# Fábrica de automóveis simulada por meio de autômatos, expressões regulares e outros formalismos.

**Kelvin W. Bruggmann**[1], **Rafael G. de Mello**[1], **Vinicius Gasparini**[1]

[1]Universidade Estadual do Estado de Santa Catarina (UDESC)
Caixa Postal 631 – 89.219-710 – Joinville – SC – Brazil

{kelvinwelter.b,rafaelgranzademello}@gmail.com, v.gasparini@edu.udesc.br

***Abstract.*** *Looking for simulate and represent a production line management environment in a car factory, we developed a solution to this problem from the use of deterministic finite automata, regular expressions and regular grammar. Represent a demand, modeling the production line and errors handling using automata brings greater power and autonomy to the management environment.*

***Resumo.*** *Com o objetivo de simular e representar um ambiente de gerenciamento de linhas de produção em uma fábrica de automóveis, desenvolvemos uma solução para esse problema a partir da utilização de autômatos finitos determinísticos, expressões regulares e gramaticas regulares. Representar a demanda, modelar a linha de produção e tratar erros utilizando autômatos traz um grande poder e maior autonomia para o ambiente de gerenciamento.*

## 1. Introdução

Sob o contexto de utilizar autômatos e linguagens formais em uma aplicação condizente com a realidade e utilizar de seus potenciais para maior facilidade e autonomia no projeto, é necessário que haja entendimento sobre do que se trata tais conceitos.
Esses conceitos devem então serem aplicados em nosso problema, de acordo com as especificações e objetivos em mente. Visto que nossa fábrica deve ser capaz de lidar com demandas errôneas, falhas na produção. Sabendo lidar com autonomia em relação a defeitos graves de fabricação ou defeitos comuns e reconhecendo pedidos inválidos.
Tendo os conceitos e o problema já em mente, podemos partir para a modelagem de autômatos necessários para a realização do projeto onde trataremos os diversos problemas a serem resolvidos a partir da construção de autômatos para lidarem com esses casos. Com autômatos modelados corretamente e tendo consciência de como deve ser tratados os casos de erro, verificação da demanda, fabricação, etc.. Pode-se então iniciar a abordagem mais prática do projeto, expondo códigos, raciocínios utilizados e mostrando a construção do algoritmo de forma concisa.
Após abordarmos todas essas questões, podemos demonstrar resultados e expor o que foi obtido. Trataremos de expor as devidas conclusões sobre o trabalho, abordar utilidades encontradas para esse trabalho. Todos esses tópicos abordados tem como objetivo a maior compreensão do conteúdo a relatar. Iniciando com um contexto geral e partindo até a abordagem em relação ao código.

## 2. Contexto

Com o intuito de criar um ambiente que consiga ter autonomia para lidar com erros, consiga dar liberdade para o usuário realizar determinada encomenda ou mudar aspectos no

ambiente de gerenciamento, foi necessário o uso de autômatos e formalismos ao projeto. Inicialmente tais conceitos podem parecer complexos e com pouco uso, porém isso facilmente se prova o contrário. Esse projeto serve de exemplo para tal prova, mostrando o quão úteis e abrangentes são esses conceitos. Para o devido entendimento do trabalho e seu funcionamento, é necessário saber pelo menos o básico sobre alguns formalismos e modelos que são relevantes e serviram como base teórica e prática do trabalho. Iniciaremos então explicando no que consiste um autômato.

### 2.1. Autômatos

De forma geral, um autômato consiste em uma modelo matemático para uma máquina de estados finitos, isto é, um autômato funciona como o reconhecedor de uma determinada linguagem, muito usado na área de compiladores, editores de texto, etc.. Citado como uma máquina de estados finitos, é importante então, ter conhecimento sobre o que são estados. Os estados em um autômato nada mais é do que uma forma de comportamento do sistema, ou seja, um estado representa como o sistema está se comportando em determinado momento, esse comportamento deve ser relevante ao sistema, caso o contrário não existe sentido em tratá-lo como um estado. Tendo isso em mente, é importante também saber o que um autômato reconhece, sabemos que autômatos reconhecem linguagens mas que linguagens são essas? Isso deve ser esclarecido antes de continuarmos o nosso projeto.

### 2.2. Linguagens

Para descrevermos uma linguagem de forma breve e informal, não é preciso muito esforço, basta definirmos uma linguagem como uma forma de comunicação. Porém, precisamos ir além disso, nosso autômato trabalhará com linguagens formais, dessa forma, é necessário termos ciência do que se trata tal termo. Uma linguagem formal trata-se de uma linguagem sobre um conjunto de caracteres, que chamaremos de alfabeto, desse alfabeto, a linguagem formal é o que definirá as regras e condições para criarmos um conjunto de palavras sintaticamente corretas sobre esse alfabeto. Ou seja, a linguagem formal nos dá as regras para formarmos uma palavra sobre determinado alfabeto.
Com essas regras em mãos podemos gerar também uma expressão regular que representa as possíveis palavras que podemos criar com as nossas regras e alfabeto definidos. Muito útil para percorrer palavras e determinar se elas são parte de uma linguagem através de suas especificações escritas na forma de uma expressão regular. Agora com uma base já estabelecida e formalismos já apresentados podemos seguir em frente, antes de seguirmos para a parte prática é importante descrevermos o que estamos visando, quais nossos objetivos e como queremos alcançar determinados alvos.

## 3. Descrição do Problema

De forma geral, é simples dizermos o que queremos construir: um ambiente de gerenciamento de linhas de produção em uma fábrica de automóveis. Porém, somente citar isso não basta, já que mais para frente dificultaria o entendimento da parte prática. Então, faz-se necessário abordar a descrição do problema de uma forma diferente, abordaremos o problema de forma dividida, de parte em parte para que com o maior entendimento de problemas especificados, o entendimento geral também se fará presente.
O que queremos então? Imagine o seguinte cenário: uma determinada marca de carros

decide que o melhor modelo de negócios seria fabricar carros através de demanda. Ou seja, a fábrica não iria gastar recursos e mão de obra para fabricar carros sem a certeza de que aqueles carros eram realmente necessários naquele momento, assim, com esse novo cenário seria necessário realizar encomendas para a fábrica. Assim, funcionários de concessionárias da marca deveriam realizar encomendas de acordo com a demanda de cada loja. Isso criaria um estoque muito mais direcionado para o cenário das diferentes distribuidoras espalhadas pelo país, já que cada uma teria maior estoque do que se tem maior demanda. A partir disso então, queremos que um funcionário em sua respectiva concessionária possa realizar uma encomenda de carros, porém, devemos estar cientes de que o funcionário pode cometer erros durante o processo. Essa situação deve ser tratada de maneira a impedir que demandas errôneas sejam fabricadas.
Também é importante tratarmos de outros aspectos em relação a fábrica. Por exemplo, é importante lembrar que cada carro poderá ter suas cores diferenciadas, um carro de modelo X não necessariamente deve ter as mesmas cores de um carro de modelo Y. Algo trivial e fortemente estabelecido na indústria automotiva. Faz-se necessário também, a diferenciação de carros de mesmo modelo por outros aspectos, como por exemplo: poderemos ter um carro de modelo X *hatch* ou *sedan*, ou então um modelo Y com motores 1.4 e 1.6. Dessa forma, precisamos tratar e resolver o problema de forma que possa haver diversificação e autonomia de cores e modelos entre os carros da marca.
Em suma, precisamos simular a fábrica de forma a permitir liberdade na produção dos carros e a não aceitar erros na demanda, como por exemplo um pedido de carro com a cor não especificada, ora, a fábrica não pode adivinhar a cor e então não deve aceitar o pedido. Ademais, precisamos simular as linhas de produção da fábrica, não basta somente tratar o que já falamos sobre porém não fabricar o carro em si. Para simularmos uma linha de produção buscaremos nos ater a realidade porém não de forma exagerada, utilizaremos estados relevantes e de importância numa fabricação, não reduzindo a fabricação nos mínimos detalhes já que esse não é a proposta em tal ambiente.
De modo geral, esse é o problema que trabalharemos em cima e o qual será modelado computacionalmente o nosso ambiente de gerenciamento de linhas de produção em nossa fábrica, trataremos agora de como iremos modelar e trabalhar em cima de tal criação.

## 4. Modelagem computacional

Tendo nosso problema já descrito e especificado, podemos iniciar a modelagem e especificar uma solução para nosso problema, Ou seja, vamos agora demonstrar e pensar como resolver o problema da melhor forma. Quando pensamos nesse problema de forma geral, podemos perceber facilmente que ele pode ser abordado de diferentes maneiras e com detalhes distintos, a abordagem que iremos adotar então é a seguinte: dividiremos a demanda e a produção em dois autômatos distintos, todavia, ligados entre si.
Além disso, iremos modelar uma forma de tratarmos os erros na parte da demanda e também erros decorrentes da produção. O que acontecerá então é o seguinte, iremos realizar a modelagem de um autômato para a produção e outro para demanda, com isso em mãos poderemos definir outros pontos importantes do trabalho e a abordagem de tais. Seguiremos então para a modelagem do autômato da demanda.

### 4.1. Modelagem relacionada a demanda

Para iniciarmos, faz-se necessário termos em mente o que queremos que aconteça no nosso processo de aceitação (ou recusa) da demanda. Imagine então a seguinte situação

que queremos atingir, nossa fábrica terá 8 opções de carros, cada carro terá 4 opções de cores e também 2 opções possíveis de modelos. Ou seja, haverá 8 possibilidades distintas de combinação em cada carro. Partindo desse pressuposto, apesar da grande variedade de opções, ainda é possível que o usuário envie uma demanda errônea.
Imagine então o seguinte cenário, o usuário nos envia uma demanda requisitando um carro de modelo Passat, porém nossa fábrica não fabrica Passats, como proceder diante de tal fato? Ou então, é requisitado um carro em uma cor em que este não é fabricado. Precisamos tratar isso e tomaremos a seguinte abordagem, se uma etapa do pedido estiver errada, ignoraremos todo o pedido de tal carro e a partir disso, nossa máquina não irá passar para outros estados, já que não há nenhuma transição que se adequará a demanda errônea requisitada, em outras palavras, nosso autômato de verificação "morre". Assim, o autômato não chegará ao final, onde seria enviado o pedido para a linha de produção, e dessa forma não haverá pedidos errados na linha de produção.
Após o erro ser impedido de seguir para a produção, o autômato continuaria fazendo exatamente essa função com os outros carros requisitados, desde que não houvesse sido atingido o limite de erros na linha de produção da fábrica. Que limite seria esse? Esse limite seria um total de 7 erros em uma linha de produção, caso atingíssemos um número maior que esse, nossa linha de produção teria seus processos abortados. Assuma que isso se deve devido ao fato de que em uma fábrica real, tamanho número de erros trata-se de algo muito além do permitido e tolerado, então faz-se necessário uma inspeção na linha de produção para determinar o que está acontecendo. Essa inspeção durará um dia de trabalho e por isso nossa fábrica deve abortar e cancelar qualquer processo destinado a essa linha. Com nosso problema definido, pode-se então modelar como será nosso autômato em relação a demanda, disponível Figura 1, que obedece a seguinte 5-tupla.

