Trabalho Final LFA

Alunos: Ygor Finger Kappaun e Pedro Guilherme Merisio Schabarum

O objetivo deste trabalho é criar um aplicação que converta uma Gramática Regular (no formato BNF) em um Autômato Finito Determinístico Mínimo. A natureza não-determinística de muitas gramáticas é um desafio significativo nesse processo, pois uma única produção pode resultar em vários caminhos. Para resolver isso, a aplicação segue um processo em etapas: primeiro, a gramática é transformada em um Autômato Finito Não-Determinístico (AFND) para lidar com ambiguidades e transições vazias; depois, o algoritmo de Construção de Subconjuntos é usado para tornar o autômato determinístico; e, finalmente, o algoritmo de Particionamento por Equivalência é empregado para minimizar os estados.

A implementação foi feita na linguagem Python, selecionada por sua solidez na manipulação de strings e estruturas de dados. Os objetivos particulares do projeto, como detalhado, são:

- **Carregamento**: leitura de um arquivo de texto que contém gramáticas com produções terminais, não-terminais e transições vazias (epsilon).
- **Conversão**: transformação das regras de produção em um grafo de transições não determinístico (AFND).
- **Determinização**: processo de conversão de um AFND em um AFD funcional, eliminando o não-determinismo.
- **Otimização**: redução do AFD ao menor número possível de estados, mantendo inalterada a linguagem reconhecida.
- **Persistência**: Exportação da tabela de transições final para um arquivo no formato CSV.

Algoritmo de Conversão (GR para AFND): O processo inicia-se com o *parser* do arquivo de texto. O algoritmo lê cada produção da Gramática Regular e a mapeia diretamente para uma transição no grafo.

- Regras da forma <A> ::= x<B> geram uma transição delta(A, x)={B}.
- Regras da forma <A> ::= x (terminal puro) geram uma transição para um estado final auxiliar virtual, denominado Z_FIM.
- Esta etapa preserva o não-determinismo natural da gramática (ex: <S> ::= a<A> | a<B>), resultando em um AFND.

Para converter o AFND em AFD, implementou-se o algoritmo de Construção de Subconjuntos. O autômato determinístico é construído incrementalmente

1. O estado inicial do AFD é definido como o conjunto contendo o estado inicial do AFND.

2. Para cada novo conjunto de estados descoberto, calcula-se a união das transições de todos os seus membros para cada símbolo do alfabeto.
3. O resultado dessa união torna-se um novo estado no AFD (representado, no código, por um frozenset para garantir imutabilidade e permitir seu uso como chave de dicionário).

Este método garante que, se o AFND podia estar em A ou B, o AFD estará no estado único {A,B}.

Minimização: A otimização do AFD é realizada através do algoritmo de minimização por particionamento (similar ao Método de Moore/Hopcroft).

1. Primeiramente, os estados são classificados em dois grupos principais: Finais e Não-Finais.
2. De forma iterativa, o algoritmo analisa se todos os estados de um mesmo grupo apresentam o mesmo comportamento (ou seja, transicionam para grupos equivalentes com símbolos idênticos).
3. Um grupo é fragmentado quando exibe comportamentos diferentes.
4. O processo termina quando não há mais divisões possíveis (estabilidade), levando à fusão dos estados equivalentes.

Resultados e Testes

Para validar a pipeline completa, utilizou-se uma gramática projetada especificamente para gerar redundância de estados.

Arquivo de Entrada (entrada.txt):

<S> ::= a<A> | b<B>

<A> ::= c<F>

<B> ::= c<F>

<F> ::= ε

Observe que, de forma lógica, os não-terminais <A> e <B> são equivalentes: ambos consomem o símbolo 'c' para atingir o estado final e não apresentam outras ramificações.

Saída: Após o processamento (Conversão -> Determinização -> Minimização), o software gerou a seguinte tabela de transições (exportada em CSV):

Estado	Simbolo	Destino	Eh_Inicial	Eh_Final
AB	c	F		
F	-	-		X
S	a	AB	X	
S	b	AB	X	

Nota-se que o algoritmo reconheceu corretamente a equivalência entre os trajetos. O autômato minimizado unificou os dois estados intermediários distintos para as ramificações 'a' e 'b' em um único estado composto AB. Isso demonstra a eficácia da etapa de minimização, que diminui a complexidade do autômato sem modificar a linguagem reconhecida (as palavras "ac" e "bc").

Conclusão

O estudo mostrou a criação de uma ferramenta abrangente para trabalhar com Linguagens Regulares. A aplicação da conversão por meio de AFND intermediário provou ser eficiente para lidar com as ambiguidades próprias das Gramáticas Regulares, ao passo que a fase de minimização assegurou a eficácia do autômato resultante.

Os testes realizados confirmaram que o sistema pode lidar com gramáticas complexas, resolver não-determinismos por meio da construção de subconjuntos e eliminar estados redundantes. Como trabalho futuro, a ferramenta poderia ser aprimorada para aceitar expressões regulares como entrada ou para criar automaticamente visualizações gráficas (grafos) dos autômatos.