Lucas Viana

Simulador Autômato com Pilha

Arquivos:

- main.c: Arquivo com função main, apenas execução dos códigos.
- lib.h: Header com structs e assinatura das funções utilizadas.
- lib.c: Implementação de todas funções do programa.
- automato.txt: Exemplo do formato de autômato a ser seguido.
- entradas.txt: Arquivo com entradas a serem testadas pelo simulador.

Compilação e execução do código:

```
gcc -o simulador main.c lib.c

./simulador automato.txt entradas.txt
```

Formato do Autômato:

Alfabeto: { a, b }
Estados: { s0, s1, s2, s3, s4 }
Estado inicial: s4
Finais: { s4 }
Auxiliar: { a, b }
Transicoes:
1: [s0, s0, a, $, a]
2: [s0, s1, b, a, $]
3: [s1, s1, b, a, $]
4: [s1, s0, a, $, a]
5: [s1, s4, $, $, $]
6: [s2, s2, b, $, b]
7: [s2, s3, a, b, $]
8: [s3, s3, a, b, $]
9: [s3, s2, b, $, b]
10: [s3, s4, $, $, $]
11: [s4, s0, a, $, a]
12: [s4, s2, b, $, b]

- Sendo transições formadas por:

<identificação>: [<estado atual>, <prox estado>, <fita>, <desempilha>, <empilha>]

- Deve seguir exatamente o formato indicado, caso contrário pode causar falha no reconhecimento do autômato. Cuidar com linhas vazias e diferenciação entre letras maiúsculas/minúsculas.
- $ representa o símbolo vazio (Epsilon)

Exemplo entradas:

ab
abab
ababab
ba
aaabbb
aabbbbbaaa

a
b
aba
bb
ababb

- Linha vazia representa a palavra vazia

Estrutura do programa:

```
typedef struct Estado {
  char *id;
  int final;
  Transicao *transicoes;
} Estado;
```

Representa um estado, guarda id (nome), se é um estado final e uma lista encadeada de transições.

```
typedef struct Transicao {
  char *fita;
  char *desempilha;
  char *empilha;
  struct Estado *proximoEstado;
  struct Transicao *prox;
} Transicao;
```

Representa transição, tendo os símbolos da transição (fita, pilha, pilha), para qual estado a transição leva e um ponteiro para próxima transição formando uma lista encadeada.

```
typedef struct AP {
  char **alfabeto;
  char **auxiliar;
  HashTable *estados;
  Estado *inicial;
  Pilha *pilha;
} AP;
```

Representa a sêxtupla, com exceção dos estados finais e transições, ambos registrados na struct Estado.

Lógica do Programa:

Simulação é feita por função recursiva:

*int simulaAutomatoRecursivo(Estado *estadoAtual, Pilha *pilha, char *entrada, int i);*

A cada chamada, itera sobre todas as transições do estado atual, comparando símbolos da fita/pilha com símbolos da transição atual. Ao encontrar uma transição válida chama recursivamente incrementando a posição da fita (i), exceto em transições vazias (ε, ε, ε), em que a posição não é alterada. Quando encontra uma transição que aceita a palavra retorna 1 e a palavra é aceita, caso contrário retorna 0 e palavra é recusada.

Dessa forma, passa por todas transições independentemente de já ter encontrado uma transição válida e possibilita a análise de autômatos não determinísticos.

Comentários:

- O Código possui prints com execução de cada transição, porém estão todos comentados, pois o autômato pode ter não-determinismo e gerar muitas transições causando poluição.

- Todos símbolos possuem tipo (char *) para alocação dinâmica. Por isso alfabetos (de entrada e pilha) são representados por (char **).

Exemplo de execução do programa: