Programação Concorrente – Aula 5

Gilmar Gimenes Rodrigues

1 Ações Atômicas "Compostas" ou Coarse-Grained

Uma ação atômica coarse-grained (de granularidade grossa) é uma sequência de ações atômicas fine-grained executada de modo indivisível.

Notação para ação atômica coarse-grained:

```
<comando_1; comando_2; ...; comando_n;>
```

Esta é uma ação atômica incondicional (não começa com `await`). A sequência de comandos tem que ter a propriedade da terminação (com certeza sua execução sempre termina), como uma seqüência de atribuições. Nenhum dos comandos poderia ser uma chamada a uma função que entra em loop e nunca retorna.
Exemplos:

```
<x = 0; y = 0;>

<newElement.next = head; head = newElement;>
```

Sincronização baseada numa condição:

```
<await condição; comando_1; comando_2; ...; comando_n;>
```

Esta é uma ação atômica condicional (começa com `await`). Como anteriormente, a sequência de comandos tem que ter a propriedade da terminação. Exemplos:

- Ação atômica com sincronização:

  ```
  <await n > 0; n = n - 1;>
  ```

  Depois do ";" a condição é verdadeira com certeza.

- Só atomicidade:

  ```
  <x = x + 1; y = y + 1;>
  ```

- Só sincronização:

  ```
  <await x > y;>
  ```

Numa ação incondicional, se houver só um comando dentro e se ele for A.M.O. (que satisfaz a propriedade "at most once"), então podemos tirar o "<" e ">". Exemplos:

```
<x = 7;>              equivale a          x = 7;

<await x > 0;>        equivale a          while (x > 0)
                                              ;

<await x > y>       não equivale a        while (x > y)
                                              ;
```

A implementação de ações coarse-grained condicionais (com `await`) na sua forma geral é cara. Mas casos especiais, como a operação P num semáforo, podem ser implementados eficientemente.

2 Políticas de Escalonamento e Justiça (Imparcialidade).

2.1 Políticas de Escalonamento

Num monoprocessador:

- Escalonamento "run-to-completion" (não preemptivo) : threads rodam até o fim, ou até aguardar algum evento.
- Escalonamento preemptivo com timeslicing e round-robin: cada thread tem uma fatia de tempo para ser executada e quando esse tempo acaba, entra outra thread.

Num multiprocessador:

- Execução paralela.

Ação elegível: é a próxima ação a ser executada por alguma thread. Se você tem várias threads então você tem várias ações elegíveis a cada instante. A política de escalonamento determina a escolha de uma ação entre as várias elegíveis.

Como ações atômicas só podem ser executadas em paralelo (num multiprocessador, como uma máquina SMP) se não interferirem uma com a outra, pode-se modelar a execução paralela como execução serial entrelaçada das ações atômicas.

2.2 Justiça Incondicional (Unconditional Fairness)

Toda ação incondicional que for elegível num certo instante será com certeza executada em algum instante futuro.

Exemplo:

```
boolean continue = true;

thread_1() {                          thread_2() {
    while (continue)                      continue = false;
        ;                             }
}
```

O comportamento desse programa muda de acordo com a política de escalonamento.

A política "run-to-completion" não é incondicionalmente justa. Se a thread 1 ganhar a CPU primeiro, ela nunca dará chance à thread 2. Teremos então uma ação elegível (a atribuição `continue = false`) que nunca será executada. Por outro lado, se a thread 2 ganhar a CPU primeiro, ambas as threads terminarão.

A política time-slicing tem justiça incondicional e a execução paralela também.

2.3 Justiça Fraca (Weak Fairness)

Uma política de escalonamento é fracamente justa se ela for incondicionalmente justa e se toda ação atômica condicional elegível cuja condição se torna verdadeira e permanece verdadeira daí para frente é com certeza executada em algum instante futuro.

2.4 Justiça Forte (Strong Fairness)

Uma politica de escalonamento é fortemente justa se ela for incondicionalmente justa e se toda ação atômica condicional elegível cuja condição se torna veradeira e volta a ser verdadeira repetidamente (infinitas vezes) é com certeza executada em algum instante futuro.

Exemplo:

```
boolean continue = true;
boolean tenta = false;

thread_1() {
    while(continue) {
        tenta = true;
        tenta = false;
    }
}

thread_2() {
    <await tenta; continue = false;>
}
```

Que nível de justiça é necessário para garantir que esse programa acabe?

- Com justiça forte: o programa acaba.
- Com justiça fraca: não sabemos se o programa acaba ou não (justiça fraca não diz nada neste caso).