$M_{ver} = (\Sigma, Q, \delta, q0, q0)$

onde

$\Sigma = \{b, w, j, g, u, m, z, n, L, A, Z, R, B, P, V, @, \&, !, \#, \$, ; \}$

$Q = \{q0, q1t, q1c, q1m, q2t, q2c, q2m, q3t, q3c, q3m, q4t, q4c, q4m, q5t, q5c, q5m, q6t, q6c, q6m, q7t, q7c, q7m, q8t, q8c, q8m\}$

$\delta$ : Tabela 1

Da imagem percebe-se alguns aspectos do autômato, o modelo tem seu estado inicial e final como os mesmos, ou seja, ele começa lendo um pedido e termina lendo um novo pedido. A partir disso, também é possível perceber uma abordagem que tomamos em relação a separação dos pedidos, como podemos saber que um pedido foi finalizado e iniciar a verificação de um novo requisito de carro? Para resolução desse problema, definimos que nossos pedidos devem ser enviados com o uso de um separador, isso traz maior facilidade na modelagem do autômato e também na construção do código, perceba na figura acima, utilizamos o caractere ';' como separador entre as encomendas.
Perceba também outros aspectos como a independência e autonomia que cada tipo de carro tem em relação aos seus modelos e cores, tudo isso visando um ambiente de gerenciamento mais fidedigno a realidade. Agora com um modelo de demanda já definido e planejado para implementação, nossos carros já estão sendo aceitos ou rejeitados, precisamos nos preocuparmos agora com a fabricação dos carros que foram aceitos ou seja, trabalhar e modelar a nossa fábrica antes de nos aprofundarmos no código em si.

