Universidade Federal de Ouro Preto - UFOP
Gabriel Geovanni Moreira Silva - 20.1.8099
Matheus Oliveira Vilela - 20.1.8127
Ian Langkammer Batista - 20.1.8008
Leonardo Moreira - 20.1.8006

1 - Introdução

O objetivo do trabalho é desenvolver um programa que permite a manipulação de linguagens formais através de reconhecedores finitos (AFDs) através de um conjunto de funcionalidades, sendo eles: visualização, complemento, interseção, união e reconhecimento de palavras. Ao longo da próxima sessão serão abordados com maior especificidade o funcionamento de cada um dos métodos supracitados, que são os nossos respectivos problemas que o algoritmo visou solucionar.

2 - Solução do problema

2.1 - Visualização

Para a aplicação da funcionalidade de visualização criamos uma função "*converterParaDot"* a qual recebe como parâmetro o arquivo com o AFD e o nome do arquivo de saída que são informados pelo usuário. A função utiliza a função *fopen(nomeArquivo, "w")* para indicar que será para escrita. Em seguida, insere no arquivo as informações necessárias para configuração do arquivo "*.dot*".

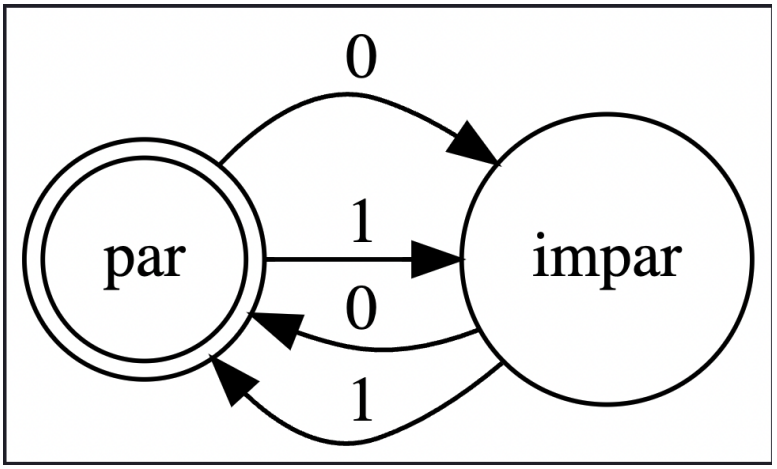
Após inserir as configurações necessárias no arquivo de saída, é instanciado ponteiros para armazenar cada caracter das transições do AFD utilizado. Posteriormente, o código entra em um laço de repetição para separar os caracteres e armazená-los em seus respectivos ponteiros. Em seguida, os caracteres são escritos no arquivo no formato "*estado -> estado [label = símbolo];"*

Por último, o arquivo é salvo, com o nome passado pelo usuário, no diretório indicado na flag "*--output"*.

afd.txt

```
2
par
impar
2
0
1
4
par 0 impar
par 1 impar
impar 0 par
impar 1 par
par
1
par
```

afd.svg



Universidade Federal de Ouro Preto - UFOP
Gabriel Geovanni Moreira Silva - 20.1.8099
Matheus Oliveira Vilela - 20.1.8127
Ian Langkammer Batista - 20.1.8008
Leonardo Moreira - 20.1.8006

2.2 - Complemento

Para a aplicação do complemento criamos a função "*converterParaTxt"* a qual, assim como a função da funcionalidade anterior, recebe o arquivo com o AFD e o nome do arquivo de saída.

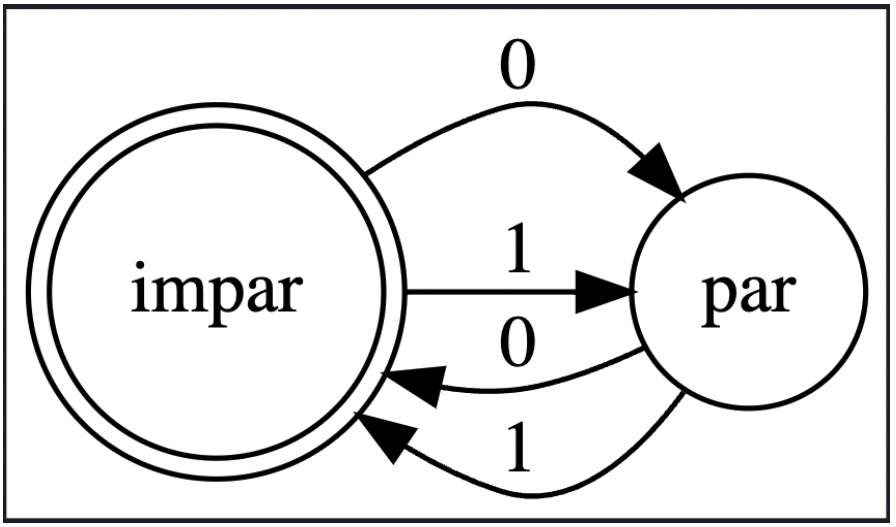
A parte inicial da função ocorre da mesma forma demonstrada em visualização, utilizando a função "*fopen(nomeArquivo, "w")"*. Os processos seguintes da função são laços de repetição que são responsáveis por escrever no arquivo a quantidade de estados e quais são, a quantidade de símbolos no alfabeto e os símbolos e a quantidade de transições e as transições logo em seguida.

