da primeira lista que não foi encontrada na segunda lista;
- *final*, que apaga a segunda lista após todas as palavras da primeira lista terem sido procuradas; e
- *contabil*, que conta quantas palavras da primeira lista foram aceitas, isto é, a cardinalidade da intersecção das listas.

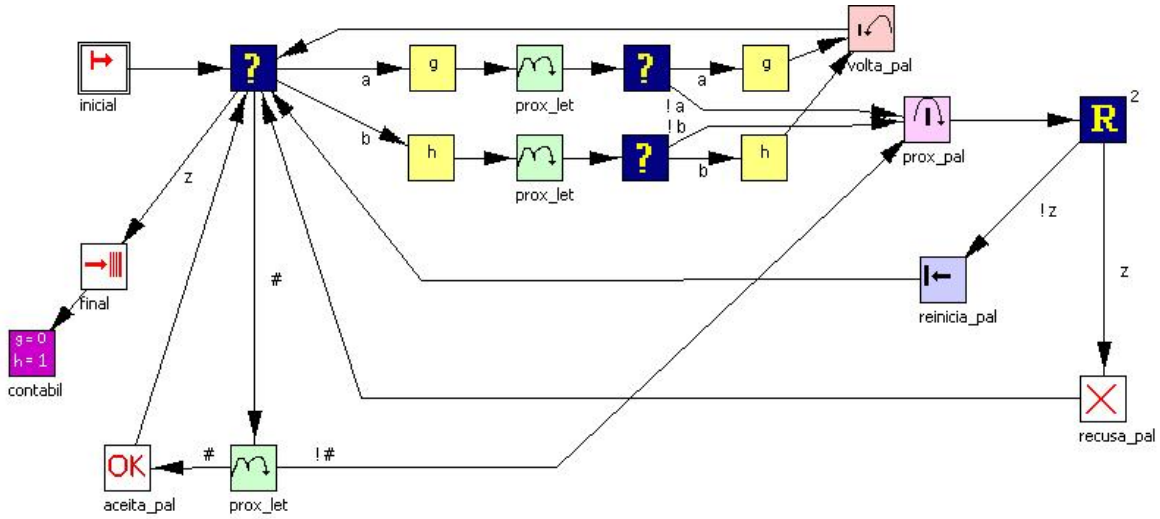


Figura 7. Máquina de Turing que implementa a solução proposta no algoritmo 2.

4.2.1 Submáquina *inicial*

A submáquina *inicial* (Figura 8) percorre toda a primeira lista (até encontrar o seu fim, marcado pelo símbolo z). Os primeiros três movimentos da cabeça da fita para a direita servem para "pular" a sequência #z# no início da lista. Encontrado este símbolo, avança-se a cabeça da fita e, na sequência, troca-se o símbolo # por um símbolo z, que marca então a primeira letra da primeira palavra da segunda lista.

Depois disso a submáquina move, repetidamente, a cabeça da fita para a esquerda até encontrar, novamente, o símbolo z, que marca a separação entre as listas, e continua movendo a cabeça da fita para a esquerda até encontrar novamente um símbolo z que marca o início da primeira lista. A cabeça da fita move-se novamente para

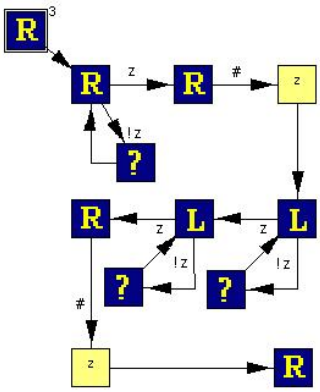


Figura 8. Submáquina responsável por marcar a primeira letra das primeiras palavras da primeira e segunda listas.

a direita e troca o símbolo # por um símbolo z que marca então a primeira letra da primeira palavra da primeira lista. Finalmente, a cabeça da fita avança para a direita e a comparação da primeira palavra da primeira lista com a primeira palavra da segunda lista poderá começar.

4.2.2 Submáquina *prox_let*

Esta submáquina (Figura 9) avança a cabeça da fita até a próxima letra a ser verificada na palavra que está sendo analisada na segunda lista. O funcionamento desta submáquina consiste em mover a cabeça da fita para a direita até encontrar o final da primeira lista (símbolo z). Em seguida, o avanço da cabeça da fita continua até encontrar a última letra da palavra da segunda lista que está sendo analisada, ou seja, até encontrar a última letra que vem após a marcação da palavra (símbolo z) ou uma letra já analisada (símbolos g ou h).