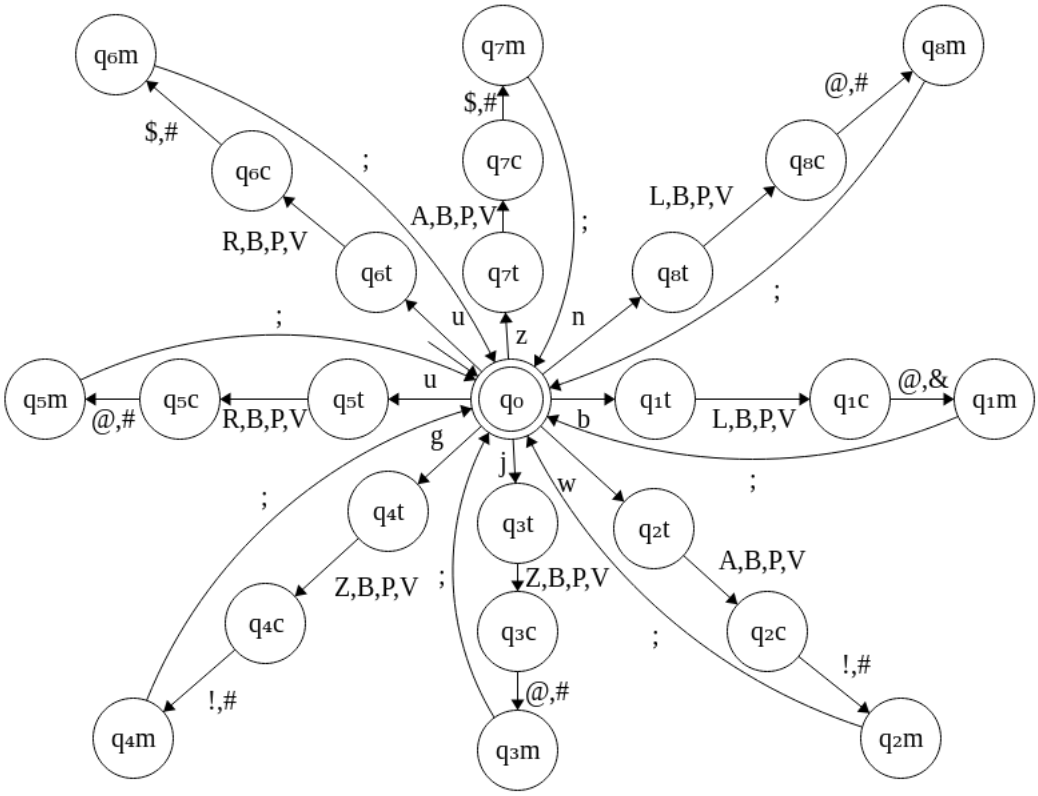


**Figura 1. Autômato de verificação de demanda**

| δ | b | w | j | g | u | m | z | n | L | A | Z | R | B | P | V | @ | & | ! | # | $ | ; |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| q0 | q1t | q2t | q3t | q4t | q5t | q6t | q7t | q8t | | | | | | | | | | | | | |
| q1t | | | | | | | | | q1c | | | | q1c | q1c | q1c | | | | | | |
| q1c | | | | | | | | | | | | | | | | q1m | q1m | | | | |
| q1m | | | | | | | | | | | | | | | | | | | | | q0 |
| q2t | | | | | | | | | | q2c | | | q2c | q2c | q2c | | | | | | |
| q2c | | | | | | | | | | | | | | | | | | q2m | q2m | | |
| q2m | | | | | | | | | | | | | | | | | | | | | q0 |
| q3t | | | | | | | | | | | q3c | | q3c | q3c | q3c | | | | | | |
| q3c | | | | | | | | | | | | | | | | q3m | | | q3m | | |
| q3m | | | | | | | | | | | | | | | | | | | | | q0 |
| q4t | | | | | | | | | | | q4c | | q4c | q4c | q4c | | | | | | |
| q4c | | | | | | | | | | | | | | | | | | q4m | q4m | | |
| q4m | | | | | | | | | | | | | | | | | | | | | q0 |
| q5t | | | | | | | | | | | | q5c | q5c | q5c | q5c | | | | | | |
| q5c | | | | | | | | | | | | | | | | q5m | | | q5m | | |
| q5m | | | | | | | | | | | | | | | | | | | | | q0 |
| q6t | | | | | | | | | | | | q6c | q6c | q6c | q6c | | | | | | |
| q6c | | | | | | | | | | | | | | | | | | | q6m | q6m | |
| q6m | | | | | | | | | | | | | | | | | | | | | q0 |
| q7t | | | | | | | | | | q7c | | | q7c | q7c | q7c | | | | | | |
| q7c | | | | | | | | | | | | | | | | | | | q7m | q7m | |
| q7m | | | | | | | | | | | | | | | | | | | | | q0 |
| q8t | | | | | | | | | q8c | | | | q8c | q8c | q8c | | | | | | |
| q8c | | | | | | | | | | | | | | | | q8m | | | q8m | | |
| q8m | | | | | | | | | | | | | | | | | | | | | q0 |

**Tabela 1. Tabela da Função Delta da Autentificação**

Perceba que nosso alfabeto utiliza-se de letras e símbolos para a leitura da demanda. Porém, nossos carros tem seus nomes, modelos e cores especificadas. Utilize as tabelas 2(a), 2(b) e 2(c) para legendas.

(a) Tabela de cores

| Cores | |
|---|---|
| Nome | Símbolo |
| Azul | A |
| Branco | B |
| Bronze | Z |
| Limão | L |
| Prata | R |
| Preto | P |
| Vermelho | V |

(b) Tabela de modelos

| Modelos | |
|---|---|
| Nome | Símbolo |
| Hatch | @ |
| Lounge | $ |
| Mini | & |
| Sedan | # |
| SUV | ! |

(c) Tabela de tipos

| Tipos | |
|---|---|
| Nome | Símbolo |
| b | Bozzola |
| w | Weiss |
| j | Jonk |
| g | Gasparini |
| u | Utiama |
| m | March |
| z | Granza |
| n | Brugmann |

**Tabela 2. Tabelas de legendas**

### 4.2. Modelagem da fábrica

No momento em que pensamos em uma fábrica, imediatamente pensamos em alguns aspectos: paralelismo, diversas etapas de montagem, possibilidade de erros e um grande fluxo de montagem. Para alçarmos uma simulação o mais próximo possível da realidade de uma fábrica de automóveis é necessário alcançarmos esses aspectos. Dessa forma, pensaremos em um modelo que reproduza com sucesso tais requisitos.

Podemos iniciar então pensando, como ocorre a fabricação de um carro? Quais etapas são necessárias para que um carro seja fabricado? Essas perguntas podem ter suas respostas amplamente variadas dependendo da profundidade de detalhes a ser descritos. Porém, em nosso ambiente de gerenciamento não iremos nos ater aos detalhes mais minuciosos, será suficiente representarmos como estados em nosso autômato somente as etapas principais e que representam mudanças relevantes na construção do nosso veículo, essas etapas foram baseadas em fábricas reais como demonstrado em pesquisa realizada pela quatro rodas (2011). Dessa forma, iremos criar nossa linha de produção com 7 etapas, sendo estas: estamparia, estruturação, funilaria, motor,montagem,pintura e finalização. Tendo essas etapas definidas já é possível imaginar nossa linha de produção, porém, devemos nos recordar que estamos lidando com uma fábrica simulando a realidade e assim passível de erros.

Tendo em vista que será possível ocorrer erros em nossas fábricas, devemos lidar com esses erros. Para isso, devemos primeiro dividir nossos erros em dois grupos, erros graves e erros comuns. Erros graves de fabricação tratam-se de situações raras porém possíveis onde o defeito é algo que impossibilita o retrabalho do carro e assim, o carro deve ser descartado. Esses erros graves tem possibilidade de ocorrência durante o processo de estamparia. Caso houvesse erro nesse processo então, não haveria como reparar o veículo de maneira adequada, impossibilitando a sua venda e decorrente disso sendo descartado. Os processos de fabricação que não são passíveis de erros graves ainda teriam erros a lidar, erros comuns, esse tipo de ocorrência trata-se de casos onde é possível o reparo, então o veículo vai para a área de reparo e volta para a linha de produção com os devidos consertos realizados e podendo seguir normalmente na fábrica(erros comuns sempre

serão consertados), a contagem de erros será gravada numa fita de saída comum à todas as linhas de produção .
Com tudo isso estabelecido então, já podemos modelar um autômato com essas características e que ficaria então conforme a figura 2, respeitando a seguinte modelagem de uma máquina de Moore:

$M_{lin} = (\Sigma, Q, \delta, estamparia, fim, \Delta, \delta_f)$

onde

$\Sigma = \{x, 0, +, -, 1, 2, L, P, B, V, A, Z, Y, e\}$

$Q = \{estamparia, estruturacao, funilaria, motor, montagem, pintura, fim, erro, fix_E, fix_F, fix_M, fix_P, fix_G\}$

$\delta$ : Tabela 3(b)

$\Delta = \{e, 0, f, s\}$

$\delta_f$ : Tabela 3(a)

*obs: 'x' representa o tipo do carro, 'Y' representa a cor do carro e '+' e '-' representam os modelos possíveis.

