Ferramenta educacional para auxiliar no ensino do modelo fork-join

Lucas da Silva Marcos[1], **Rodolfo Adamshuk Silva**[1]

[1]Universidade Tecnológica Federal do Paraná (UTFPR)
Dois Vizinhos, Paraná

lucasmarcos@alunos.utfpr.edu.br, rodolfoa@utfpr.edu.br

Abstract.

Resumo. *A importância da programação concorrente no campo da ciência da computação é inegável. No entanto, a complexidade inerente ao ensino desta disciplina em ambientes acadêmicos é um desafio significativo. Este artigo propõe um compilador como ferramenta educacional destinada a facilitar o ensino do modelo de programação fork-join, um paradigma utilizado na programação concorrente.*

1. Introdução

O ensino de programação concorrente enfrenta um desafio significativo devido à natureza complexa e abstrata dos conceitos envolvidos. A programação concorrente é um paradigma de programação que envolve a execução simultânea de múltiplas tarefas, introduz uma nova camada de complexidade que muitos estudantes acham difícil de compreender. A comunicação entre processos, que pode ocorrer por exemplo por *memória compartilhada* ou *passagem de mensagem*. *problema* x *tarefa* x *processo*. A compreensão desses conceitos é essencial para o desenvolvimento de sistemas robustos e escaláveis, e a falta de experiência prática e ferramentas de ensino adequadas pode dificultar a aprendizagem.

O modelo fork-join, introduzido por [Conway 1963], apresenta desafios adicionais para os estudantes, especialmente em relação à decomposição adequada de tarefas e à gestão eficiente de recursos compartilhados.

Com a necessidade de aprender e por em prática os conceitos de programação concorrente, este trabalho tem por objetivo desenvolver uma ferramenta de ensino que auxilie em tais objetivos.

Mesmo utilizando um modelo mais simples de programação concorrente, nota-se ainda os estudantes ainda possuem dificuldades para utilizar a sintaxe do modelo.

Diversas abordagens na literatura apresentam o uso de ferramentas educacionais para o ensino de programação para estudantes universitários como um recurso de apoio aos estudantes.

[Raiol et al. 2015] apresenta uma ferramenta para o ensino de programação Logo para estudantes do 1 ano do curso de sistemas de informação. Essa ferramenta ajuda a desenvolverem ou ampliarem suas habilidades com a programação. [Cardoso and Faria 2019] apresenta a ferramenta Scratch no ensino superior. A tecnologia é um forte aliada a educação. Ótimo agregador de beneficios a educação.

A motivação para o desenvolvimento de uma ferramenta educacional que suporte o ensino do modelo fork-join em uma linguagem de alto nível surge da necessidade de tornar os conceitos de concorrência mais acessíveis e tangíveis para os estudantes.

Ao fornecer uma linguagem de programação abstrata e familiar, enriquecida com operações específicas do modelo fork-join, espera-se reduzir a curva de dficuldade no aprendizado e permitir que os alunos se concentrem nos conceitos fundamentais, em vez de se preocuparem com detalhes de implementação.

A ferramenta será desenvolvida utilizando conceitos de compiladores para a verificação da sintaxe dos comandos de fork-join utilizados para o ensino desse conceitos, onde o compilador irá realizar a análise léxica, . . . , e sintática, . . .

O objetivo principal deste trabalho é desenvolver um compilador que permita aos alunos expressar o modelo fork-join em uma linguagem de alto nível de forma eficiente. O compilador fornecerá recursos de análise léxica e sintática para identificar erros no código.

Além disso, a ferramenta educacional incluirá recursos para visualização da decomposição de tarefas em forma de grafo, juntamente com métricas associadas para avaliar o desempenho e a eficiência dos programas desenvolvidos.

2. Trabalhos relacionados

Nessa seção estão descritos os tópicos mais importantes e os trabalhos relacionados para a fundamentação teórica da ferramenta proposta.

2.1. Programação concorrente

[Van Steen and Tanenbaum 2017] e [Coulouris et al. 2001].

É uma maneira de organizar um programa em vários fluxos de execução. A pricilpal dificuldade em programação concorrente é a comunicação e coordenação entre processos. As duas maneiras principais que essa comunicação ocorre é por memoria compartilhada e passagem de mensagem.

Na memória compartilhada os processos concorrentes compartilham um espaço de memória comum, permitindo a comunicação direta entre eles por meio da leitura e escrita em variáveis compartilhadas.

E na passagem de mensagem, os processos concorrentes se comunicam trocando mensagens entre si, sem compartilhar diretamente memória. As mensagens podem ser enviadas e recebidas através de canais de comunicação específicos.