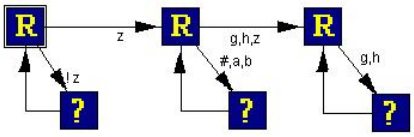


Figura 9. Submáquina responsável por posicionar a cabeça da fita na próxima letra da palavra que está sendo analisada na segunda lista.

4.2.3 Submáquina *prox_pal*

A submáquina *prox_pal* (Figura 10) tem como objetivo "pular" para a próxima palavra da segunda lista, ou seja, desmarcar a palavra atual, revertendo suas letras analisadas de g e h para a e b, e marcar a próxima palavra da lista. Esse procedimento é feito movendo a cabeça da fita para a esquerda trocando os símbolos g por a e h por b até encontrar o início da palavra marcada pelo símbolo z. Este símbolo é revertido para # para desmarcar esta palavra, e a cabeça da fita avança para a direita até o fim da palavra marcado pelo símbolo #. Este é substituído pelo símbolo z para marcar a próxima palavra da lista.

4.2.4 Submáquina *volta_pal*

Esta submáquina retorna à cabeça da fita para após a última letra já verificada na palavra da primeira lista que está sendo analisada. O funcionamento da submáquina *volta_pal* consiste em mover a cabeça da fita até o início da palavra da segunda lista sendo analisada (indicado pelo símbolo z). Após, a submáquina continua até o início da segunda lista (indicado novamente pelo símbolo z) e segue até encontrar a última letra já verificada

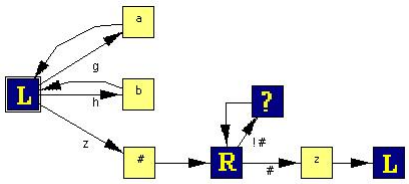


Figura 10. Submáquina responsável por "pular" para a próxima palavra da segunda lista.

da palavra da primeira lista analisada (símbolos g ou h) ou o seu início (indicado pelo símbolo z). Finalmente, a cabeça da fita avança para a direita para posicionar na próxima letra a analisar.

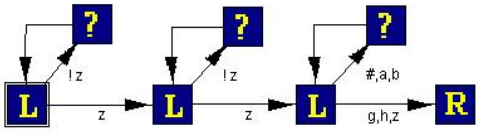


Figura 11. Submáquina responsável por retornar para a letra a ser verificada da palavra da primeira lista.

4.2.5 Submáquina *reinicia_pal*

Esta submáquina (Figura 12) reverte as letras já verificadas da palavra da primeira lista que está sendo analisada e reposiciona a cabeça da fita na primeira letra desta palavra. Isso é feito retornando a cabeça da fita até o início da palavra da segunda lista analisada (marcado pelo símbolo z), o início da segunda lista (marcado de novo pelo símbolo z). A partir daí, continua-se movendo a cabeça da fita para a esquerda, trocando-se os símbolos g por a e h por b até encontrar o início da palavra da primeira lista que está sendo analisada.

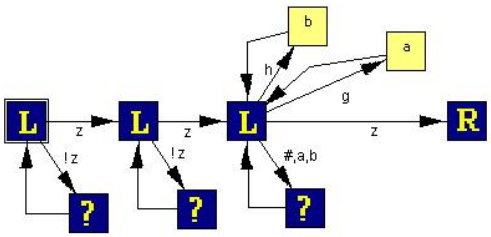


Figura 12. Submáquina responsável por retornar para a primeira letra da palavra da primeira lista.

4.2.6 Submáquina *aceita_pal*

A submáquina ilustrada na Figura 13 é chamada quando uma palavra foi encontrada na primeira e na segunda lista. Esta submáquina deve reverter as letras verificadas das palavras na primeira e na segunda lista, além de indicar a próxima palavra da primeira lista e a primeira palavra da segunda lista. Assim, esta submáquina prepara a fita para que a próxima palavra da primeira lista possa ser procurada.

