Relatório do Projeto: Simulador de Linguagens Regulares

Alunos:

- Marcus Vinicius Ramos de Araujo
- Guilherme Colonhese Camargo

Introdução

Este relatório apresenta o desenvolvimento de um simulador de linguagens regulares que permite a criação, edição e simulação de expressões regulares, autômatos finitos (determinísticos e não determinísticos) e gramáticas regulares. O objetivo principal é proporcionar uma ferramenta interativa que auxilie no aprendizado e compreensão dos conceitos fundamentais da Teoria da Computação.

Funcionalidades Implementadas

1. **Expressões Regulares:**
    - Entrada e validação de expressões regulares seguindo a sintaxe padrão.
    - Simulação de correspondência de expressões regulares com cadeias de entrada fornecidas pelo usuário.
    - Indicação clara de aceitação ou rejeição das cadeias em relação à expressão regular.
2. **Autômatos Finitos:**
    - Editor visual para criação e edição de autômatos finitos, permitindo adicionar estados, transições e definir estados iniciais e de aceitação.
    - Suporte a múltiplos símbolos em uma única transição, inclusive para auto-transições (self-loops).
    - Opção para definir o autômato como determinístico (AFD) ou não determinístico (AFND).
    - Permissão para múltiplos estados iniciais, conforme necessário.
    - Simulação de autômatos com múltiplas entradas, exibindo resultados individuais para cada uma.
    - Simulação passo a passo (teste de mesa), permitindo ao usuário acompanhar a execução do autômato em cada etapa.
    - Destaque visual dos estados atuais durante a simulação, facilitando a compreensão do funcionamento do autômato.
3. **Gramáticas Regulares:**
    - Editor para definição de gramáticas regulares, incluindo não-terminais, terminais e produções.
    - Conversão automática de gramáticas regulares em autômatos finitos não determinísticos.
    - Exibição do autômato resultante da gramática definida.
    - Simulação do autômato gerado a partir da gramática com cadeias de entrada fornecidas pelo usuário.

Descrição da Implementação

- **Front-End:**
  - Desenvolvido em React com TypeScript.
  - Utilização da biblioteca `react-flow-renderer` para criação e manipulação dos grafos representando os autômatos.
  - Uso do `react-bootstrap` para componentes de interface do usuário.
- **Componentes Principais:**
  - **RegexInput e RegexSimulator:**
    - `RegexInput`: Componente para entrada de expressões regulares pelo usuário.
    - `RegexSimulator`: Componente que recebe a expressão regular e uma entrada, realizando a simulação e indicando se a entrada é aceita.
  - **AutomatonEditor:**
    - Permite a criação e edição visual de autômatos finitos.
    - Implementa funcionalidades para adicionar estados, transições, e definir estados iniciais e de aceitação.
    - Suporta múltiplos símbolos em transições e diferencia entre AFD e AFND.
    - Permite múltiplos estados iniciais.
  - **AutomatonSimulator:**
    - Realiza a simulação de autômatos com entradas fornecidas pelo usuário.
    - Suporta múltiplas entradas, exibindo resultados individuais.
    - Oferece uma opção de simulação passo a passo, permitindo ao usuário acompanhar cada etapa da execução do autômato.
    - Destaca visualmente os estados atuais durante a simulação.
  - **GrammarEditor:**
    - Permite ao usuário definir gramáticas regulares, incluindo não-terminais, terminais e produções.
    - Converte a gramática definida em um autômato finito não determinístico.
    - Exibe o autômato gerado e permite sua simulação.
- **Funcionalidades Específicas:**
  - **Permissão de Múltiplos Símbolos em Transições:**
    - Ao adicionar transições, o usuário pode inserir múltiplos símbolos separados por vírgulas.
    - Transições com os mesmos estados de origem e destino são combinadas em uma única aresta, com os símbolos concatenados no rótulo.
  - **Visualização de Auto-Transições (Self-Loops):**
    - Implementação de um componente personalizado `SelfLoopEdge` para representar auto-transições.
    - Ajuste do caminho SVG para desenhar loops acima dos nós, garantindo a correta orientação dos rótulos.
  - **Simulação Passo a Passo:**

        - O usuário pode optar por simular o autômato passo a passo.
        - A cada passo, o usuário visualiza o símbolo atual, os estados ativos e pode avançar para o próximo símbolo.
        - Estados ativos são destacados no editor visual durante a simulação.
    - **Múltiplos Estados Iniciais:**
        - Possibilidade de marcar múltiplos estados como iniciais.
        - Atualização dinâmica da visualização para refletir os estados iniciais adicionais.

Desafios e Soluções

- **Representação de Múltiplos Símbolos em Transições:**
    - *Desafio:* Garantir que múltiplos símbolos em uma transição sejam representados de forma clara e sem sobreposição de arestas.
    - *Solução:* Combinar símbolos em uma única aresta entre dois estados, concatenando os símbolos no rótulo e evitando a criação de arestas duplicadas.
- **Visualização de Auto-Transições:**
    - *Desafio:* Desenhar auto-transições que sejam visualmente claras e com rótulos corretamente orientados.
    - *Solução:* Desenvolver o componente `SelfLoopEdge` personalizado, ajustando o caminho da curva Bezier para posicionar o loop e corrigir a orientação do texto.
- **Simulação Passo a Passo:**
    - *Desafio:* Permitir que o usuário acompanhe a execução do autômato em cada etapa, visualizando os estados atuais e transições.
    - *Solução:* Implementar estados internos no `AutomatonSimulator` para controlar o índice atual, estados ativos e conclusão da simulação, além de destacar os estados ativos no editor visual.
- **Múltiplos Estados Iniciais:**
    - *Desafio:* Adaptar a lógica e visualização para suportar múltiplos estados iniciais sem comprometer a usabilidade.
    - *Solução:* Atualizar o modelo de dados para permitir múltiplos estados iniciais e ajustar a visualização, criando nós e arestas de início para cada estado inicial.

Testes Realizados

- **Validação de Expressões Regulares:**
    - Testes com expressões regulares válidas e inválidas para assegurar que a validação está funcionando conforme esperado.
    - Verificação da correspondência de cadeias de entrada com as expressões regulares.
- **Simulação de Autômatos:**
    - Criação de autômatos finitos determinísticos e não determinísticos, incluindo casos com múltiplos estados iniciais e transições com múltiplos símbolos.
    - Simulação de entradas válidas e inválidas, confirmando a aceitação ou rejeição conforme esperado.

    - Utilização da simulação passo a passo para verificar a atualização correta dos estados ativos e a progressão através dos símbolos de entrada.
- **Conversão de Gramáticas em Autômatos:**
    - Definição de diversas gramáticas regulares e conversão para autômatos finitos não determinísticos.
    - Simulação dos autômatos gerados para validar a correspondência com as linguagens definidas pelas gramáticas.

Conclusão

O desenvolvimento do simulador de linguagens regulares proporcionou uma compreensão aprofundada dos conceitos teóricos relacionados a expressões regulares, autômatos finitos e gramáticas regulares. A implementação das funcionalidades exigidas, bem como a superação dos desafios encontrados, resultou em uma ferramenta robusta e interativa que pode ser utilizada como suporte educacional no estudo da Teoria da Computação.