Escalonamento de transações - T2 CI218

Aluno Wagner J. Kramer Vieira

GRR20186095

Objetivo

Implementar os algoritmos

- Teste de seriabilidade por conflito
- Teste de visão equivalente

Observações

- O trabalho foi implementado em inglês porquê não gosto de misturar os idiomas durante a implementação, me limitei somente aos comentários em português e o nome do arquivo principal.
- O algoritmo de detecção de ciclos saiu inteiro da minha cabeça, visto que eu ainda não fiz grafos. Eu acredito que estaja correto, mas pode ser que exista um caso em que não funcione.
- A maioria das estruturas de dados é implementada em forma de lista encadeada ou duplamente encadeada, somente no algoritmo de visão equivalente é usado um array (agora fazendo o relatório eu acredito que poderia ser uma lista também, mas no momento o array foi mais rápido)
- Quando falo de "transação", estou me referindo a lista de operações com o mesmo identificador.

Execução

Basta compilar, basta rodar:

$make Isto gerará o arquivo **escalona**, que deve ser utilizado seguindo as especificações.

Limpeza

Para limpar a pasta, basta executar:

$make clean

Implementação

Leitura e estruturação dos dados

Ao serem lidos, os dados já são separados em escalonamentos, sendo que sempre que todas as operações de uma transação acabam (todos commitam), é encerrado o escalonamento e criado um novo para continuar a leitura.

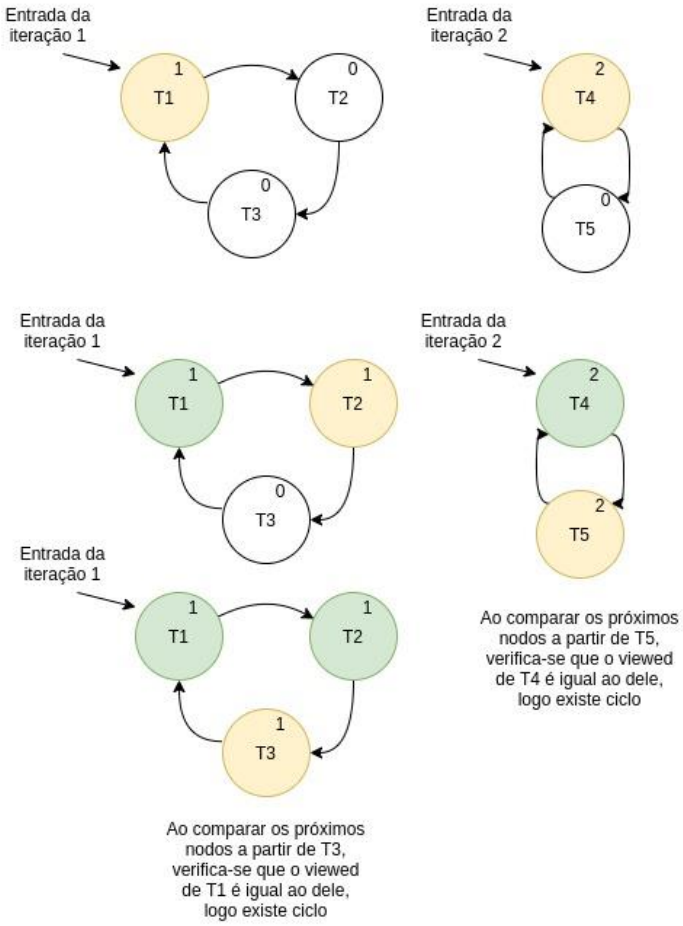
Criação de arestas

Para cada transação em *Tj*, é verificado se a última operação em *Ti, antes* de *Tj* satisfaz uma das propriedades para se cria uma aresta:

- Aresta *Ti* -> *Tj* para cada *r*(*x*) em *Tj* depois de *w*(*x*) em *Ti*
- Aresta *Ti* -> *Tj* para cada *w*(*x*) em *Tj* depois de *r*(*x*) em *Ti*
- Aresta *Ti* -> *Tj* para cada *w*(*x*) em *Tj* depois de *w*(*x*) em *Ti*

Detecção de ciclos

- Todas as transações começam com a propriedade viewed sendo 0
- Cria-se uma variável que representa um iterador, iniciando em 1
- Itera-se sobre todas as transações do escalonamento, sempre que é encontrado uma transação com a propriedade viewed igual a zero, adiciona-se 1 no iterador, seta-se o valor da aresta como o valor atual do iterador e segue-se todas as arestas da transação, setando o viewed igual ao valor do iterador sempre que se troca de novo, quando a aresta conecta uma transação a outra com o mesmo valor na propriedade viewed, significa que aquela transação já foi vista, logo se tem um ciclo.
- O escalonamento é serial se não existe ciclo no grafo.



Na imagem, o nodo amarelo é o que está atualmente sendo avaliado, o verde é o que já foi avaliado e o preto e branco nunca foi visto naquela iteração.

Visão equivalente

- Detecta-se qual a última operação feita no escalonamento
- Remove-se todos os commits, pois não são mais necessários
- Duas informações principais são passadas recursivamente para a função que executa o algoritmo de visão equivalente
    - Quais são as transações sendo utilizadas na permutação
    - Quais são as propriedades que já foram lidas
- Para cada transação, é verificado se já está sendo utilizado na permutação, caso não esteja, é verificado se está fazendo uma escrita em um valor que já foi lido nas outras permutações (compara-se cada escrita da transação com as leituras feitas anteriormente), bem como adiciona as propriedades que são lidas naquela transação a lista de propriedades lidas, que é passada recursivamente para a próxima execução da função.
- Caso todas as transações estejam sendo utilizadas na permutação, então é verificado se a última transação da permutação (ou visão) é escrita, caso seja uma leitura e a última operação do escalonamento seja uma escrita, retorna-se falso, caso contrario, verdadeiro.
- Sempre que é encontrado uma escrita de valor após o mesmo valor ter sido lido, retorna-se falso.
- Se o algoritmo retorna verdadeiro, existe visão equivalente, caso contrário, não.