Conforme indicado na Figura 13, as quatro tarefas a serem executadas são (1) reverter as letras da palavra da segunda lista que estava sendo analisada, ou seja, mover a cabeça da fita para a esquerda trocando g e h por a e b até encontrar o símbolo z, que deve ser trocado pelo símbolo #; (2) continuar movendo a cabeça da fita para a esquerda até o início da segunda lista, onde o símbolo z deve ser escrito para marcar a primeira palavra da segunda lista; (3) reverter as letras da palavra da primeira lista que estava sendo analisada, ou seja, mover a cabeça da fita para a esquerda trocando g e h por a e b até encontrar o símbolo z, que deve ser trocado pelo símbolo #; (4) mover

a cabeça da fita para a direita até o final desta palavra que estava sendo analisada e foi aceita, ou seja, avançar até o próximo símbolo #, que deve ser trocado pelo símbolo z para apontar para a próxima palavra da primeira lista.

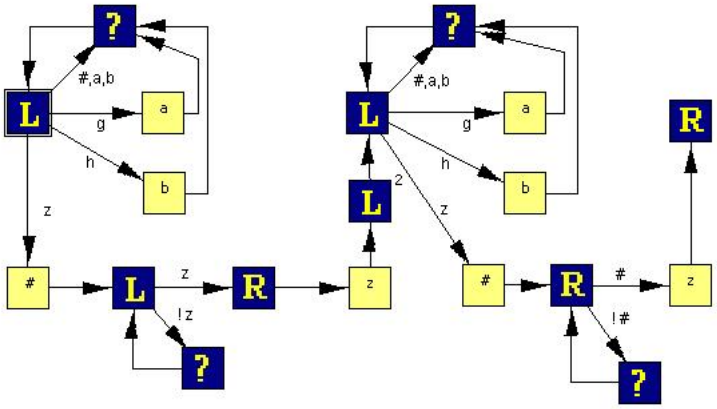


Figura 13. Submáquina responsável por marcar como aceita a palavra da primeira lista encontrada na segunda lista, marcar a próxima palavra da primeira lista e a primeira palavra da segunda lista.

4.2.7 Submáquina *recusa_pal*

A submáquina *recusa_pal*, apresentada na Figura 14, faz uma tarefa semelhante à submáquina *aceita_pal*, porém neste caso, ao invés de aceitar a palavra da primeira lista que estava sendo analisada, a palavra é recusada, ou seja, é apagada da primeira lista. Assim, esta submáquina também volta ao início da segunda lista, ou seja, move a cabeça da fita para a esquerda até encontrar o símbolo z e aponta novamente para a primeira palavra da segunda lista, escrevendo mais um símbolo z. Em seguida, a submáquina continua movendo a cabeça da fita para a esquerda até chegar à palavra que estava sendo analisada (próximo símbolo z) e começa a apagar todas as letras desta palavra, ou seja, escreve o símbolo # até encontrar, avançando para a direita, um símbolo #. Finalmente, este símbolo # é substituído pelo símbolo z, que indica a próxima palavra da primeira lista a analisar.

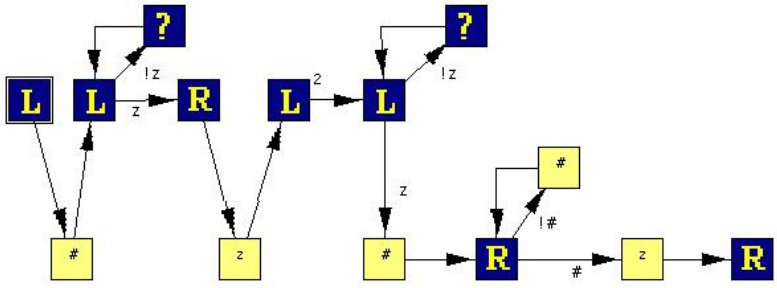


Figura 14. Submáquina responsável por apagar a palavra da primeira lista que não foi encontrada na segunda.

4.2.8 Submáquina *final*

Esta submáquina (Figura 15) é chamada quando todas as palavras da primeira lista foram analisadas e aceitas ou recusadas. A execução da submáquina *final* apaga toda a segunda lista, ou seja, escreve o símbolo # e avaną a cabeça da fita até encontrar o final da segunda lista (o próximo símbolo z). Depois, move a cabeça da fita para a esquerda até o início da primeira lista, ou seja, até encontrar o segundo símbolo z.

