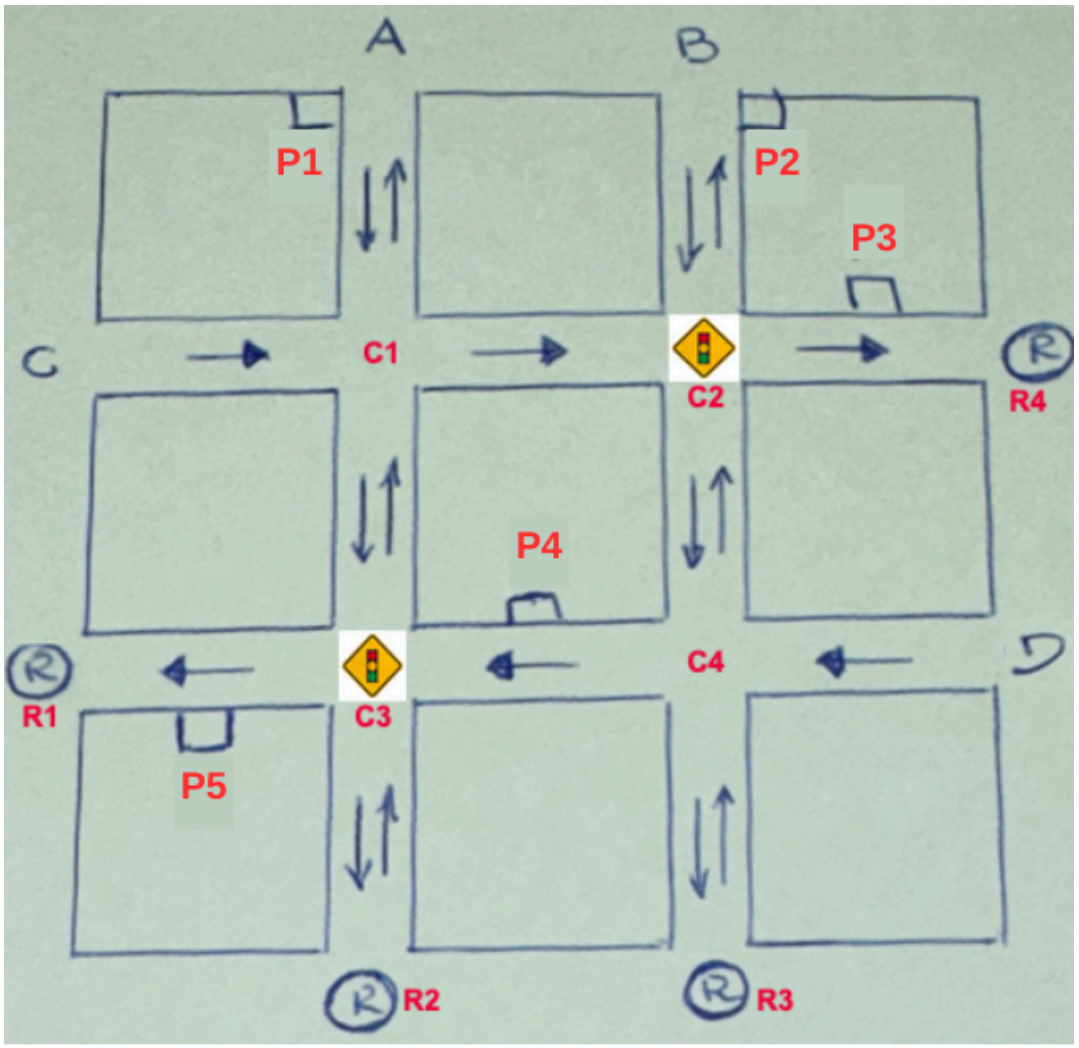
Relatório Final da Matéria Linguagens Formais e Autômatos

Professor: Ricardo Ferreira Martins.
Alunos: Gustavo de Souza; José Augusto Laube.

Objetivo principal: Desenvolver um conjunto de modelos para um sistema de trânsito simplificado, com base na imagem abaixo. O foco está em promover a simulação de tal sistema, baseado nos modelos de autômatos, estudados durante a disciplina (AFDs e APs).

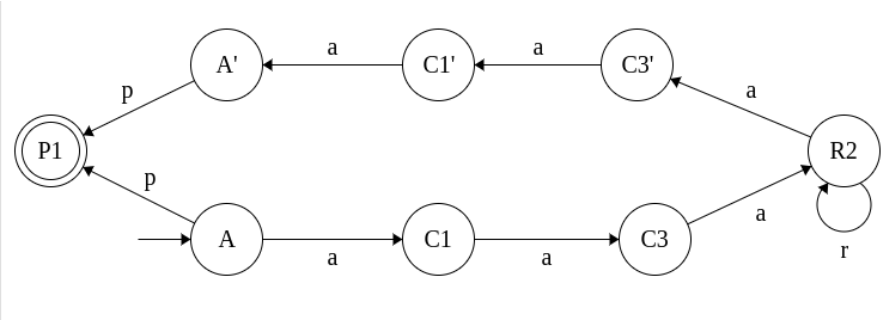


Modelos

O primeiro passo tomado para o desenvolvimento do trabalho foi a modelagem de autômatos finitos determinísticos que representassem os possíveis fluxos de um carro partindo de um ponto inicial selecionado pelo usuário (A, B, C ou D). Esses modelos não levam em conta curvas, semáforos ou os sensores posicionados nas vias, eles apenas representam o fluxo de um veículo até um estado final (estacionamento P). Os 4 modelos

permitem palavras de entrada que representam o fluxo do carro entre quarteirões, e as transições representam a movimentação de quarteirão por quarteirão, de quarteirão para rotatória, da rotatória para o quarteirão novamente (somente a modelagem A e B permitem) e do quarteirão para um estacionamento (desde que seja uma transição válida para um dado estado).

