Lista 2 — Introdução à Teoria da Computação; Fundamentos Matemáticos; Computação e Representação

Teoria da Computação / Linguagens Formais e Autômatos

Prof. Jefferson O. Andrade

IFES — Campus Serra

2024/2

1 Introdução

Para esta tarefa você deve resolver os problemas da Seção 3. A sua resposta deve ser preparada em um documento em LaTeX. A classe do documento deve ser `scrartcl`, com fonte tamanho 11. Cada exercício deve ser respondido em uma seção própria. Caso o exercício possua itens/partes, cada item/parte deve ser respondido em uma subseção.

A qualidade da formatação do seu relatório será um dos fatores considerados na correção. O LaTeX permite a criação de documentos extremamente elegantes, use o potencial da ferramenta.

2 Orientações

Execução: Os alunos de Linguagens Formais e Autômatos (graduação) podem fazer o trabalho em duplas, se desejarem. Os alunos de Teoria da Computação (mestrado) devem fazer o trabalho individualmente.

Colaboração: você pode colaborar com outros alunos que estão atualmente matriculados nesta disciplina em *brainstorming* e pensando em abordagens para soluções, mas você deve escrever as soluções por conta própria e não pode compartilhá-las com outros alunos.

Violações graves: compartilhar perguntas ou soluções com qualquer pessoa fora desta disciplina, incluindo postagem em sites externos, constitui uma violação do código de ética. Em particular, você não pode obter ajuda de alunos ou materiais de anos anteriores desta disciplina ou equivalente. Casos de **plágio** serão relatados à coordenação de curso e encaminhados ao conselho de ética.

Formato de submissão: Todos os problemas desta lista são problemas de programação que serão testadas pelo professor. Deste modo devem ser entregues dois artefatos: um relatório de solução, e os códigos fontes. Os dois artefatos devem ser entregues como **um único arquivo ZIP** (não é RAR, não é LHA, é **ZIP**). O arquivo ZIP deve obrigatoriamente ter a seguinte estrutura:

```
lista1
├── <<nome-do-relatório>>.pdf
└── problemas
    ├── probl1
    │   ├── README.md
    │   └── ...
    └── probl2
        ├── README.md
        └── ...
```

1. **Relatório de solução:** Contendo a solução de **todas** as questões, inclusive as que pedem implementação. As questões que pedem implementação de código como solução devem incluir um breve comentário sobre a ideia gera da solução e apontar para o diretório que contém o código fonte.

   O PDF submetido deve ser digitado no mesmo formato que este. Inclua o enunciado dos problemas (ou um resumo, se o enunciado for grande) e escreva "Solução do Problema X – Título do Problema X" antes da sua solução. Poderão ser deduzidos pontos se você enviar em um formato diferente.

2. **Códigos fontes:** Os códigos fontes, quando necessários para resolver os problemas, poderão ser desenvolvidos em qualquer linguagem, desde que sejam satisfeitas as seguintes condições:

   a) A linguagem deve estar disponível, de modo livre, para o sistema operacional Linux, mais especificamente o NixOS.

   b) Você deve incluir juntamente com a sua solução um arquivo de *build*, se for o caso, e instruções de execuções de execução do programa.

   Os códigos fontes de todos os problemas devem estar organizados dentro de um diretório chamado `problemas`. Dentro do diretório `problemas` deverá haver um subdiretório para cada problema, e.g., `probl1`. Dentro deste subdiretório haverá o(s) arquivo(s) com código fonte com a solução do problema. Para cada problema deve ser entregue o código fonte do programa que resolve o problema e um arquivo `README.md`, em formato Markdown, contendo:

   - Nome do autor.
   - Uma **breve** explicação do código que foi implementado.
   - Uma descrição de como executar as funções.

   Um exemplo do conteúdo do arquivo `README.md` é dados abaixo.

```
# Lista 1 -- Teoria da Computação / Linguagens Formais e Autômatos -- 2024/2

**Introdução à Teoria da Computação & Fundamentos Matemáticos**

**Autor:** Jefferson O. Andrade

## Problema 0

Este problema pede a construção de uma função que receba como entrada um número
$n$ representando o tamanho de uma cadeia de bits e retorne uma sequência com
todas as cadeias em binário com o tanho indicado.

**Solução**

A solução adotada foi implementar uma recursão. O caso base é quando o
comprimento é zero. A única cadeia de tamanho zero é a cadeia vazia. As cadeias
```

```
estão sendo representada como listas em Scala. Nos demais casos, gera-se todas
as cadeias de tamanho $n-1$ e em seguida gera-se duas semi-duplicadas, uma com 0
acrescentando à frente de cada cadeia, e outra com 1 acrescentado à frente de
cada cadeia. O resultado é uma lista contendo as duas listas com as
semi-duplicatas.

**Execução**

Para executar a função no prompt do Scala execute, por exemplo:

```scala
allBinStrings(5)
```

Para executar usando o SBT use:

```bash
sbt run
```
```

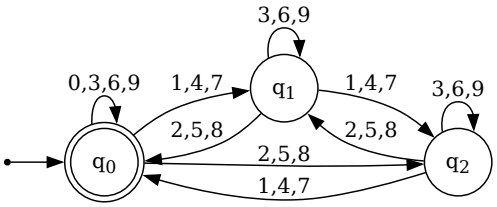
3 Problemas

Problema 1 — Múltiplos de 3, 5 ou 7 (*1 pts*)

Construa um NFA$_\varepsilon$ que reconheça a linguagem $L_1 = \{1^n \mid n \text{ é múltiplo de 3, 5 ou 7}\}$.