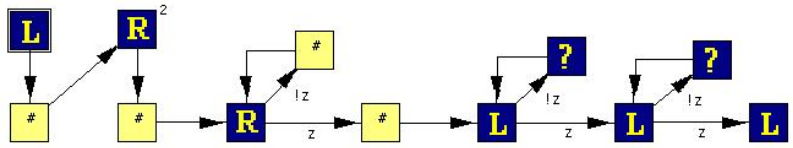


Figura 15. Submáquina responsável por apagar a segunda lista, após a procura das palavras da primeira lista.

4.2.9 Submáquina *contabil*

Após determinar as palavras componentes da intersecção das duas listas, a submáquina *contabil* (Figura 16) conta o número de palavras que foram aceitas, ou seja, as palavras que não foram apagadas. Esta máquina avança a cabeça da fita para a direita e a cada nova palavra encontrada, ou seja, cada vez que uma letra a ou b é encontrada após um símbolo #, uma submáquina chamada *mais_um* é executada para incrementar um contador binário. Finalmente, a cabeça da fita retorna para o início da lista.

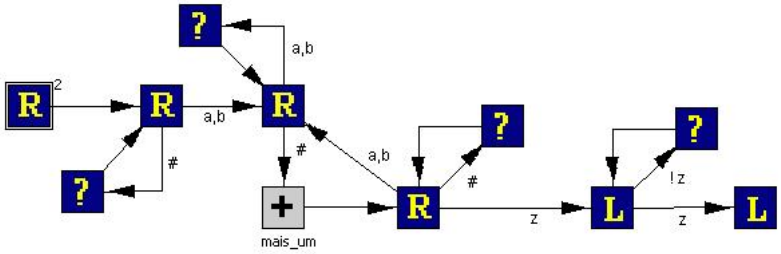


Figura 16. Submáquina responsável por contar quantas palavras da primeira lista foram aceitas, isto é, a cardinalidade da intersecção das listas.

4.2.10 Submáquina *mais_um*

Esta submáquina (Figura 17) faz um incremento em um contador binário que marca onde está a cabeça da lista com o símbolo z. Em seguida, recua a esquerda até uma posição após o início da lista e soma mais a um contador binário. Finalmente, retorna ao ponto original marcado pelo símbolo z e transforma-o novamente em #. Neste contador, um dígito 0 (representado pelo símbolo g) é incrementado, ou seja, é transformado em dígito 1 (representado pelo símbolo h); já um dígito 1 é incrementado sendo transformado em 0, mas o dígito da esquerda deve ser, por sua vez, também incrementado.

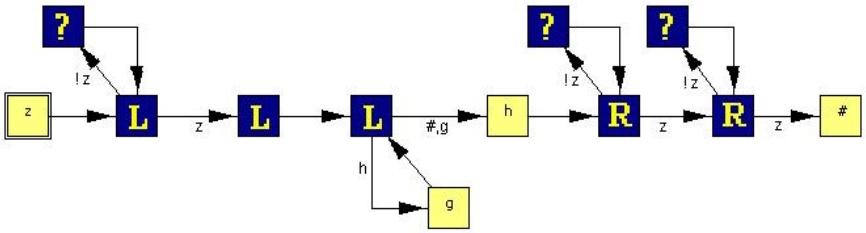


Figura 17. Submáquina responsável por somar *mais um* em um contador binário à esquerda da lista de aceitos.