(a) Fita de saída

| δf | saída |
|---|---|
| estamparia | 0 |
| estruturacao | 0 |
| funilaria | 0 |
| motor | 0 |
| montagem | 0 |
| pintura | 0 |
| fim | s |
| erro | f |
| fixE | e |
| fixF | e |
| fixM | e |
| fixG | e |
| fixP | e |

(b) Fita de entrada

| δ | x | Y | + | - | 0 | 1 | 2 | e |
|---|---|---|---|---|---|---|---|---|
| estamparia | estruturacao | | | | | | | erro |
| estruturacao | | | | | funilaria | | | fixE |
| funilaria | | | motor | motor | | | | fixF |
| motor | | | | | | montagem | | fixM |
| montagem | | | | | | | pintura | fixG |
| pintura | | fim | | | | | | fixP |
| erro | | | | | | | | |
| fim | | | | | | | | |
| fixE | | | | | funilaria | | | |
| fixF | | | motor | motor | | | | |
| fixM | | | | | | montagem | | |
| fixG | | | | | | | pintura | |
| fixP | | fim | | | | | | |

**Tabela 3. Tabelas de deltas da linha de produção**

A figura 2 representa uma única linha de produção da fábrica, porém, em nossa fábrica faz-se necessário o uso de 8 autômatos como esse para que seja possível trabalhar com 8 carros diferentes, isto é, não podemos fabricar todos os carros em uma mesma linha de montagem já que cada carro tem um conjunto específico de peças e maneiras únicas de serem montados. Em razão disso, iremos utilizar 8 linhas e para coordenarmos essas linhas será necessário a criação de uma ferramenta, que será exposta em breve.

Com nosso autômato da linha de produção modelado e nossa demanda modelada, já temos grande parte do trabalho teórico feito, porém é necessário juntarmos e implementarmos tudo o que temos até então de forma síncrona e conjunta. Para esse feito ser realizado utilizaremos de algumas sacadas que serão descritas a seguir.

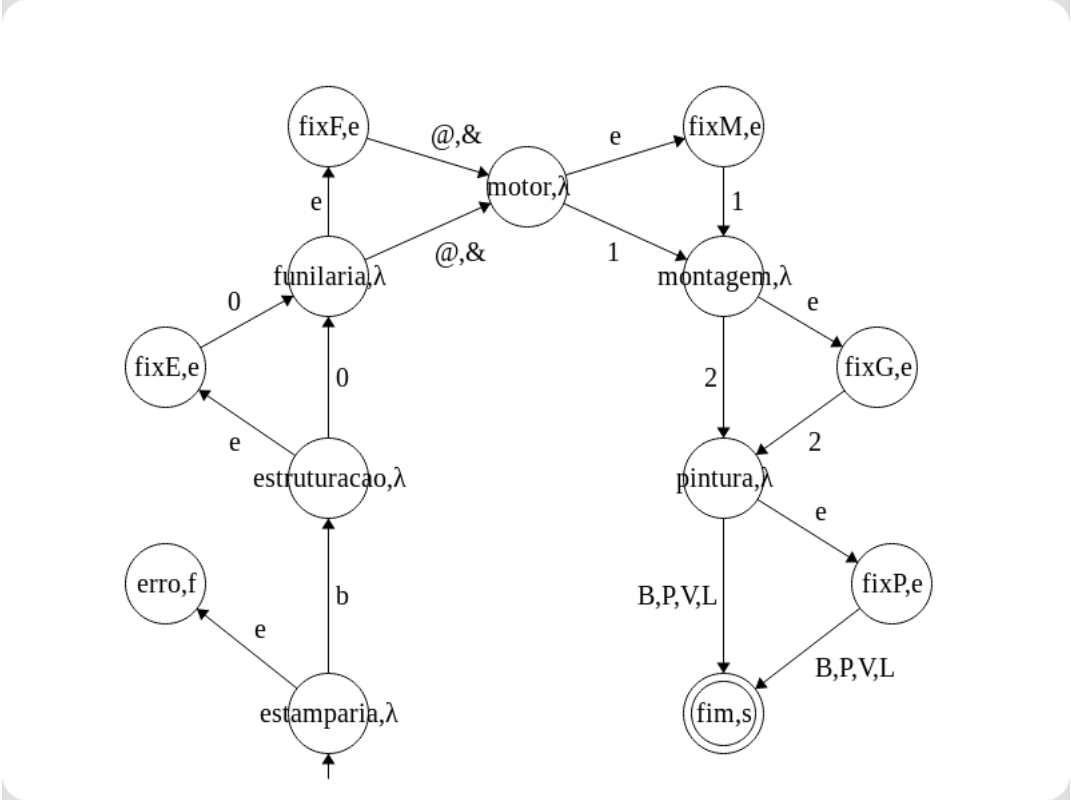


**Figura 2. Exemplo de autômato da linha de produção (linha Bozzola)**

## 5. Implementação computacional

Para a simulação computacional do modelo de autômatos utilizamos a linguagem de programação Python. Pelo fato do problema demandar várias instâncias de autômatos, expressões regulares e outros tipos de interação, optou-se pelo paradigma da orientação de objeto.

### 5.1. Classes e objetos

Este trabalho opera sob as estruturas de autômatos finitos determinísticos (AFD), expressões regulares (ER) e máquinas de Moore.