Também encontramos em programação concorrente os conceitos de problema, tarfea e processo. Problema refere-se a uma questão ou desafio a ser resolvido por um sistema computacional. Tarefa é uma unidade de trabalho dentro de um programa ou sistema, geralmente relacionada à resolução de uma parte específica de um problema maior. Processo é uma instância em execução de um programa em um sistema computacional. Cada processo tem seu próprio espaço de memória e fluxo de execução, podendo realizar múltiplas tarefas de forma concorrente.

2.2. Fork-join

[Conway 1963] e [Dennis and Van Horn 1966].

Um exemplo de código fork-join.

2.3. Compiladores

Compiladores são programas de computador que traduzem de uma linguagem para outra [Louden 2004].

As fases de um compilador incluem:

Análise Léxica. Análise Sintática. Análise Semântica. Geração de Código Intermediário. Otimização de Código. Geração de Código Executável.

A análise léxica é a primeira fase do processo de compilação, onde o código-fonte é lido e convertido em tokens. Esses tokens representam os componentes básicos da linguagem, como palavras-chave, identificadores, operadores e símbolos especiais.

A análise sintática é a segunda fase do processo de compilação, onde os tokens gerados pela análise léxica são organizados em uma estrutura hierárquica que representa a sintaxe do programa. Esta fase verifica se o código-fonte está gramaticalmente correto de acordo com as regras da linguagem de programação.

2.4. Recursos educacionais para ensino de programação

[Raiol et al. 2015] e [Cardoso and Faria 2019].

3. Proposta

A natureza dinâmica da criação e sincronização de threads pode resultar em problemas de concorrência, como condições de corrida e deadlocks, que são difíceis de identificar e corrigir sem um entendimento profundo do modelo fork-join. Como resultado, muitos alunos enfrentam dificuldades ao desenvolver e depurar programas baseados neste paradigma.

O desenvolvimento de um compilador e plataforma com o foco no uso em sala de aula.

Será especificado uma gramática formal da linguagem de alto nível utilizada. A linguagem possuirá os primitivos Fork e Join como modelo de concorrência.

Será construído um analisador léxico, reportando aos alunos erros nos programas.

Será construído um analisador sintático, que modelara a estrutura do programa e reportará erros de construção.

Como saída esperada teremos, além das mensagens de erro ou de sucesso, um grafo de tarefas e dependências, além de métricas do programa especificado, como o caminho crítico e o grau máximo de concorrência.

Também será construído uma plataforma web, como um editor de textos integrado e renderização do grafo e métricas. Um exemplo de plataforma sendo o Compiler Explorer https://godbolt.org.

4. Metodologia

Será um pesquisa aplicada e exploratória.

Aplicaremos o protótipo em sala de aula, coletaremos dados e feedback dos alunos. O feedback será uma pesquisa qualitativa com as opiniões dos alunos sobre o uso do sistema em um formulário online.

O cronograma das atividades proposta está descrito abaixo.

4.1. Atividades

- **Atividade 1**: atividade 1;
- **Atividade 2**: atividade 2.

	Julho	Agosto	Setembro	Outubro	Novembro	Dezembro
Atividade 1	■					
Atividade 2		■				
Atividade 3			■			

5. Resultados esperados

Espera-se que a ferramenta proposta contribua significativamente para o aprimoramento do ensino de programação concorrente, fornecendo um recurso para o aprendizado dos alunos e auxiliando os professores no processo de ensino.

Referências

Cardoso, L. R. and Faria, D. d. S. E. (2019). O uso do scratch como ferramenta de auxílio no ensino superior. *Anais do Seminário Científico do UNIFACIG*, (5).

Conway, M. E. (1963). A multiprocessor system design. In *Proceedings of the November 12-14, 1963, fall joint computer conference*, pages 139–146.

Coulouris, G., Dollimore, J., and Kindberg, T. (2001). *Sistemas distribuidos*, volume 6. Addison Wesley Madrid.

Dennis, J. B. and Van Horn, E. C. (1966). Programming semantics for multiprogrammed computations. *Communications of the ACM*, 9(3):143–155.

Louden, K. (2004). *Compiladores - Princípios e Práticas*. Pioneira Thomson Learning.

Raiol, A. A., Sarges, J., Souza, A., Silva, S., and BEZERRA, F. d. L. (2015). Resgatando a linguagem de programaçao logo: Uma experiência com calouros no ensino superior. WORKSHOP SOBRE EDUCAÇÃO EM COMPUTAÇÃO.

Van Steen, M. and Tanenbaum, A. S. (2017). *Distributed systems*. Maarten van Steen Leiden, The Netherlands.