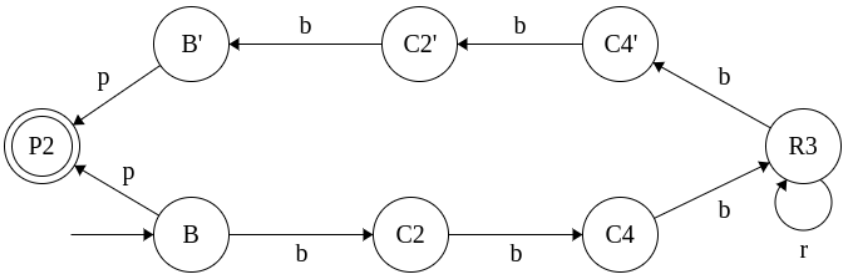
Fluxo contínuo do carro que inicia em A:



Q = ({P, a, r}, {P1, A, C1, C3, R2, C3', C1', A'}, S, {A}, {P1})

S(A, a) = (C1)
S(A, P) = (P1)
S(C1, a) = (C3)
S(C3, a) = (R2)
S(R2, r) = (R2)
S(R2, a) = (C3')
S(C3', a) = (C1')
S(C1', a) = (A')
S(A', P) = (P1)

Fluxo contínuo do carro que inicia em B:

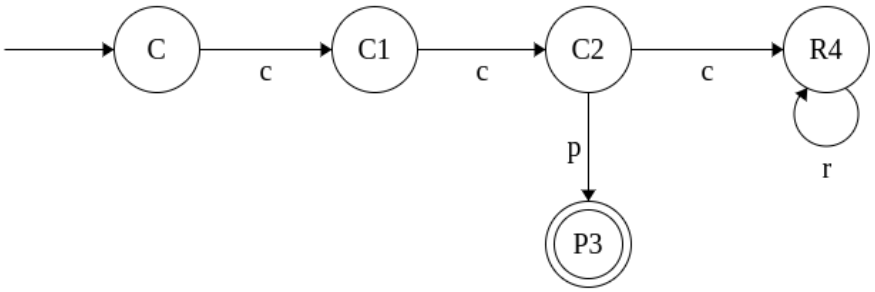


Q = ({P, b, r}, {P1, C2, C4, R3, C2', C4', B'}, S, {B}, {P2})

S(B, b) = (C2)
S(B, P) = (P2)
S(C2, b) = (C4)
S(C4, b) = (R3)
S(R3, r) = (R3)
S(R3, b) = (C4')

S(C4', b) = (C2')
S(C2', b) = (b')
S(B', P) = (P2)

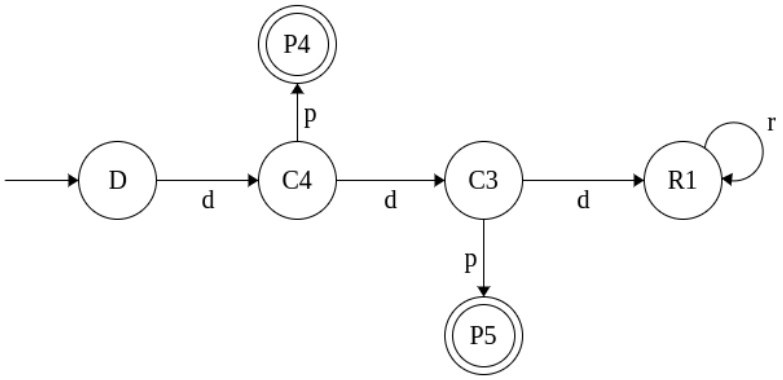
Fluxo contínuo do carro que inicia em C:



Q = ({P, c, r}, {P3, C, C1, C2, R4}, S, {C}, {P3})

S(C, c) = (C1)
S(C1, c) = (C2)
S(C2, c) = (R4)
S(C2, P) = (P3)
S(R4, r) = (R4)

Fluxo contínuo do carro que inicia em D:

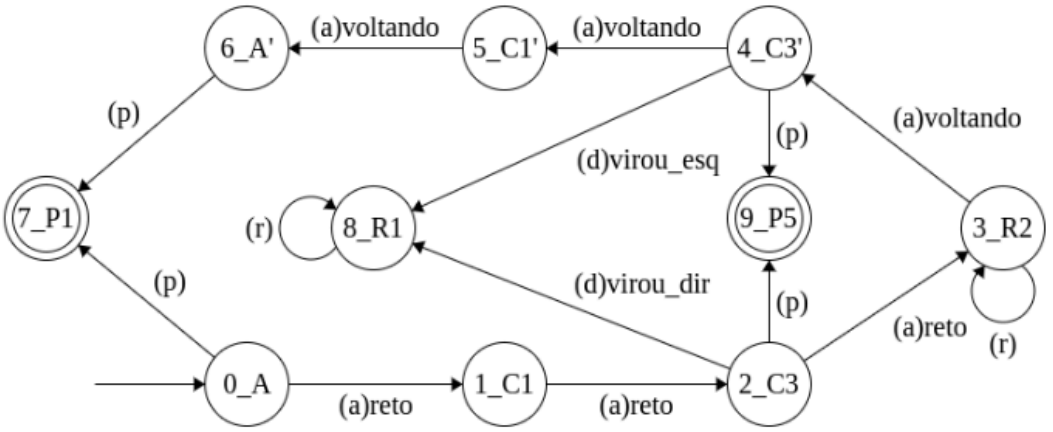


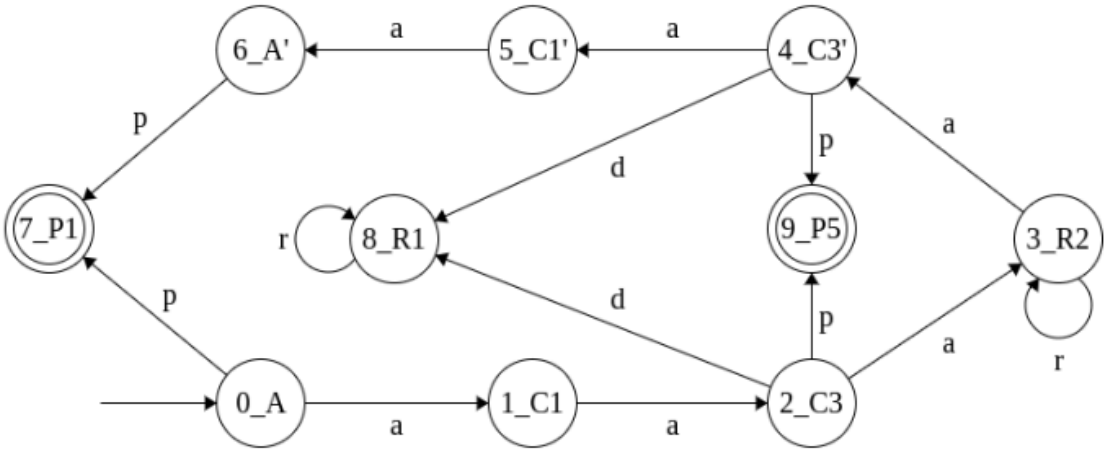
Q = ({P, d, r}, {P4, P5, D, C4, C3, R1}, S, {D}, {P4, P5})