Para representar os AFDs, usou-se a classe $DFA$. Sua descrição é dado por meio da 5-tupla do autômato referente a esta. A 5-tupla segue o padrão descrito pelo Apêndice 1.

Para representar as ERs utilizou-se a classe $Regex$. Sua descrição é dada pelo padrão de expressões regulares detalhada no Apêndice 2.

Por fim, para representar uma máquina de Moore se utiliza da classe $Moore$. Este objeto é caracterizado pelo padrão informado no Apêndice 3.

### 5.2. Linhas e Fábricas

Entrando no contexto do problema, cada tipo de carro é fabricado na sua respectiva linha de produção. Cada linha de produção, conforme especificado, possui uma capacidade diferente de produtividade. Isto é, uma linha pode possuir 8 estágios de montagem mas

mesmo assim ter capacidade máxima de 4 carros simultaneamente na linha. Para empregar essa característica limitante foi utilizado o artificio das fábricas.

Uma fábrica é um conjunto de linhas de produção. Cada linha de produção é uma simulação virtual de capacidade 1. Ou seja, se uma fábrica possui capacidade de produção de 4 carros simultâneos, nossa fábrica contará com 4 linhas de produção. A linha de produção é instanciada pela classe $Assembly\_Line$. $Assembly\_Line$ é possui uma instância de uma máquina de Moore. A fábrica é uma instância de $Factory$. Além do gerenciamento das linhas de produção ocupadas, cada fábrica é responsável pelo contagem de erros.

## 5.3. A montagem

Uma linha de produção é uma instância da classe $Moore$ e possui duas principais métodos. $run\_with\_word$, responsável por nomear um carro utilizando uma técnica de $hash$, iniciar o método $transition\_to\_state\_with\_input$ com a palavra aceita e por fim validar ou não a produção. Já o método $transition\_to\_state\_with\_input$ é responsável pelo avanço do autômato de maneira propriamente dita. Este avançando possui duas características dignas de nota, uma é a aleatoriedade na transição e outra é a espera para avanço. A aleatoriedade foi empregada aqui para simular uma falha de montagem dado uma confiabilidade da linha (descrito no Apêndice 3). A espera para avanço é definido no inicializar o $pooling$ do $Maestro$, características que serão abordadas em breve.

## 5.4. Maestro

A classe $Maestro$ é o grande coordenador desta implementação. Responsável pela alocação otimizada da produção, avanço da demanda de produção e estatísticas. $Maestro$ possui inicialização informando quais fábricas ele terá de gerenciar e quais são as demandas de cada tipo de carro.

### 5.4.1. $Pooling$

O $pooling$ trabalha com intuito de simular um pseudo-paralelismo. Para cada palavra na demanda produtiva, o $pooling$ procura na listagem de fábricas se a sua fábrica correspondente está disponível. Caso esteja, essa palavra é enviada a produção e a linha é marcada como ocupada por um carro. Após esse procedimento, o maestro realiza a atualização de todas as linhas com a método $update\_factories$.

### 5.4.2. $Update\_factories$

O método $update\_factories$ opera percorrendo todas as fábricas e atualiza o estado atual de produção. Este procedimento possui grande importância para o pseudo-paralelismo do sistema. Como cada fábrica possui um mecanismo de alocação independente, a saturação de uma linha não influencia no seguir das outras.

### 5.4.3. Estruturas

A orientação a objetos propiciou uma grande versatilidade a implementação aqui apresentada. Criação de novas linhas, fábricas e seus usos se mostrou uma bem pouco burocrático devido a linguagem de programação escolhida. A tipagem dinâmica da linguagem facilitou a escrita dos algoritmos referente a simulação pois cariáveis contadoras, variáveis de controle e afins são facilmente remodeladas dado a necessidade. Outro ponto vantajoso a implementação foi o uso das estruturas nativas. Para transições de um autômato foi usado um dicionário. Dicionários são estruturas do tipo chave/valor. No caso, a chave é um tupla $(estadoorigem, símbolo)$, o valor é $estadodestino$. A função delta de um autômato portanto se condensa em apenas um grande dicionário. Outra estrutura muito útil foi a lista. Aplicada nos mais diversos contextos da implementação, podemos destacar seu uso no gerenciamento das instruções de produção das diversas linhas. Com métodos de inclusão, ordenação e remoção nativos, isso não se tornou um gargalo no desenvolvimento.

### 5.4.4. Elementos de simulação

Podemos elencar como elementos de simulação o contador de $ticks$. A cada iteração do $update\_factories$ um $tick$ é incrementado. Assim é possível ter uma noção temporal da fábrica.

### 5.4.5. Vantagens

Podemos destacar uma grande vantagem da nossa estratégia, e por consequência do nosso código, é o caráter genérico e flexível. Caso seja adicionado novos estados ao autômato, todo o sistema se dimensiona para se adequar a nova escala. Outra vantagem é o fato de tudo ser configurável por meio de autômatos e expressões regulares. Sendo somente necessário conhecimento desses formalismos para alterar o funcionamento da fábrica de automóveis. Um diferencial é a possibilidade de acompanhar o avanço da produção por meio da geração gráfica dos autômatos utilizando a ferramenta Graphviz (2009). Para visualizar as saídas geradas pelo Graphviz são gerados arquivos PDF com nome no formato $numero\_de\_serie - estado$. Esses arquivos estarão disponíveis durante e ao fim da execução na pasta *saida* criada na raiz deste presente programa.

### 5.4.6. Desvantagens

Uma desvantagem desta implementação é o uso excessivo de processamento do computador. Como podemos observar no trecho de código abaixo, os autômatos possuem uma complexidade $O(q)$, sendo $q$ a quantidade de estados. Devido ao artificio utilizado para o pseudo-paralelismo, temos que cada linha de produção possui $k$ autômatos, sendo $k$ a capacidade máxima de carros simultaneamente na linha. Considerando que a nossa simulação possui $n$ linhas de produção, isto é, $n$ modelos diferentes de carros, a complexidade final da execução esta na ordem de $O(nkq)$. Essa análise da complexidade do algoritmo não leva em conta operações para o funcionamento da simulação como é o caso da inserção em listas ($O(1)$), busca em uma lista ou de um carácter em uma *string* ($O(n)$),

entre outros métodos nativos. A relação completa pode ser encontrada na referência da linguagem na seção *Time Complexity* (2017).
A complexidade $O(nkq)$ pode ser considerada perigosa, pois podemos ter casos onde $n = k = q$, resultando numa complexidade $O(n^3)$.

```
    def run_with_word(self, word):
        self.go_to_initial_state()
        for letter in word:
            self.transition_to_state_with_input(letter)
            continue
        return self.in_accept_state()

    def transition_to_state_with_input(self, letter):
        if self.valid:
            if (self.current_state, letter) not in self.delta_function.
                keys():
                self.valid = False
                return
            self.current_state = self.delta_function[(self.
                current_state, letter)]
            self.current_letter = letter
            self.output += letter
        else:
            return
```

**Algoritmo 1. Trecho da implementação do AFD (*class_DFA.py*)**

A implementação completa está disponível na página do projeto no GitHub (2019).

## 6. Experimentos

Com o projeto devidamente modelado, projetado e implementado, faz-se necessário a realização de testes e experimentos para validação de resultados. A partir disso, será reportado nos próximos tópicos dois experimentos: uma demanda válida e outra demanda inválida.

### 6.1. Estudo de caso com demanda válida

Antes de analisar e demonstrar a execução do código, é necessário ressaltar as configurações de cada linha, isto é, qual a confiabilidade e qual a capacidade máxima de produção simultânea. Essas informações são facilmente alteradas conforme descrito no Apêndice 3 e para o atual experimento segue conforme Tabela 4.

| Linha | Bozzola | Gasparini | Jonk | Brugmann | March | Utiama | Weiss | Granza |
|---|---|---|---|---|---|---|---|---|
| Confiabilidade (%) | 95 | 15 | 85 | 90 | 55 | 67 | 80 | 100 |
| Capacidade | 4 | 6 | 7 | 6 | 5 | 6 | 4 | 5 |

**Tabela 4. Tabela de configuração do experimento**

Para o experimento de um demanda válida gerou-se um pedido d 70 carros. O foco deste experimento é analisar as estatísticas referentes a quantidade de carros que são produzidos normalmente, carros que tem erros comuns mas foram reparados, carros com

erros fatais, a fábrica parando após a ocorrência de 7 erros e como o ambiente de gerenciamento provê imagens das linhas de produção.
Executando o algoritmo via terminal as seguintes informações são expostas na tela: estatísticas gerais em relação a produção (Tabela 5.a) e também as fitas cada produção separadas por linhas de produção (Tabela 5.b). Para fins de demonstração foi suprimido as demais linha e exposto somente as estatísticas de duas linhas e a fita de produção da linha Gasparini:

(a) Estatísticas de duas linhas

```
--------------
ESTATISTICAS
--------------

Tempo de execução 100 ticks
Linha Bozzola
  Carros produzidos = 6
  Erros = 1
Linha Gasparini
  Carros produzidos = 1
  Erros = 7
Linha parou devido excesso de falhas
```

(b) Fita da Linha Gasparini

```
Linha Gasparini
  Produzidos
    1 -> 000000s
  Falhas fatais (7)
    1 -> ee0
    2 -> 0
    3 -> 0
    4 -> 0
    5 -> 0
    6 -> 0
    7 -> 0
```

**Tabela 5. Tabelas de resultados**

Percebe-se então características essenciais e objetivos alcançados, como a parada da linha devido ao excesso de falhas, contagem de erros, todo o histórico da linha na fita. A vizualização da fábricação é obtida, como já comentado, pelo software Graphviz. Nas próximas imagens esta o exemplo de um carro da linha Gasparini em três estados diferentes da produção.

