**Leitura e Gravação de autômato finito em arquivo XML**

A leitura e gravação de autômatos finitos em XML (*eXtensible Markup Language*) é compatível com o software JFlap 7, através da criação de arquivos no formato JFF.

Primeiramente, em relação à leitura, a partir da elaboração do autômato são extraídos dados em relação aos estados, tais como identificador, nome, coordenadas e se é um estado inicial e/ou final; e sobre as transições, registrando a origem e o destino assim como o símbolo de aceitação da transição. Adicionalmente são incluídas informações de codificação de caracteres e versão do XML, sendo importante ressaltar que estão versão não suporta a presença de notas de texto nos estados ou no autômato como um todo, as quais se presentes serão ignoradas.

Em relação à gravação, foi utilizada a API nativa do Java para interpretação de XML através do uso de classe de “*parseamento”* DocumentBuilderFactory, a qual viabiliza a extração das *tags, sub-tags* e seus respectivos atributos.

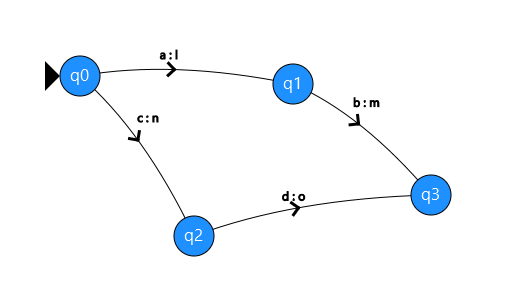


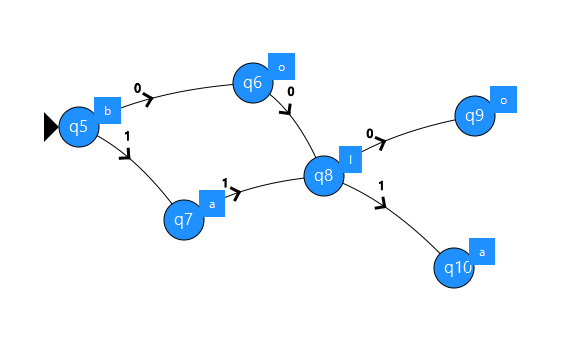
**Máquinas de Mealy e Moore**

Inicialmente, foram realizadas adaptações na versão inicial de representação de autômatos, de forma a representar as devidas singularidades de ambas máquinas adicionais.

Em relação a implementação do algoritmo de aceitação de uma ou múltiplas entradas para cada máquina, partiu-se do princípio de que o autômato está representado corretamente e é determinístico, desta forma, a partir de uma cadeia de entrada, faz-se necessário somente o percurso do autômato com saída enquanto houvesse transições válidas, caso não houvesse transições válidas ou esgota-se as transições o processo é encerrado e a saída exibida, cabe ressaltar que em autômatos com saída não há presença de estados finais.

Por fim, é importante relembrar que para as Máquinas de Moore, o símbolo é adicionado ao entrar no estado, enquanto para as Máquinas de Mealy, o símbolo é adicionado caso a transição seja realizada.





**Conversões**

**Expressão para Autômato**

A implementação baseia-se no algoritmo de conversão de Thompson e juntamente em uma solução iterativa apoiada pelo uso de estruturas de dados de pilhas em cada abertura e fechamento de parênteses presentes na expressão de forma a compor sub-expressões. Esta implementação suporta a presença de fechamentos abertos e positivos.

**Autômato para Expressão**

A implementação inicialmente verifica o número de estados finais do autômato finito, caso este seja maior do que um, são criadas transições vazias de forma a existir somente um estado final, em seguida, para cada estado não inicial Ei, são criadas transições duplas que envolvem todas as transições que chegam ao estado Ei e partem dele, cada par (transição que chega, transição que sai) gera uma nova transição. Quando todos os pares forem esgotados, somente existirá um que do estado inicial e chega ao único final, representando toda a expressão regular.

**Autômato para Gramática**

A implementação após armazenar o respectivo autômato finito em uma estrutura de dados, atribui para cada estado uma respectiva letra, e cada transição de cada estado é transformada em uma regra composta pelo símbolo de aceitação da transição e a letra correspondente ao estado alvo. Todo estado final é acrescido de uma transição vazia.

**Gramática para Autômato**

A implementação avalia cada regra da gramática individualmente, em que são extraídos o símbolo de aceitação da transição e o estado alvo. No autômato é adicionado um estado final, em que todas as transições que possuírem somente um símbolo terminal ou forem vazias serão direcionados ao estado adicional.