S(D, d) = (C4)
S(C4, d) = (C3)
S(C3, d) = (R1)
S(C4, P) = (P4)
S(C3, P) = (P5)
S(R1, r) = (R1)

Após a apresentação dos nossos modelos de fluxo simples iremos introduzir os AFDs com a possibilidade de efetuar curva após um semáforo, o desenvolvimento de novos AFDs para representar os fluxos de carros com a possibilidade de efetuar curvas ao passar por um dos semáforos é essencial para ampliar a modelagem do sistema. Isso permitirá simular situações mais complexas e próximas do comportamento real de tráfego, onde veículos frequentemente mudam de direção. Além disso, ao incorporar essas transições, tentaremos avaliar o impacto das curvas na eficiência e segurança do fluxo, garantindo que o sistema proposto possa lidar com cenários de maior variabilidade e complexidade, fundamentais para a validação de soluções em ambientes de tráfego. A formalização visual dos autômatos foi feita da seguinte forma: para nomeação dos estados foi usado um número seguido do caracter "_" para representar como da mesma forma como foi feita no código da nossa implementação, assim como as transições com um caracter dentro do parênteses como (a) que representa o alfabeto utilizado no código.

Fluxo do veículo com curva que inicia em A:

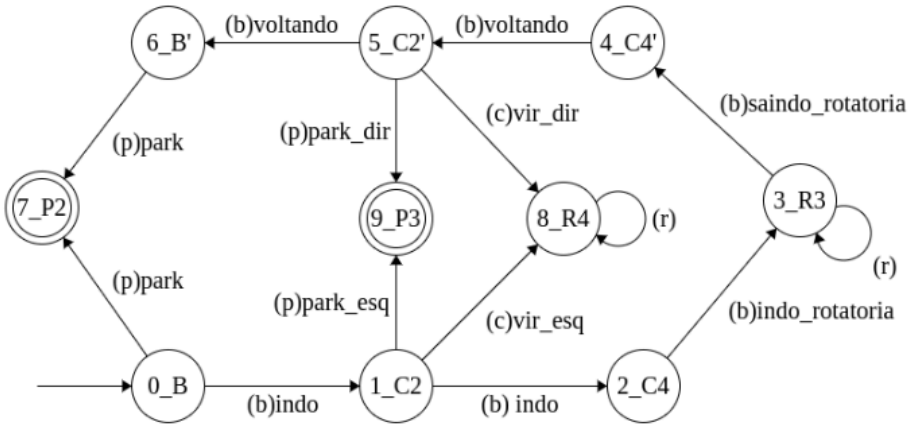


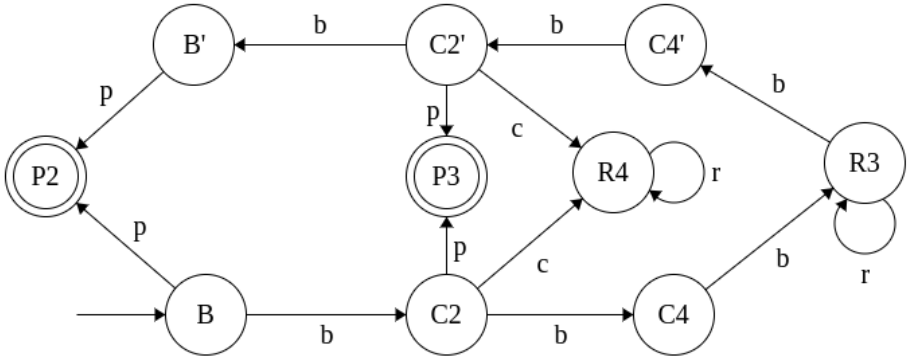


Q = ({a, d, p, r}, {0_A, 1_C1, 2_C3, 3_R2, 4_C3', 8_R1, 9_P5, 5_C1', 6_A', 7_P1}, S, {0_A}, {7_P1, 9_P5})

S(0_A, a) = (1_C1)
S(0_A, p) = (7_P1)
S(1_C1, a) = (2_C3)
S(2_C3, a) = (3_R2)
S(2_C3, d) = (8_R1)
S(2_C3, p) = (9_P5)
S(3_R2, r) = (3_R2)
S(3_R2, a) = (4_C3')
S(8_R1, r) = (8_R1)
S(4_C3', a) = (5_C1')
S(4_C3', p) = (9_P5)
S(4_C3', d) = (8_R1)
S(5_C1', a) = (6_A')
(6_A', p) = (7_P1)

Fluxo do veículo com curva que inicia em B:

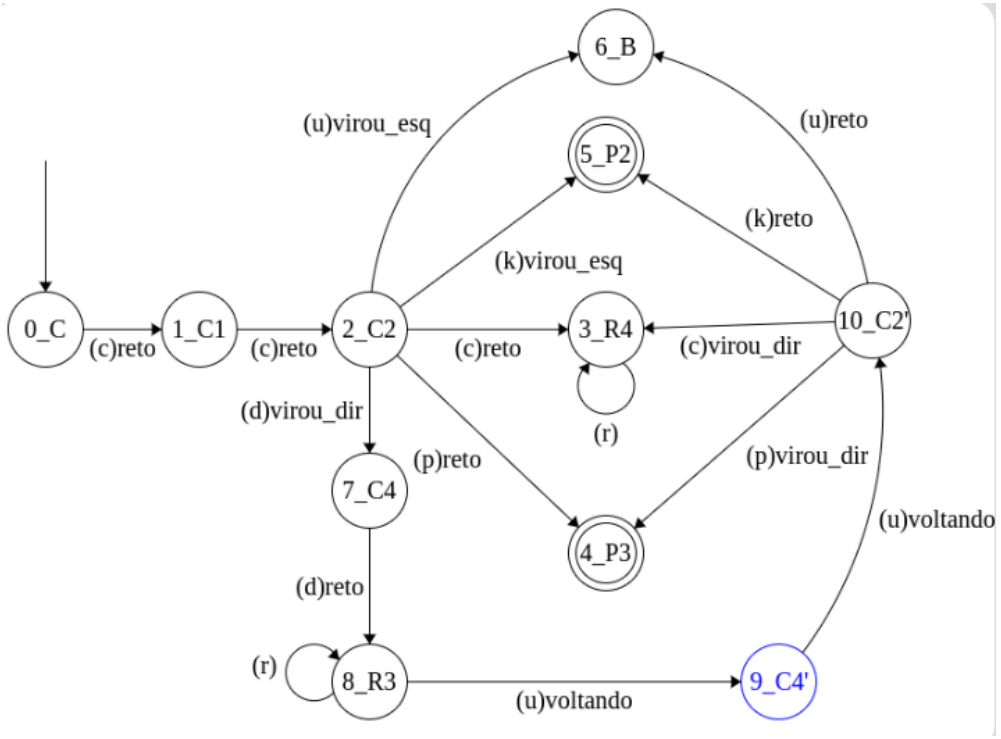


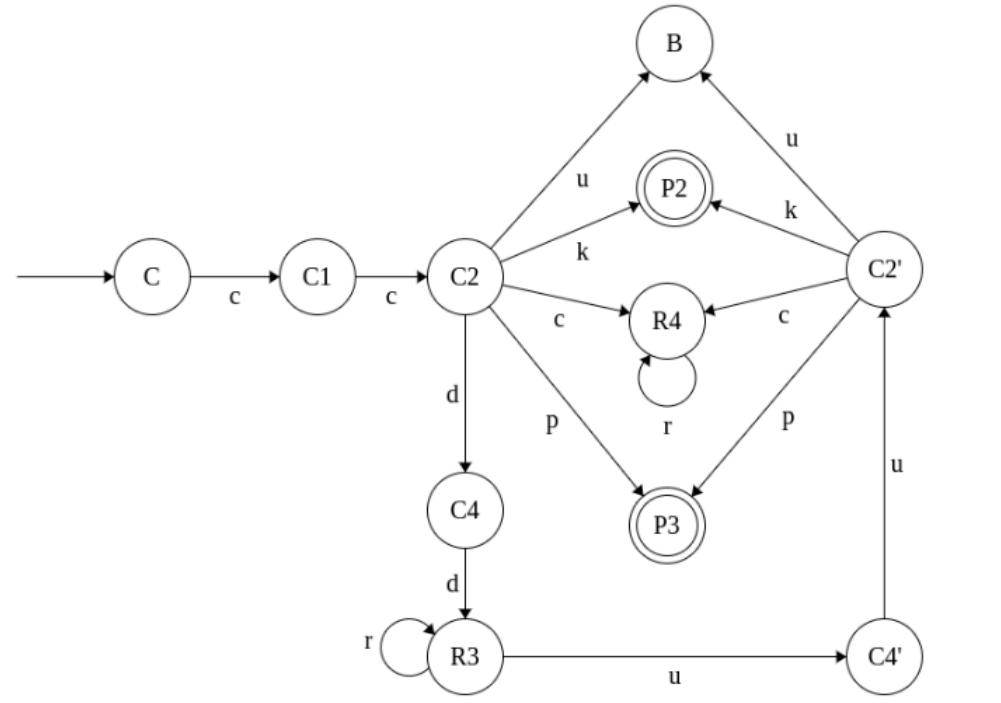


Q = ({b, c, P, r}, {B, C2, C4, R3, R4, P3, C4', C2', B', P2}, S, {B}, {P2, P3})

S(B, b) = (C2)
S(B, P) = (P2)
S(C2, b) = (C4)
S(C2, c) = (R4)
S(C2, P) = (P3)
S(C4, b) = (R3)
S(R3, r) = (R3)
S(R3, b) = (C4')
S(R4, r) = (R4)
(C4', b) = (C2')
(C2', P) = (P3)
S(C2', b) = (B')
S(B', P) = (P2)

Fluxo do veículo com curva que inicia em C:



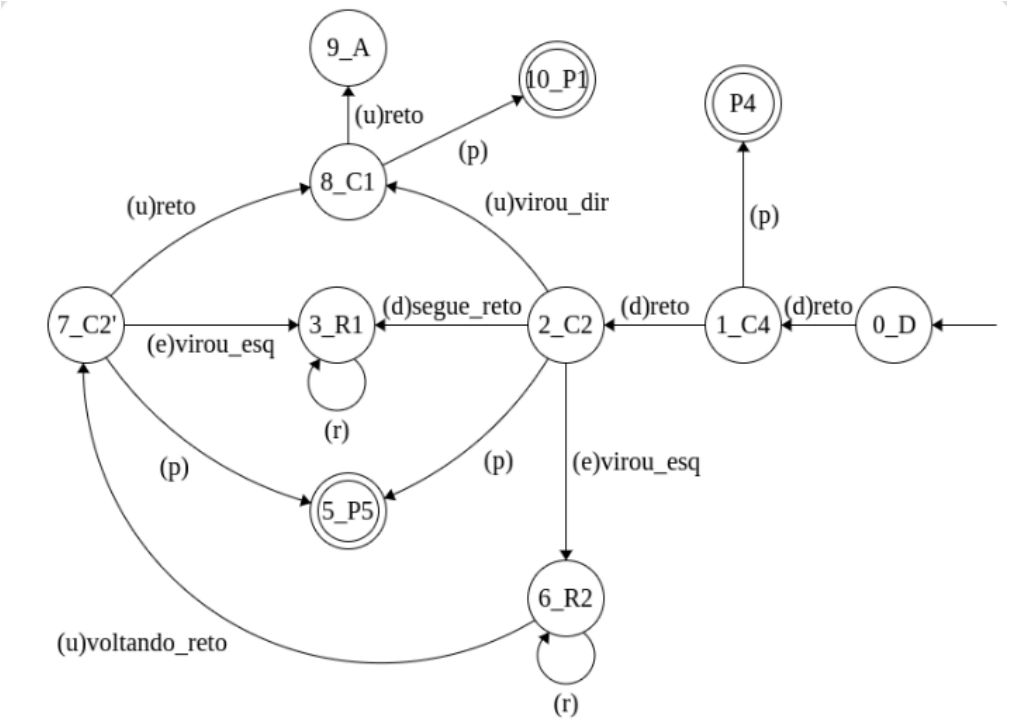


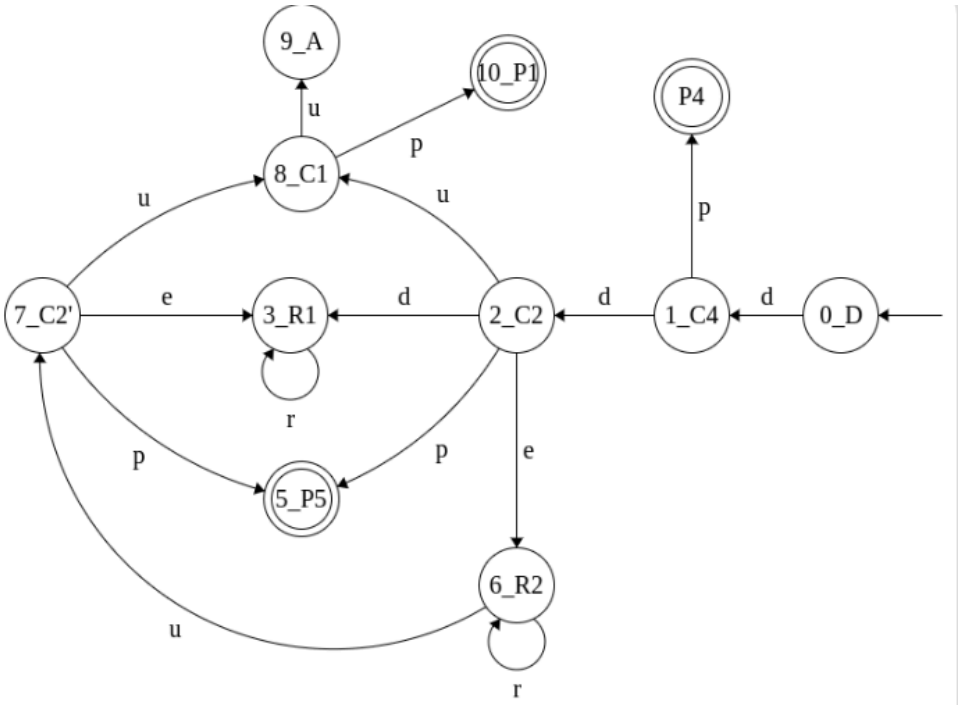
Q = ({c, u, K, P, r, d}, {C, C1, C2, B, P2, R4, P3, C2', C4', C4, R3}, S, {C}, {P2, P3})

S(C, c) = (C1)
S(C1, c) = (C2)
S(C2, c) = (R4)
S(C2, u) = (B)
S(C2, K) = (P2)

S(C2, P) = (P3)
S(C2, d) = (C4)
S(R4, r) = (R4)
S(C4, d) = (R3)
S(R3, r) = (R3)
S(R3, u) = (C4’)
S(C4’, u) = (C2’)
S(C2’, P) = (P3)
S(C2’, c) = (R4)
S(C2’, K) = (P2)
S(C2’, u) = (B)

Fluxo do veículo com curva que inicia em D:



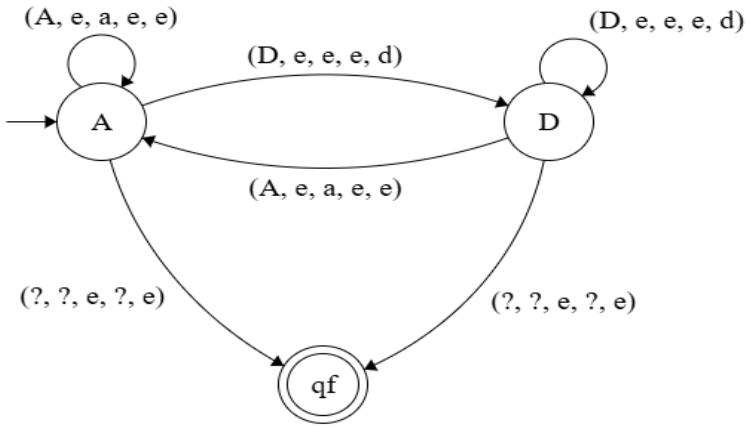


Q = ({d, P, r, u, e}, {0_D, 1_C4, P4, 2_C2, 6_R2, 5_P5, 3_R1, 7_C2', 8_C1, 10_P1, 9_A}. S, {0_D}, {P4, 5_P5, 10_P1})

S(0_D, d) = (1_C4)
S(1_C4, d) = (2_C2)
S(1_C4, P) = (P4)
S(2_C2, e) = (6_R2)
S(2_C2, P) = (5_P5)
S(2_C2, d) = (3_R1)
S(2_C2, u) = (8_C1)
S(6_R2, r) = (6_R2)
S(6_R2, u) = (7_C2')
S(7_C2', u) = (8_C1)
S(7_C2', e) = (3_R1)
S(7_C2', P) = (5_P5)
S(3_R1, r) = (3_R1)
S(8_C1, u) = (9_A)
S(8_C1, P) = (10_P1)

Para fazer o semáforos tivemos que modelar dois autômatos, para cada um, o primeiro é uma AP com duas pilhas que vai ler a palavra inicial e, e vai salvar em cada pilha os A's, B's, C's ou D's, depois vai rodar um algoritmo que vai ler as pilhas correspondentes de cada semáforo e vai montar a palavra que vai ser a entrada para o segundo autômato, ou seja, esse algoritmo vai ver qual filha é maior, e vai concatenando as letras quanto esvazia a pilha para formar a nova palavra de entrada, então os semáforos são inicializados, e são dados pelos seguintes autômatos:

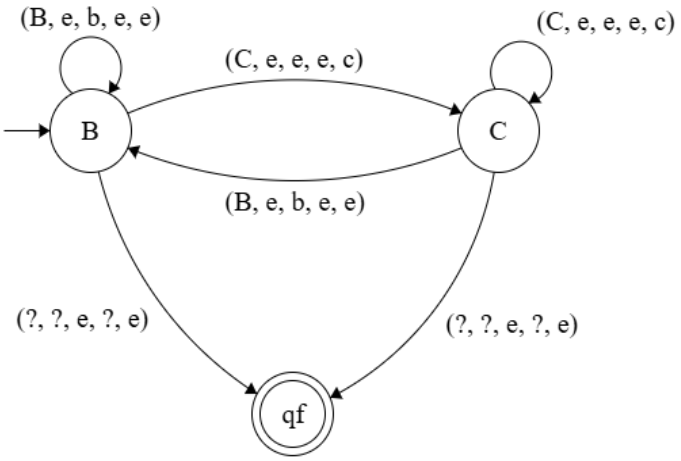
Autômato que salva as pilhas de A e D:



Q = ({A, D}, {A, D, qf}, S, {A}, {qf}, {a, d})

S(A, A, e, e) = (A, a, e)
S(A, D, e, e) = (D, e, d)
S(D, D, e, e) = (D, e, d)
S(D, A, e , e) = (A, a, e)
S(A, ?, ?, ?) = (qf, e, e)
S(D, ?, ?, ?) = (qf, e, e)

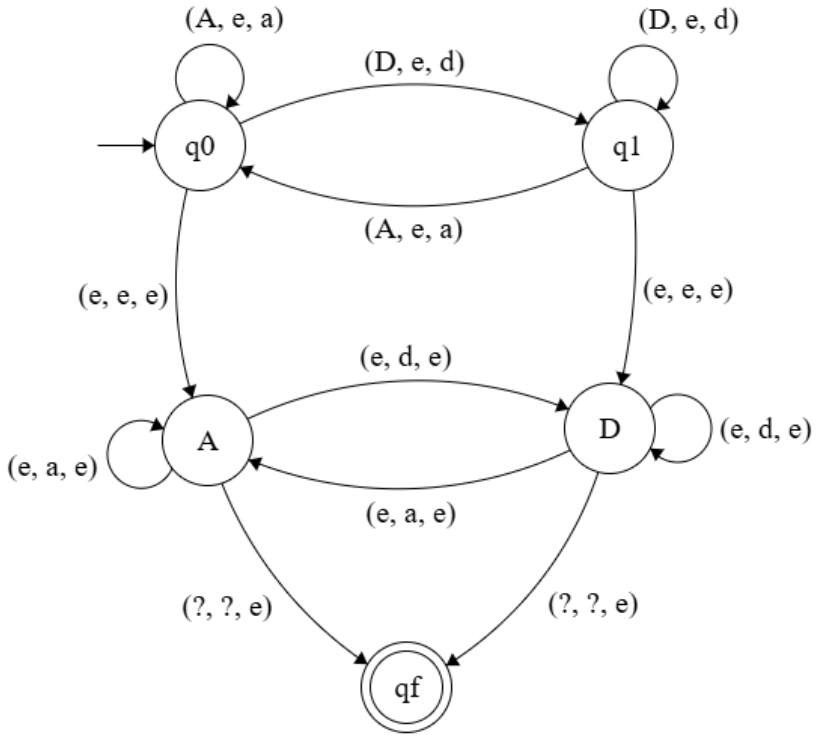
Autômato que salva as pilhas de B e C:



Q = ({B, C}, {B, C, qf}, S, {B}, {qf}, {b, c})

S(B, B, e, e) = (B, b, e)
S(B, C, e, e) = (C, e, c)
S(C, C, e, e) = (C, e, c)
S(C, B, e , e) = (B, b, e)
S(B, ?, ?, ?) = (qf, e, e)
S(C, ?, ?, ?) = (qf, e, e)

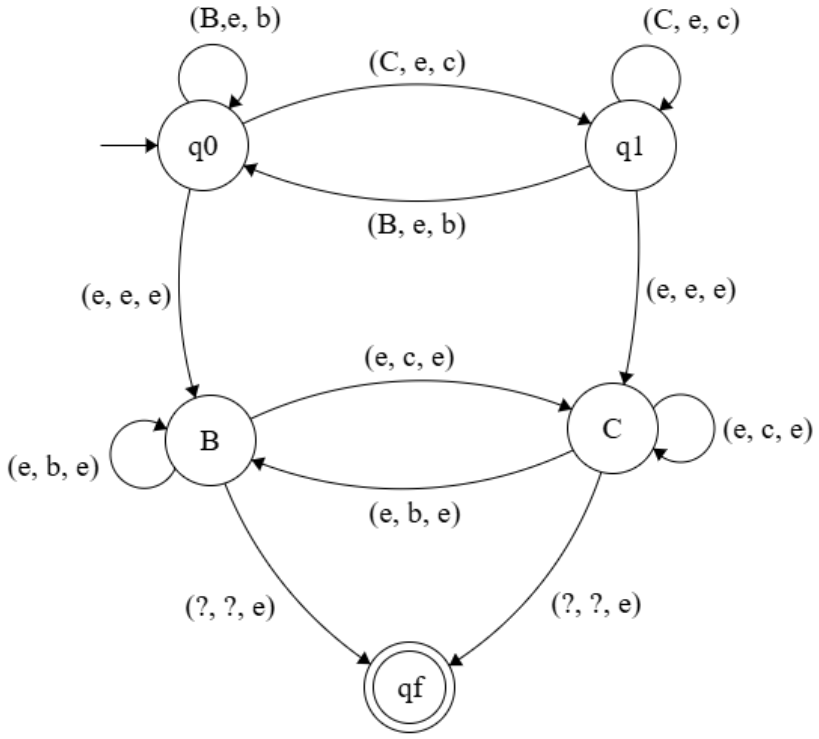
Sinaleiro Cruzamento C3:
Estados q0 e q1 são responsáveis por ler a palavra e empilhar, os estados A e D, são o semáforo em si, logo se está no estágio A, e semáforo está aberto para carros que vem de C1 e R2, e fechados para os que vem de C4, já quando está em D o sinaleiro está aberto para os carros em C4 e fechado para os em C1 e R2, e no estágio qf não tem mais carros tentando passar por aquele semáforo.



Q = ({A, D}, {q0, q1, A, D, qf}, S, {q0}, {qf}, {a, d})

S(q0, A, e) = (q0, a)
S(q0, D, e) = (q1, d)
S(q1, D, e) = (q1, d)
S(q1, A, e) = (q0, a)
S(q0, e, e) = (A, e)
S(q1, e, e) = (D, e)
S(A, e, a) = (A, e)
S(A, e, d) = (D, e)
S(D, e, d) = (D, e)
S(D, e, a) = (A, e)
S(A, ?, ?) = (qf, e)
S(D, ?, ?) = (qf, e)

Sinaleiro Cruzamento C2:
Estados q0 e q1 são responsáveis por ler a palavra e empilhar, os estados B e C, são o semáforo em si, logo se está no estágio B, e semáforo está aberto para carros que vem de B e C4, e fechados para os que vem de C1, já quando está em C o sinaleiro está aberto para os carros em C1 e fechado para os em B e C4, e no estágio qf não tem mais carros tentando passar por aquele semáforo.