Problema 2 — Linguagem $L(M_2)$ (*2 pts*)

Descreva, em português, que linguagem é reconhecida pelo autômato M_2 abaixo:



Problema 3 — Regex Crossword (*4 pts*)

Parte I

Acesse o site Regex Crossword[1] e realize as seguintes atividades:

1. Resolva pelo menos 3 dos quebra-cabeças da seção *Puzzles*.
2. Resolva pelo menos 2 dos quebra-cabeças da seção *Challenges/Intermediate*
3. Resolva pelo menos 2 dos quebra-cabeças da seção *Challenges/Experienced*
4. Resolva pelo menos 2 dos quebra-cabeças da seção *Challenges/Hexagonal*

[1] `https://regexcrossword.com/`

Parte II

Acesse a seção *Player Puzzles* e crie um quebra-cabeças original seu. O seu quebra cabeças deve ter pelo menos 25 células, com dicas não triviais. Soluções que gerem mensagens criativas serão mais apreciadas.

Problema 4 — Regex Compiler (*8 pts*)

Escreva um programa que seja capaz de ler uma expressão regular pura[2] e gerar o autômato finito com movimentos vazios correspondente.

Para a sintaxe das expressões regulares, siga a seguinte definição:

Definition 1 (Expressão Regular)**.** *Uma expressão regular é definida sobre um alfabeto de entrada Σ. O conjunto de expressões regulares, ER, é definido indutivamente como:*

1. $\varnothing \in ER$.
2. $\varepsilon \in ER$.
3. *Se* $x \in \Sigma$, *então* $x \in ER$.
4. *Se* $r_1 \in ER$ *e* $r_2 \in ER$, *então* $r_1 r_2 \in ER$ *(concatenação).*
5. *Se* $r_1 \in ER$ *e* $r_2 \in ER$, *então* $r_1 \cup r_2 \in ER$ *(união).*
6. *Se* $r_1 \in ER$, *então* $r_1\star \in ER$ *(estrela).*

Represente os símbolos especiais da seguinte forma:[3]

Símbolo	Representação
$\varnothing$	`\null`
ε	`\empty`
$\cup$	`|`
$\star$	`*`

Depois de feita a conversão, o NFA$_\varepsilon$ gerado deve ser impresso da seguinte forma:[4]

- Exatamente uma linha contendo a string `Initial state:` e o nome do estado inicial.
- Exatamente uma linha contendo a string `Accepting states:` e os nomes dos estados de aceitação separados por espaço.
- Uma linha para cada transição entre dois estados no seguinte formato: `qi -> qj: x y z`. Onde, `qi` é o estado inicial da transição; `qj` é o estado final da transição; `x y z` são os símbolos que disparam a transição.
- Para transições vazias, use o símbolo `\empty`.

Por exemplo, para a expressão regular `(a|\empty)(ab|bb)*` seu programa deve imprimir um autômato equivalente a:[5]

[2] Uma "expressão regular pura" é uma expressão regular composta apenas pelas expressões regulares básicas (caracteres do alfabeto de entrada, vazio e expressão nula) e as operações básica (concatenação, união e estrela).

[3] Além destas definições, se o alfabeto de entrada contiver o caractere '\', você irá precisar representá-lo por '\\'.

[4] É imperativo que o formato abaixo seja seguido, porque a saída dos seu programa será lida por um outro programa, desenvolvido pelo professor, que irá validar a tradução.

[5] Note que o uso do padrão de nomes `qn` para os estados é meramente um exemplo, seu programa pode gerar outros nomes.

```
Initial state: q0
Accepting states: q16
q0 -> q1: \empty
q0 -> q3: \empty
q1 -> q2: a
q2 -> q4: \empty
q3 -> q4: \empty
q4 -> q5: \empty
q5 -> q6: \empty
q5 -> q16: \empty
q6 -> q7: \empty
q6 -> q11: \empty
q7 -> q8: a
q8 -> q9: \empty
q9 -> q10: \empty
q10 -> q15: \empty
q11 -> q12: b
q12 -> q13: \empty
q13 -> q14: b
q14 -> q15: \empty
q15 -> q6: \empty
q15 -> q16: \empty
```

Que corresponde ao seguinte NFA$_\varepsilon$: