Uma proposta de implementação de árvores rubro-negras em Haskell de forma concorrente

AUTOR: THAÍS S. HÜBNER – BOLSISTA PBIP/UFPEL

EMAIL: TSHUBNER@INF.UFPEL.EDU.BR

ORIENTADOR – ANDRÉ RAUBER DU BOIS

Sumário

- Processadores multicore
- Memória Transacional
- Motivação
- A linguagem Haskell
- Métodos de sincronismo
- STM Haskell
- Proposta de implementação
- Tipos de dados algébricos
- Conclusão

Processadores Multicore

- Possui dois ou mais núcleos completos de processamento no mesmo chip
- Necessita que a programação seja realizada de forma concorrente, para explorar o máximo desempenho da arquitetura
- Utilização de bloqueios(locks) é o modelo mais usual de sincronização porém são de difícil utilização e muito propensos a erros de programação

Memória Transacional

- Nova abstração para programação baseada na ideia de transações de banco de dados
- Maior facilidade de programação
- Ausência de deadlocks
- Composabilidade de código

- Árvores rubro negras são uma aplicação clássica para testar memórias transacionais porque possuem baixa taxa de conflitos

Motivação

- Árvores rubro negras possuem um bom pior caso de tempo de execução (O(log n))
- Mais eficientes em inserção, remoção e busca
- Apesar dessa estrutura ser uma das mais usadas para testar memórias transacionais, não existe uma implementação em STM Haskell
- O objetivo deste trabalho é implementar árvores rubro negras em STM Haskell e comparar essa implementação com outros métodos de sincronização disponíveis em Haskell

A linguagem Haskell

- Estilo declarativo – não há conceito de comandos como atribuição, a exemplo de outras linguagens, e sim de *casamento de padrões*
- Conceitos sofisticados – polimorfismo, funções generalizadoras de alta-ordem, avaliação de expressões/funções sob demanda ou preguiçosa (lazy evaluation)
- Recursividade
- Fácil e rápido aprendizado
- Aplicações – prototipação em geral, ensino de programação, inteligência artificial, construção de compiladores, etc.

Métodos de sincronismo

- Haskell possui diferentes abstrações para realizar o sincronismo entre threads
  - STM Haskell
  - Variáveis de sincronização (MVars)
  - Acesso a instruções de baixo nível (IORef + AtomicModifyIORef)
- Estas abstrações aliadas a facilidade da linguagem funcional Haskell fornecem uma excelente ferramenta para o desenvolvimento de programas paralelos

STM Haskell

- Uma biblioteca da linguagem funcional Haskell que fornece a abstração de memórias transacionais para programação paralela
- Baseada em transações de bancos de dados
- Sincronismo realizado pelo sistema transacional, evitando deadlocks

Proposta de implementação

- Estudar alguns algoritmos de árvores rubro-negras concorrentes
- Implementar as árvores rubro-negras na linguagem funcional Haskell
- Utilizar os três métodos de sincronismo (MVar, IORef + atomicModifyIORef e STM Haskell)
- Analisar os resultados e determinar qual método apresentará melhor desempenho e facilidade de implementação

Tipo de dados algébricos

- Um tipo algébrico permite definir novos tipos de dados através da especificação do formato de seus elementos
- Tipos algébricos podem ser tipos compostos formados pela combinação de outros tipos já existentes
- Definimos um tipo algébrico através da enumeração dos construtores dos elementos do tipo sendo definido

Exemplo

```
data Cor = Vermelho | Preto

data Arvore = Nodo {var:: TVar Int,
esquerda::TVar Arvore, direita:: TVar Arvore,
pai::TVar Arvore, cor:: TVar Cor } | Null
```

Conclusão

- Estado atual da implementação
- Próximos passos
  - Estudo de algoritmos concorrentes para árvores rubro negras
  - Implementação desses algoritmos utilizando as outras abstrações de sincronismo de Haskell
  - Testes e análise de resultados

Uma proposta de implementação de árvores rubro-negras em Haskell de forma concorrente

AUTOR: THAÍS S. HÜBNER – BOLSISTA PBIP/UFPEL

EMAIL: TSHUBNER@INF.UFPEL.EDU.BR

ORIENTADOR – ANDRÉ RAUBER DU BOIS