Q= ({B, C}, {q0, q1, B, C, qf}, S, {q0}, {qf}, {b, c})

S(q0, B, e) = (q0, b)
S(q0, C, e) = (q1, c)
S(q1, C, e) = (q1, c)
S(q1, B, e) = (q0, b)
S(q0, e, e) = (B, e)
S(q1, e, e) = (C, e)
S(B, e, b) = (B, e)
S(B, e, c) = (C, e)
S(C, e, c) = (C, e)
S(C, e, b) = (B, e)
S(B, ?, ?) = (qf, e)
S(C, ?, ?) = (qf, e)

Implementação

A nossa implementação é um conjunto de 3 arquivos:

main.c -> contém a montagem dos autômatos, o menu e a chamada para funções;
lib.h -> assinatura das funções assim como as structs usadas;

lib.c -> contém a implementação das funções para reconhecimento das palavras, simulação dos autómatos individualmente e o fluxo conjunto dos autómatos.

Os modelos dos autômatos para o fluxo foram implementados utilizando o código que já haviamos implementado no TC complementar de AFDs, assim foi necessário remodelar as transições e estados para que fosse possível encaixar no nosso modelo como pode ser visto no exemplo abaixo:

```
// MODELO SEM CURVAS
Automaton afd_a;

// definindo os estados
afd_a.num_estados = 8;
afd_a.estado_inicial = 0;
afd_a.estados_aceitos[0] = 7; // estado final
afd_a.estados_de_aceitacao = 1; // quantidade de estados finais

// deefinindo o alfabeto
afd_a.alfabeto[0] = 'a';
afd_a.alfabeto[1] = 'r';
afd_a.alfabeto[2] = 'p';

// definindo as transições
afd_a.num_transicoes = 9;
afd_a.transitions[0] = (Transicao){0, 'a', 1};
afd_a.transitions[1] = (Transicao){0, 'p', 7};
afd_a.transitions[2] = (Transicao){1, 'a', 2};
afd_a.transitions[3] = (Transicao){2, 'a', 3};
afd_a.transitions[4] = (Transicao){3, 'r', 3};
afd_a.transitions[5] = (Transicao){3, 'a', 4};
afd_a.transitions[6] = (Transicao){4, 'a', 5};
afd_a.transitions[7] = (Transicao){5, 'a', 6};
afd_a.transitions[8] = (Transicao){6, 'p', 7};
```

Esse formato de criação de AFD se repete 8 vezes no nosso código **main.c** pois cada um representa as possibilidades de fluxo para cada carro a partir de um ponto inicial (4 para fluxo sem curva e 4 para fluxo com curva). Caso seja de interesse do usuário, ele pode testar nossos modelos que formalizamos acima através do nosso menu, a simulação de cada modelo é feita pela função **simulate_afd(),** o menu de iteração pode ser visto a seguir:

```
=====================================================================

---- TRABALHO FINAL DA MATÉRIA LINGUAGENS FORMAIS E AUTÔMATOS ----

Professor: Ricardo Ferreira Martins.
Alunos: Gustavo de Souza; José Augusto Laube.

=====================================================================



================ MENU DE SELEÇÃO ================
(1) Simular AFDs de fluxo sem curva no semaforo;
(2) Simular AFDs de fluxo com curva no semaforo;
(3) Simular fluxo com semaforos (SEM TROCA DE DIREÇÃO);
(4) Simular fluxo com semaforos (COM TROCA DE DIREÇÃO);
(0) Para finalizar o programa;
```

Partindo para a simulação dos fluxos, elas foram feitas com o auxílio de duas funções para cada semáforo, foi necessário dividir em duas funções para cada fluxo (um com curva e outro sem curva) pois o carro que inicia em A só cruzará a via com um carro que inicia no fluxo D, e o mesmo se aplica para os carros que iniciam B e C, assim o fluxo é simulado

somente com os carros que possuem a possibilidade de se encontrarem, abaixo é possível ver a chamada das funções no main:

```
// Simulação do fluxo
// fluxo divido em rezão dos modelos e não haver conflito de vias
criarFluxoComSemaforo_BC(&afd_b, &afd_c, palavra_b, palavra_c,
palavra_bc, &c2);

criarFluxoComSemaforo_AD(&afd_a, &afd_d, palavra_a, palavra_d,
palavra_ad, &c3);
```

Essas funções basicamente simulam os autômatos a partir de caminhos pré-definidos que representam todas as possibilidades de rota de um carro a partir de seu ponto inicial, essa geração aleatória é feita pela função **definir_caminhos_eficientes(),** na simulação, os carros efetuam transações sincronizados pois não é possível aplicar a noção de tempo na implementação, então todos avançam uma transição por ciclo, porém, como estamos constantemente verificando a quantidade de carros na via e essa sendo a lei principal do nosso semáforo, o autômato de uma via vai avançar até encontrar o semáforo e o mesmo estiver com a cor verde (a via com mais carros tem a preferência do verde). O nosso código registra passo a passo a progressão dos autômatos que consequentemente ilustra o avanço dos carros a partir de um ponto inicial até o seu estacionamento assim como os estados do semáforo, abaixo é possível ver uma ilustração do fluxo sem curvas:

```
Caminhos definidos:
Palavra A: aaaraaap
Palavra B: p
Palavra C: ccp
Palavra D: dp


Insira a palavra que deseja testar:
da


--------> INÍCIO DA SIMULAÇÃO AD <--------
Simulação para o(s) carro(s): da

--- Ciclo 1 ---
Semáforo: Fluxo A está vermelho, Fluxo D está verde
Carro 1: estado atual = 0, simbolo = 'd'
Carro 1: novo estado = 1
Carro 2: estado atual = 0, simbolo = 'a'
Carro 2: novo estado = 1
Semaforo alterado: Fluxo A agora esta verde e Fluxo D está vermelho.
--- Fim do Ciclo 1 ---

--- Ciclo 2 ---
Semáforo: Fluxo A está verde, Fluxo D está vermelho
Carro 1: estado atual = 1, simbolo = 'p'
Carro 1: novo estado = 4
Carro 1 concluiu o percurso.
Carro 2: estado atual = 1, simbolo = 'a'
Carro 2: novo estado = 2
--- Fim do Ciclo 2 ---
```

Já os modelos que representam o fluxo com curvas foram mais desafiador pois os modelos são maiores e como a operação do semáforo é totalmente dependente da quantidade de

carros na via, garantir que um veículo seja contabilizado de forma correta via código foi um tanto desafiador, mas depois de inúmeros testes e modificações nos nossos modelos alcançamos um resultado ideal que pode ser visto nas seguintes funções:

```
// Simulação do fluxo
// fluxo divido em rezão dos modelos e não haver conflito de vias
criarFluxoComCurvas_BC(&afd_b_curva, &afd_c_curva, palavra_b_curva,
palavra_c_curva, palavra_bc_curva, &c2_curva);

criarFluxoComCurvas_AD(&afd_a_curva, &afd_d_curva, palavra_a_curva,
palavra_d_curva, palavra_ad_curva, &c3_curva);
```

O raciocínio que utilizamos para uma implementação perfeita foi uma constante monitoração de estados que são cruciais para determinar os possíveis movimentos de um carro ao chegar no semáforo, assim conseguimos fazer os sensores registrarem de forma correta um novo carro na via caso ele efetue uma curva e alterar o estado do semáforo se necessário. Abaixo há a lista de quais estados precisamos mapear com atenção no código pois eles representam possíveis curvas ao incidir no semáforo:

Fluxo B (BC): Estados relevantes: **1, 5**.
Fluxo C (BC): Estados relevantes: **2, 7, 10**.
Fluxo A (AD): Estados relevantes: **1, 5**.
Fluxo D (AD): Estados relevantes: **2, 6, 7, 10**.

Por fim vale ressaltar que para compilação do código em ambiente Linux pode ser feita com o seguinte comando:

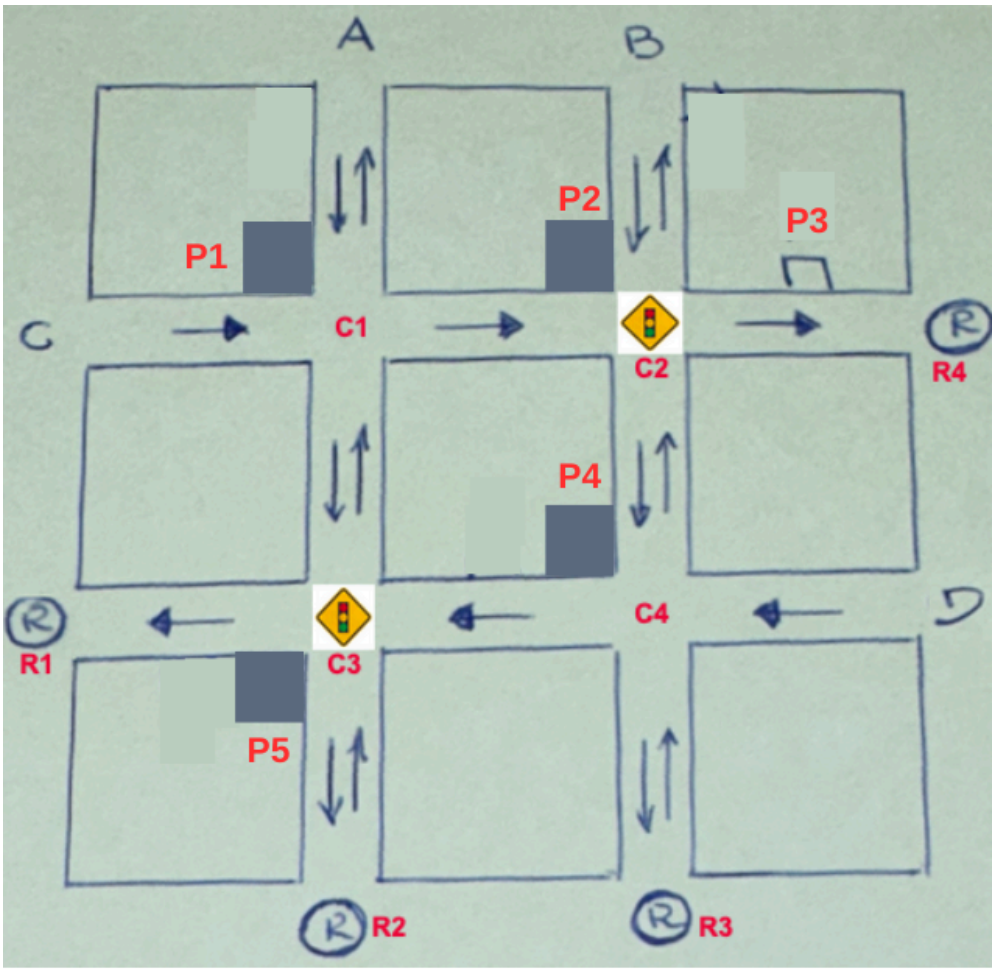
gcc -w main.c lib.c && ./a.out

E o código está disponível no GitHub juntamente com o relatório em:
https://github.com/Souza04Gustavo/LFA

Trabalhos Futuros

O trabalho foi concluído com sucesso, porém foi iniciado uma implementação gráfica do projeto que representasse a trajetória dos carros nas vias no terminal, porém a complexidade e a alta demanda de materiais adjacentes não nos permitiram investir tempo nisso visto que o foco do trabalho é a formalização dos modelos e a implementação para sua verificação, mas uma prévia da implementação pode ser encontrada no GitHub com o nome de **teste_grafico.c**. Além disso, através das nossas implementações conseguimos observar que o posicionamento dos estacionamentos poderiam facilitar a progressão do fluxo, visto que vias como a via B e a via A no fluxo sem troca de via, só possuem uma unidade de estacionamento, e caso o carro queira seguir em frente e não estacionar de primeira, deve efetuar todo o trajeto até retornar de fato ao ponto em que o mesmo se encontra e caso tenha mais carros nessas vias do que em suas adjacentes, o tempo de

espera é maior. Outro ponto que notamos que poderia melhorar o fluxo é o posicionamento de estacionamentos em esquinas dos quarteirões, para que assim duas vias possam aproveitar do mesmo estacionamento e isso traria a possibilidade e trajetos menores, diminuindo o número de ciclos e possivelmente o tempo de espera de carros em semáforos vermelhos. Abaixo há uma representação de possíveis melhores posicionamentos dos estacionamentos que melhoram o fluxo de veículos:



Conclusão

Foi um trabalho árduo desenvolver uma implementação que representasse corretamente os nossos modelos e permitissem um fluxo com trocas de via mas foi possível a partir de muito esforço e auxílio do professor com a formalização dos modelos antes da codificação, essa sequência de passos foi com toda certeza essencial para a confecção do trabalho e nos guiou para o que acreditamos que seja resultados esperados. Assim conseguimos modelar através de autômatos o fluxo de veículos na vias fornecidas pelo professor, bem como o sinaleiro e seu funcionamento, baseado no fluxo de cada via que é analisado por um sensor, que também foi modelado por um autômato. Portanto esse trabalho trouxe uma visão prática de como autômatos podem ser usados na nossa vida profissional para resolver problemas como esse e vários outros, sendo uma ferramenta computacional muito poderosa.