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

Rascunho.

- seis parágrafos, dois problema, dois motivação, dois objetivo
- concorrência
- fork-join
- linguagem de alto nível
- ferramenta de ensino
- compilador
- grafo e métricas

O ensino de programação concorrente enfrenta um desafio significativo devido à natureza complexa e abstrata dos conceitos envolvidos. *A concorrência, que envolve a execução simultânea de múltiplas tarefas*, introduz uma nova camada de complexidade que muitos estudantes acham difícil de compreender. A compreensão *dos conceitos* (memória compartilhada/passagem de mensagem problema/tarefa/processo) é essencial para o desenvolvimento de sistemas robustos e escaláveis, a falta de experiência prática e ferramentas adequadas pode dificultar o aprendizado eficaz desses conceitos.

O modelo fork-join, introduzido por [Conway 1963] em 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 torar os conceitos de PC . . . , este trabalho tem por objetivo desenvolver uma ferramenta . . . .

Comunicação por memória compartilhada Comunicação por troca de mensagens Coordenando o acesso aos recursos Dependências Condições de corrida Exclusão mútua

Mesmo utilizando um modelo mais simples de programação concorrente, 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.

Logo e Scratch no Brasil. Fulano et al (200XX) apresenta uma ferramenta para o ensino de programação XXX para estudantes do XX anos do curso de XXXX. Essa ferramenta ajuda no xxxxxxx. Ciclano et al. (20XX) apresenta a ferramenta XXX ....

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 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. 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

Referencial teórico

Paradigmas de programação (memória compartilhada/ passagem de mensagem) Problema/tarefa/processo/aplicação.

Referência Exemplo do artigo (código)

Usar os conceitos do livro (compilador) todas as fase e um subseção para léxica e sintática.

Trabalhos relacionados (compiladores)

Artigos sobre o modelo fork-join [Conway 1963] e [Dennis and Van Horn 1966].

Livre sobre sistemas distribuídos [Coulouris et al. 2001] e [Van Steen and Tanenbaum 2017].

Livro sobre compiladores [Louden 2004].

Ferramentas educacionais [Raiol et al. 2015].

Material de apoio para aprendizagem de Programação Concorrente.

Compiladores em sala de aula.

Compiladores de Pascal usados para o ensino de matemática.

2.1. Maio

memoria compartilhada e passagem de mensagem problema/tarefa/processo exemplo das fases do compilador

Recursos Educacionais Fork Join (artigos 1960) Computadores (Geral) Análise léxica Análise sintática

(Fork/Join) Compiladores Recursos educacionais para ensino de programação Trabalhos relacionados

2.2. ChatGPT

Problema/Tarefa/Processo:

No contexto da programação concorrente soluções que envolvem tanto a memória compartilhada quanto a passagem de mensagens entre processos. A tarefa de projetar e implementar programas concorrentes que utilizam corretamente esses modelos, torna-se uma prioridade no processo de aprendizado dos alunos.

Exemplo das Fases do Compilador:

As fases do compilador fornecem uma estrutura essencial para a compreensão do processo de tradução de código-fonte em linguagem de programação para código executável. Um exemplo dessas fases inclui:

Análise Léxica: Nesta fase, o compilador examina o código-fonte e divide-o em tokens, que são os elementos básicos da linguagem de programação. Isso envolve a identificação de palavras-chave, identificadores, operadores e símbolos especiais.

Análise Sintática: Após a análise léxica, o compilador utiliza gramáticas formais para verificar a estrutura sintática do código-fonte e garantir que ele esteja corretamente construído de acordo com as regras da linguagem de programação. Esta fase envolve a construção de uma árvore sintática ou uma representação intermediária do código.

Análise Semântica: Esta fase verifica se o código-fonte possui significado e semântica corretos, verificando as regras de escopo, tipos de dados e outras restrições semânticas da linguagem de programação.

Geração de Código Intermediário: Após as fases anteriores, o compilador pode gerar um código intermediário que representa uma forma simplificada do programa original. Este código intermediário é mais fácil de ser manipulado e otimizado em comparação com o código-fonte original.

Otimização de Código: Nesta fase, o compilador realiza uma série de transformações no código intermediário para melhorar sua eficiência e desempenho. Isso inclui técnicas como eliminação de código morto, propagação de constantes e reordenação de instruções.

Geração de Código Executável: Por fim, o compilador traduz o código intermediário otimizado em código executável específico para a plataforma de destino, como código de máquina ou código de byte.

Recursos Educacionais:

No âmbito dos recursos educacionais para o ensino de programação concorrente, é

fundamental fornecer materiais que abordem conceitos específicos, como o modelo fork-join, desde seus primórdios até suas aplicações contemporâneas. Além disso, recursos relacionados à análise léxica e sintática, compiladores e exemplos práticos de implementação de sistemas concorrentes são essenciais para auxiliar os alunos no desenvolvimento de habilidades sólidas nessa área.

Fork Join (Artigos 1960):

Os artigos e documentos relacionados ao modelo fork-join, especialmente aqueles que remontam aos anos 1960, fornecem uma visão histórica e aprofundada do desenvolvimento desse paradigma de programação concorrente. Explorar esses recursos permite aos alunos entender a evolução do modelo fork-join ao longo do tempo e suas implicações nas práticas de desenvolvimento de software.

Computadores (Geral):

Recursos educacionais que abrangem aspectos gerais dos computadores, desde sua arquitetura até seus componentes e funcionamento, oferecem uma base sólida para o entendimento dos princípios subjacentes à programação concorrente. Isso inclui a análise de hardware, sistemas operacionais e redes, que são componentes essenciais no contexto da concorrência e distribuição de tarefas.

Trabalhos Relacionados:

Examinar trabalhos relacionados no campo da programação concorrente e ferramentas educacionais existentes oferece insights valiosos sobre abordagens eficazes de ensino e as necessidades dos alunos. Estudar esses trabalhos permite identificar lacunas no conhecimento e oportunidades de melhoria na oferta de recursos educacionais para programação concorrente.

3. Maybe

A programação concorrente é uma área fundamental da ciência da computação que lida com a execução simultânea de múltiplas tarefas em um sistema computacional. Essa abordagem oferece vantagens significativas em termos de desempenho e eficiência, permitindo que sistemas computacionais realizem várias atividades de forma paralela. No entanto, a complexidade inerente à concorrência apresenta desafios consideráveis no desenvolvimento e na compreensão de sistemas concorrentes.

Modelo Fork-Join:

O modelo fork-join é um paradigma de programação concorrente amplamente utilizado, originado nos anos 1960. Ele oferece uma abordagem para decompor problemas em tarefas independentes que podem ser executadas em paralelo e, posteriormente, reunidas em um único resultado. Este modelo é particularmente relevante devido à sua capacidade de lidar com tarefas de granularidade variável e sua aplicabilidade em uma ampla gama de domínios de computação paralela e distribuída.

Memória Compartilhada e Passagem de Mensagem:

Na programação concorrente, dois modelos principais são frequentemente utilizados para facilitar a comunicação e a coordenação entre processos: memória compartilhada e passagem de mensagem. Enquanto a memória compartilhada envolve o compartilha-

mento direto de dados entre processos concorrentes, a passagem de mensagem requer a comunicação por meio de mensagens entre processos separados. Compreender as nuances e as diferenças entre esses dois modelos é essencial para projetar e implementar sistemas concorrentes robustos e eficientes.

Fases do Compilador:

O processo de desenvolvimento de software envolve a tradução de código-fonte em linguagem de programação para código executável, geralmente realizado por um compilador. As fases do compilador, que incluem análise léxica, análise sintática, análise semântica, geração de código intermediário, otimização de código e geração de código executável, fornecem uma estrutura essencial para entender o funcionamento interno dessas ferramentas. Compreender as fases do compilador é crucial para os alunos que desejam desenvolver habilidades de programação concorrente e entender como suas implementações são traduzidas em código executável.

Recursos Educacionais:

No campo da programação concorrente, a disponibilidade de recursos educacionais adequados desempenha um papel fundamental no processo de aprendizado dos alunos. Isso inclui materiais de ensino que abordam conceitos teóricos e práticos, exemplos de implementação, exercícios práticos e ferramentas de desenvolvimento específicas para programação concorrente. Explorar e utilizar esses recursos educacionais pode melhorar significativamente a compreensão e o domínio dos alunos sobre os princípios da concorrência computacional.

Trabalhos Relacionados:

A revisão de trabalhos relacionados oferece insights valiosos sobre abordagens eficazes de ensino e ferramentas educacionais existentes no campo da programação concorrente. Estudar trabalhos relacionados permite identificar lacunas no conhecimento, oportunidades de pesquisa e tendências emergentes no ensino e na prática da concorrência computacional.

4. 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.

5. 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.

5.1. Atividades

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

	Julho	Agosto	Setembro	Outubro	Novembro	Dezembro
Atividade 1						
Atividade 2						

6. 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

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.