Terminando essa sequência, o estado inicial é escrito no arquivo e se inicia dois laços de repetição para validar e mudar os estados finais pelos estados não finais. Por fim, salva-se o arquivo, como na função anterior.

afd.txt

```
2
par
impar
2
0
1
4
par 0 impar
par 1 impar
impar 0 par
impar 1 par
par
1
impar
```

afd.svg



Universidade Federal de Ouro Preto - UFOP
Gabriel Geovanni Moreira Silva - 20.1.8099
Matheus Oliveira Vilela - 20.1.8127
Ian Langkammer Batista - 20.1.8008
Leonardo Moreira - 20.1.8006

2.3 - Interseção

Para a aplicação da interseção criamos a função "calcular Interseção" a qual recebe dois afds como parâmetro e um arquivo de saída, Após isso ela junta os estados dos afds, junta os dois alfabetos e realiza o produto dos afds, então calculamos o novo estado inicial recebendo o estado que contém os dois estados iniciais dos afds, calcula a interseção dos estados finais dos dois afds e mantém apenas os que são finais nos dois afds originados do produto de afds entre esses cálculos já são realizadas as inserções no arquivo de saída utilizando a função “fprintf” do c.

afd1.txt

```
2
C
D
2
0
1
4
C 1 D
C 0 C
D 1 C
D 0 D
C
1
C
```

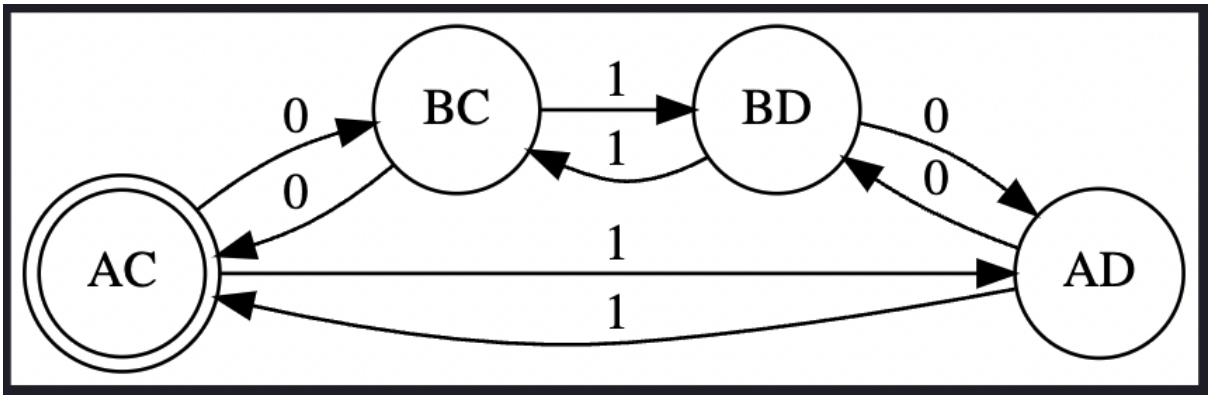
afd2.txt

```
2
A
B
2
0
1
4
A 0 B
A 1 A
B 0 A
B 1 B
A
1
A
```

afd1-intersecao-afd2.txt

```
4
AC
AD
BC
BD
2
0
1
8
AC 0 BC
AD 0 BD
AC 1 AD
AD 1 AC
BC 0 AC
BD 0 AD
BC 1 BD
BD 1 BC
AC
1
AC
```

afd1-intersecao-afd2.svg



Universidade Federal de Ouro Preto - UFOP
Gabriel Geovanni Moreira Silva - 20.1.8099
Matheus Oliveira Vilela - 20.1.8127
Ian Langkammer Batista - 20.1.8008
Leonardo Moreira - 20.1.8006