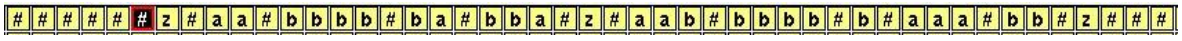
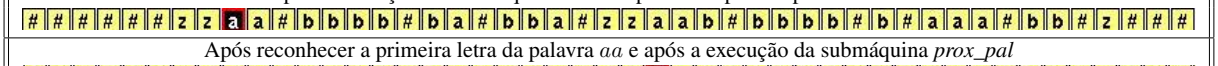
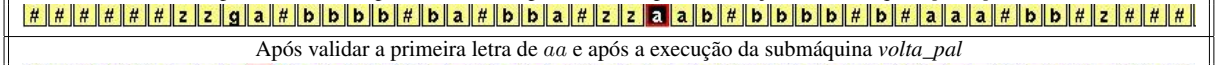
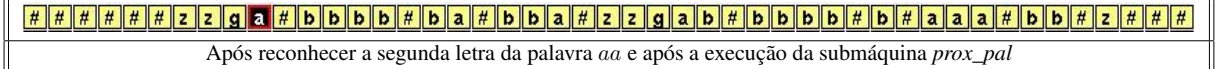
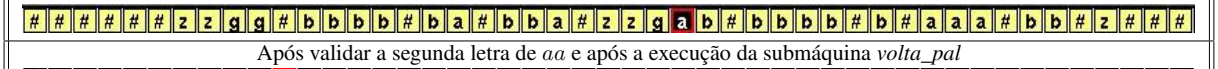
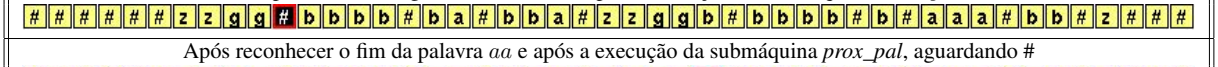
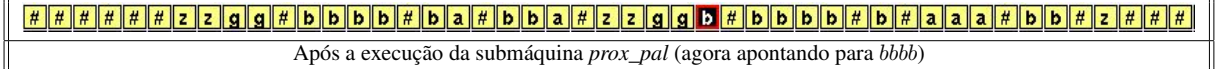
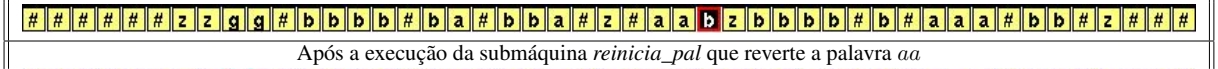
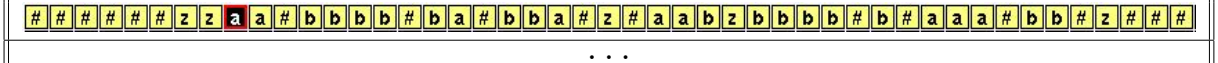
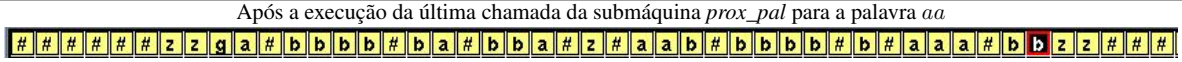
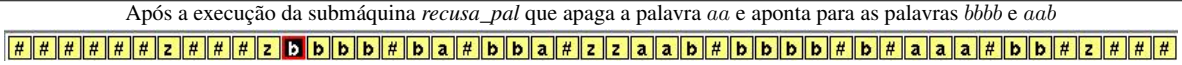
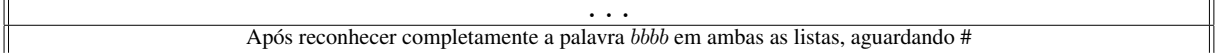
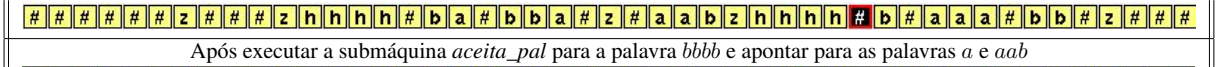
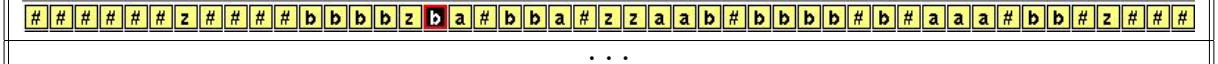
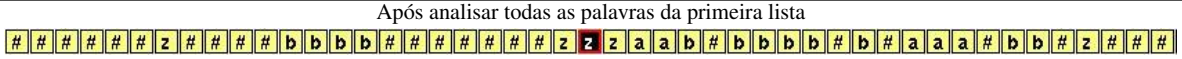
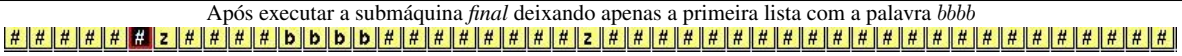
5 Exemplos de Aplicação

A Tabela 1 apresenta um exemplo de utilização com uma descrição detalhada de diversos passos. Neste exemplo recebem-se como entrada as listas:

$$aa \;\; bbbb \;\; ba \;\; bba \qquad \text{e} \qquad aab \;\; bbbb \;\; b \;\; aaa \;\; bb$$

A intersecção dessas listas possui apenas uma palavra: a palavra *bbbb*.

Tabela 1. Descrição passo a passo do primeiro exemplo de utilização.

Fita de Entrada

Após a execução da submáquina *inicial* apontando para as palavras aa e aab

Após reconhecer a primeira letra da palavra aa e após a execução da submáquina *prox_pal*

Após validar a primeira letra de aa e após a execução da submáquina *volta_pal*

Após reconhecer a segunda letra da palavra aa e após a execução da submáquina *prox_pal*

Após validar a segunda letra de aa e após a execução da submáquina *volta_pal*

Após reconhecer o fim da palavra aa e após a execução da submáquina *prox_pal*, aguardando #

Após a execução da submáquina *prox_pal* (agora apontando para *bbbb*)

Após a execução da submáquina *reinicia_pal* que reverte a palavra aa

. . .
Após a execução da última chamada da submáquina *prox_pal* para a palavra aa

Após a execução da submáquina *recusa_pal* que apaga a palavra aa e aponta para as palavras *bbbb* e aab

. . .
Após reconhecer completamente a palavra *bbbb* em ambas as listas, aguardando #

Após executar a submáquina *aceita_pal* para a palavra *bbbb* e apontar para as palavras a e aab

. . .
Após analisar todas as palavras da primeira lista

Após executar a submáquina *final* deixando apenas a primeira lista com a palavra *bbbb*

Após executar a submáquina *contabil* contando apenas uma palavra (número binário 1 - h)


Um outro exemplo de utilização é a comparação das listas:

$$a \;\; ab \;\; aaa \;\; abab \;\; bbbb \;\; bb \;\; b \;\; ba \;\; baa \qquad \text{e} \qquad ab \;\; abab \;\; bb \;\; baa \;\; a$$

Este segundo exemplo tem como intersecção as cinco palavras da segunda lista:

$$ab \;\; abab \;\; bb \;\; baa \;\; a$$

O resultado da execução na Tabela 2 mostra exatamente esse resultado, com a indicação do número binário 101 (5 em decimal) com os símbolos hgh e, em seguida as, palavras da intersecção.

Tabela 2. Descrição do resultado final do segundo exemplo de utilização.

6 Conclusão

A primeira máquina implementada é uma solução limitada que necessita que todas as palavras tratáveis estejam representadas nas instruções que identificam as palavras. Por isso esta solução difilmente pode ser estendida para um alfabeto maior.

A segunda solução apresentada é uma máquina bastante flexível, que pode ser estendida para um alfabeto maior do que apenas as letras a e b; tem como vantagem também o fato de que palavras de qualquer tamanho podem ser tratadas. Além disso, junto com a possibilidade de visualizar as palavras na intersecção, a máquina implementada mostra a cardinalidade do conjunto de intersecção, o que facilita a interpretação do resultado quando as listas possuirem um número grande de palavras.

A única limitação de uso desta máquina é o tempo de execução em listas grandes. O problema proposto é uma operação complexa, que muitas vezes é lenta, mesmo utilizando linguagens de programação. Logo, a demora de execução para listas grandes na máquina de Turing proposta não deve ser um problema de implementação, mas uma característica do próprio problema escolhido.

Referências

[1] TURING, A. M. *On computable numbers, with an application to the entscheidungsproblem*. Proc. London Math. Soc., vol. 2, no. 42, pp. 230-265, 1936.

[2] HERKEN, R. *The universal Turing machine: a half-century survey*. 2nd edition. Oxford Science, 1995.

[3] COHEN, D. I. A. *Introduction to computer theory*. 2nd edition. John Wiley & Sons, Inc, 1996.

[4] HODGES, A. *Alan Turing: one of The Great Philosophers*. Phoenix, London, 1997.

[5] MENEZES, P. F. B.; DIVERIO, T. A. *Teoria da Computação: Máquinas Universais e Computabilidade.* 2.ed. Porto Alegre: Sagra Luzzatto, 2003.

[6] *Visual Turing Machine*. On-line. Disponível em: *http://sourceforge.net/projects/visualturing/*. Acesso em: 23/04/2010.