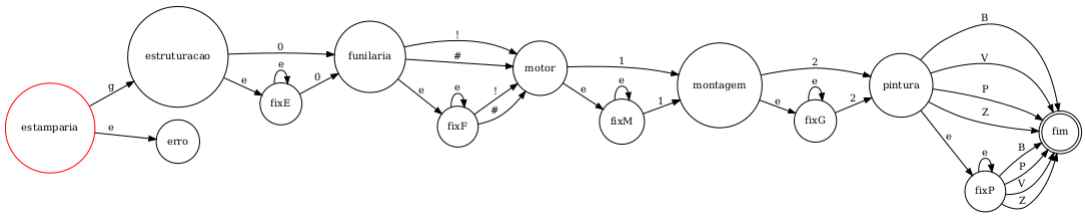


**Figura 3. Produção de um carro Gasparini no estado inicial**

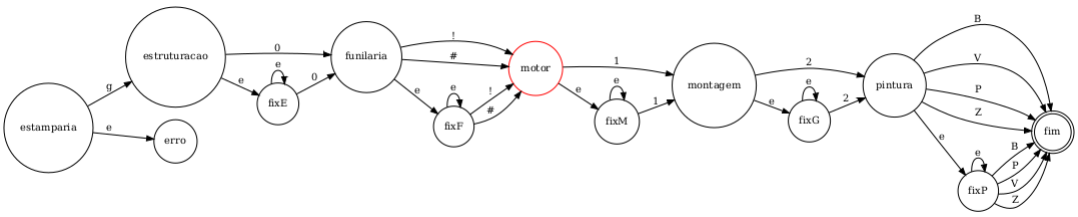


**Figura 4. Produção de um carro Gasparini em um estado intermediário**

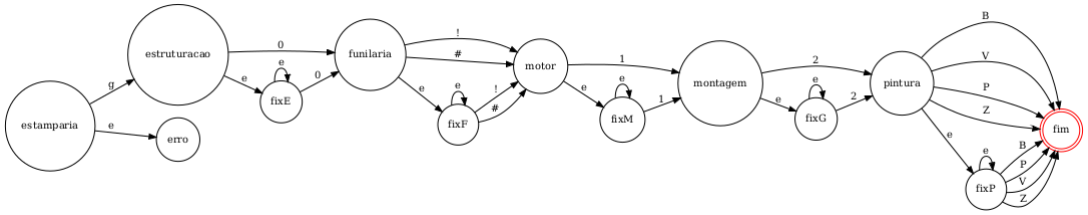


**Figura 5. Produção de um carro Gasparini no estado final**

### 6.2. Estudo de caso com demanda inválida

Se requisitarmos ao algoritmo uma demanda inválida, como por exemplo, um carro com uma cor indisponível, um modelo inexistente ou até mesmo uma demanda com a ausência do separador ”;”entre cada pedido. O que será obtido será apenas um aviso de demanda inválida. Não havia muito mais o que se fazer a respeito em relação a essa parte, já que a demanda não é válida, a imagem abaixo demonstra com clareza o resultado:

```
Deseja gerar uma demanda aleatória?
(s ou n) ->n
Digite a demanda
 ->bZ!;
Demanda inválida!
 Fechando!
```

**Figura 6. Demanda inválida sendo tratada**

## 7. Conclusões

Em razão do trabalho apresentado, podemos concluir que existe uma vasta possibilidade para o uso dos autômatos em diversas áreas de estudos. Além de poder ser implementado também para processos com fins mercadológicos como visto no exemplo da fábrica. Considerando que os formalismos e modelos utilizados e criados para tal projeto não são de tamanha complexidade. Se nos aprofundarmos ainda mais no campo de Linguagens Formais e Autômatos, o potencial e versatilidade possíveis com o uso de determinado campo pode ser ainda maior.

Processos e programas que se baseiam em linguagens formais tem muito mais liquidez ao que se refere à praticidade de se modificaram à problemas distintos, isso é, uma vez que o processo é alterado conforme o formalismo, ao se alterar o formalismo, o mesmo programa pode fazer tarefas completamente diferentes apenas por uma mudança de entrada.

Essa praticidade também está presente no código realizado pela equipe, de forma que mesmo a estrutura do trabalho sendo feito para mimetizar o funcionamento de uma fábrica de automóveis, esse mesmo trabalho pode reconhecer mudanças de estados completamente diferentes das implementadas apenas com a mudança das 6-tuplas e das máquinas de produção, possibilitando assim, usá-lo em aplicações diversas.

## Referências

[1] Carvalho I., *Como funciona uma linha de montagem de automóveis?*, Quatro Rodas, 2011.

[2] Ellson J., Gansner E., Hu Y., Janssen E., North S. et al., *Graphviz - Graph Visualization Software*, https://www.graphviz.org, 2009.

[3] Gasparini V., Granza R. e Bruggmann K., *Github: Simulando um fábrica de automóveis com autômatos*, https://github.com/VGasparini/LFA, 2019.

[4] Hopcroft J. E., Ullman J. D. e Motwani R., *Introdução à Teoria dos Autômatos, Linguagens e Computação*, Editora Campus, 2nd Edição, 2003.

[5] Python Software Fundation. *References - Time Complexity*, https://wiki.python.org/moin/TimeComplexity, 2017.

# Apêndice 1

O modelo para escrita do autômato finito determinístico esta abaixo descrito. Modelo gerado com base na representação por 5-tuplas.

```
#nome do automato
#alfabeto
#listagem dos estados
#quantidade de transicoes (n)
#as n transicoes no estilo de->para:letra
#estado inicial
#estado final
```

Segue exemplo

```
Teste
a b c
q0 q1 q2 q3
3
q0->q1:a
q1->q2:b
q2->q3:c
q0
q3
```

# Apêndice 2

O modelo para escrita de uma expressão regular. Note que aqui não é especificado os símbolos variáveis, somente a concatenação.

```
{}a{}b{}
Onde {} determina o inicio de um ou
```

Segue exemplo

```
{}0{}12{}

Aceita a palavra
a0b12c
```

# Apêndice 3

O modelo para escrita da máquina de Moore esta abaixo descrito. Modelo gerado com base na representação por 7-tuplas.

```
#nome do automato
#alfabeto
#listagem dos estados
#quantidade de transicoes (n)
#as n transicoes no estilo de->para:letra
#estado inicial
#estado final
#alfabeto da funcao
#quantidade de transicoes (m)
#as m funcoes no padrao estado->simbolo
#confiabilidade da linha
#capacidade maxima da linha
```

Segue exemplo

```
Teste
a b c
q0 q1 q2 q3
3
q0->q1:a
q1->q2:b
q2->q3:c
q0
q3
0 1
q0->0
q1->1
q2->0
q3->1
100
2
```