2.4 - União

Para a União de afds, criamos a função “calcularUniao” que recebe dois afds como entrada e um arquivo de saída, após isso criamos um novo afd que será a união dos dois, juntamos os estados utilizando o produto de afds, após isso calculamos os estados que contém algum dos estados finais anteriores e os adicionamos no conjunto de estados finais do novo afd, no final juntamos os símbolos do alfabeto e o novo estado inicial que será o estado que contém o inicial dos dois afds então escrevemos esses dados no arquivo de saída e o fechamos.

afd1.txt

```
2
A
B
2
0
1
4
A 0 B
A 1 A
B 0 A
B 1 B
A
1
A
```

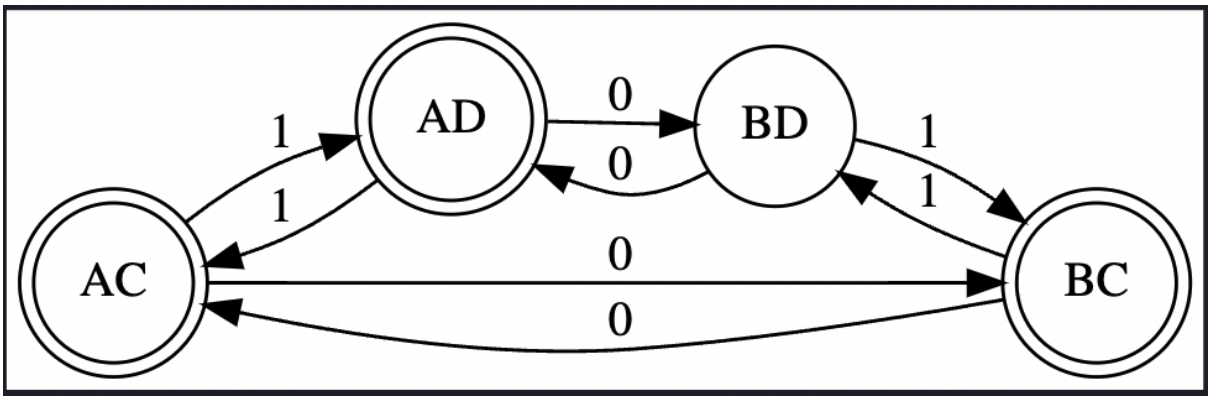
afd2.txt

```
2
C
D
2
0
1
4
C 1 D
C 0 C
D 1 C
D 0 D
C
1
C
```

afd1-uniao-afd2.txt

```
4
AC
AD
BC
BD
2
0
1
8
AC 0 BC
AD 0 BD
AC 1 AD
AD 1 AC
BC 0 AC
BD 0 AD
BC 1 BD
BD 1 BC
AC
3
AC
AD
BC
```

afd1-uniao-afd2.svg



Universidade Federal de Ouro Preto - UFOP
Gabriel Geovanni Moreira Silva - 20.1.8099
Matheus Oliveira Vilela - 20.1.8127
Ian Langkammer Batista - 20.1.8008
Leonardo Moreira - 20.1.8006

2.5 - Reconhecimento

A priori, dentro da aplicação foi criado o arquivo “reconhecer.c” com a “reconhecer.h” e suas respectivas funcionalidades necessárias para plena execução. Passando os parâmetros corretamente na chamada do app, será chamada a função “reconhecer” que primeiramente vai validar os argumentos, ou seja, se todos os parâmetros de execução necessários estão de acordo.

A posteriori, se os argumentos estiverem coesos, essa função lê um arquivo de afd e um arquivo de palavras a serem reconhecidas e salva em suas respectivas variáveis. Além disso, também é necessário um argumento do tipo *string* que consiste no nome do arquivo *output* para *printar* se as palavras foram reconhecidas ou não. Em caso de argumentos inválidos, a operação é abortada.

Após todos os argumentos serem passados e validados, a função “conhecerPalavra” é chamada passando os parâmetros supracitados. Primeiramente a função cria o arquivo de *output* com o nome passado como arg, e se não for possível a operação é abortada. Enquanto o array de palavras passados como *arg* não for completamente consumido, a função fica presa em um laço de repetição que executa para cada palavra a função “reconhecerPalavra” que recebe um afd e uma palavra, escrevendo o retorno de “reconhecerPalavra” - que só pode ser 0 ou 1 - no arquivo de *output* criado posteriormente. No final da execução a instância de manipulação do arquivo é fechada.

Cabe ressaltar a forma de funcionamento da “reconhecerPalavra”, que estabelece o estado inicial como o estado inicial encontrado no afd, em seguida ela percorre cada símbolo da palavra verificando a transição correspondente e salvando os seus estados de destino. Finalmente, a função avalia se seu último estado é um estado final, ou seja, reconhecido pela linguagem e retorna 1, ou é um estado não final e retorna 0.

Universidade Federal de Ouro Preto - UFOP
Gabriel Geovanni Moreira Silva - 20.1.8099
Matheus Oliveira Vilela - 20.1.8127
Ian Langkammer Batista - 20.1.8008
Leonardo Moreira - 20.1.8006

3 - Guia de como usar o *software*

Primordialmente, a fim de utilizar o software é necessário compilar o código informando o comando *'make'* no terminal aberto no diretório principal do repositório, que irá gerar o arquivo *afdtool,* que é imprescindível para a execução de todos os métodos.

3.1 - Visualização

Nesta funcionalidade, ao utilizar o *afdtool* é necessário informar o arquivo com a descrição do AFD após a flag ''--dot'' e o nome do arquivo após a *flag* "--output" que será gerado como resultado.

Exemplo de execução no terminal:
$ ./afdtool --dot afd.txt --output afd.dot

No exemplo acima, é lúcido ressaltar que:

- "afd.txt" se refere ao AFD a ser visualizado;
- "afd.dot" é o nome do arquivo a ser gerado com a visualização do AFD.

Por fim, para gerar o grafo é necessário informar o seguinte comando no terminal:
$ dot -Tsvg afd.dot -o afd.svg

3.2 - Complemento

Nesta funcionalidade, ao utilizar o *afdtool* é necessário informar somente dois parâmetros. Primeiro, após a *flag* "--complemento" informar a descrição do AFD, e em segundo, após a *flag* "--output", informar o nome do arquivo do AFD complemento gerado pela função a partir do AFD informado.

Exemplo de execução no terminal:
$ ./afdtool --complemento afd.txt --output afd-complemento.txt

No exemplo acima, é fulcral pontuar que:

- "afd.txt" se refere ao AFD para realizar o complemento;
- "afd-complemento.txt" se refere ao AFD gerado que é o complemento do AFD informado posteriormente.

Universidade Federal de Ouro Preto - UFOP
Gabriel Geovanni Moreira Silva - 20.1.8099
Matheus Oliveira Vilela - 20.1.8127
Ian Langkammer Batista - 20.1.8008
Leonardo Moreira - 20.1.8006

3.3 - Interseção

Nesta funcionalidade, ao utilizar o *afdtool* é necessário informar dois arquivos com as descrições dos AFDs após a *flag* "--intersecao" e informar o nome do arquivo após a *flag* "--output" que será gerado com o resultado da interseção desses dois AFDs.

Exemplo de execução no terminal:
$ ./afdtool --intersecao afd1.txt afd2.txt --output afd1-intersecao-afd2.txt

No exemplo acima, é fulcral pontuar que:

- "afd1.txt" e "afd2.txt" são os AFDs informados para realizar a interseção;
- "afd1-intersecao-afd2.txt" é o nome do arquivo de *output* do resultado dessa interseção.

3.4 - União

Nesta funcionalidade, ao utilizar o *afdtool* é necessário informar dois arquivos com as descrições dos afds .txt após a flag "--uniao" e informar também o nome do arquivo após a flag "--output" que será gerado a união desses dois AFDs.

Exemplo de execução no terminal:
$ ./afdtool --uniao afd1.txt afd2.txt --output afd1-uniao-afd2.txt

No exemplo acima, é fulcral pontuar que:

- "afd1.txt" e "afd2.txt" são os AFDs informados para realizar a união;
- "afd1-uniao-afd2.txt" é o nome do arquivo de *output* do resultado dessa união.

3.5 - Reconhecimento

Nesta funcionalidade, ao utilizar o *afdtool* é necessário informar três parâmetros, primeiramente após a *flag* "--reconhecer" a descrição do AFD e na sequência o arquivo que contém as palavras a serem reconhecidas. Após a *flag* "--output" é mister informar o nome do arquivo que será gerado com os resultados do reconhecimento de cada palavra.

Exemplo de execução no terminal:
$ ./afdtool --reconhecer afd.txt palavras.txt --output palavras-reconhecidas.txt

No exemplo acima, é fulcral pontuar que:

- "afd.txt" se refere ao AFD para reconhecer as palavras;
- "palavras.txt" se refere ao arquivo com as palavras a serem reconhecidas;
- "palavras-reconhecidas.txt" se refere ao nome do arquivo que será gerado com o resultado da execução de reconhecimento.