Теория и практика многопоточного программирования

Лекция 7

Неганов Алексей

Московский физико-технический институт (национальный исследовательский университет)
Кафедра теоретической и прикладной информатики

Москва 2020

Соисполняющиеся объекты

```
1  template <class T>
2  class waitFreeQueue {
3      atomic_int head, tail;
4      int len;
5      T *items;
6
7  public:
8      waitFreeQueue(int cap): head(0), tail(0), len(cap) {
9          items = new T[cap];
10     }
11     void enq(T x) {
12         items[tail % len] = x;
13         tail++;
14     }
15     T deq() {
16         if (tail == head)
17             throw;
18         T x = items[head % len];
19         head++;
20         return x;
21     }
22 };
```

Соисполняющиеся объекты

- **Вызов метода** — интервал между событиями `invocation` и `response`
- **Precondition** — состояние объекта до события `invocation`
- **Postcondition** — состояние объекта после события `response`
- **Side effect** — изменение состояния объекта при вызове метода

Quiescent consistency

Определение 1

Объект называется **покоящимся** *(quiescent)*, если в данный момент времени нет ожидающих вызовов методов объекта.

Принцип 1

Вызовы методов должны происходить «по одному за раз» и последовательно.

Принцип 2

Вызовы методов, разделённые периодами покоя, должны давать результат в том порядке, в котором методы вызываются **в реальном времени**.

Определение 2

Вызовы методов, удовлетворяющие одновременно принципам 1 и 2, называются **согласованными по периодам покоя** *(quiescently consistent)*.

Quiescent consistency

Определение 3

Метод называется **тотальным** *(total),* если он определён для каждого состояния объекта, и **частичным** *(partial)* в противном случае.

Теорема 1 (о неблокируемости QC)

Для любого соисполнения любых незавершённых вызовов тотальных методов существуют события завершения вызова (`response`), согласованные по периодам покоя.

Это значит, что если вызов начался для тотального метода, он не должен дожидаться завершения любого другого незаконченного вызова.

Определение 4

Свойство корректности *P* называется **композиционным** для системы объектов, если из того, что каждый объект системы обладает *P*, следует, что и система в целом обладает *P*.

Теорема 2

Свойство согласованности по периодам покоя является композиционным.

Sequential consistency

Принцип 3

Результаты вызовов методов должны получаться в том порядке, который **определён программой**.

Определение 5

Вызовы методов, удовлетворяющие одновременно принципам 1 и 3, называются **упорядоченно согласованными** *(sequentially consistent)*.

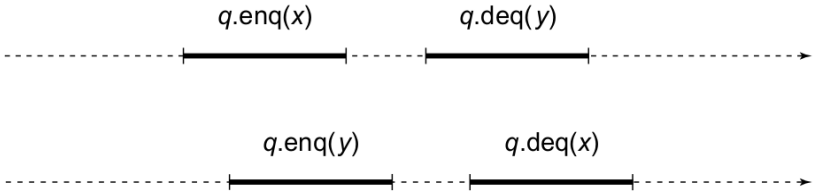
Замечание

Из упорядоченной согласованности не следует согласованность по периодам покоя, как и наоборот.

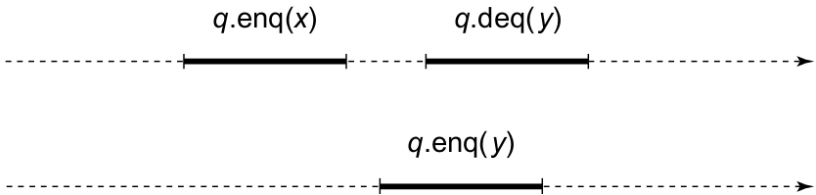
Теорема 3

Свойство упорядоченной согласованности НЕ является композиционным.

Sequential consistency



Назовите упорядоченно согласованный порядок исполнения вызовов. Является ли он единственным?



Можно ли назвать такое исполнение упорядоченно согласованным?

Принцип 4

Изменения данных, как результат применения метода, имеют место непосредственно между событиями его вызова и завершения.

Определение 6

Вызовы методов, удовлетворяющие одновременно принципам 1 и 4, называются **линеаризуемыми** *(linearizable)*.

Замечание

Из линеаризуемости очевидно следует упорядоченная согласованность, обратное неверно.

Определение 7

Точка линеаризации — то место, где изменения данных, явившиеся следствиям изменения метода, имеют эффект.

Как будет показано далее, свойство линеаризуемости является неблокирующим и композиционным.

Понятие «истории»

Определение 8

История *H* представляет собой конечную последовательность событий типа `invocation` и `response` для методов. Подпоследовательность *S* событий из *H* называется **подысторией**.

Определение 9

Будем называть событие `response` **соответствующим** событию `invocation`, если они относятся к тому же объекту и потоку исполнения. **Вызов метода** в истории *H* есть пара событий `invocation` и соответствующее ему `response`; если же такого события не существует, то вызов метода называется **ожидающим** *(pending)*.

Определение 10

Расширением истории *H* будем называть историю H', содержащую *H* и 0 или более событий типа `response` для ожидающих вызовов.

Определение 11

Завершённой историей для истории *H* будем называть историю $\mathrm{complete}(H)$, содержащую подпоследовательность пар всех соответствующих событий *H*, без ожидающих вызовов.

Определение 12

Историю H будем называть **упорядоченной** *(serialized)*, если первое событие H есть `invocation`, и каждое событие `invocation`, возможно, за исключением последнего, имеет непосредственно следующее за ним событие `response`.

Начало вызова метода m объекта x с последовательностью аргументов a^* в потоке A обозначим $\langle x.m(a^*)A\rangle$. Аналогично, $\langle x\colon t(r^*)A\rangle$ представляет собой событие конца вызова, где t — обозначение для успешного завершения метода или имя ошибки (исключения), а r^* — последовательность выходных значений.
Подистория потока $H|A$ есть подпоследовательность всех событий H, в которых поток есть A. Аналогично определяется подистория объекта $H|x$. Историю H такую, что для каждого потока A подистория $H|A$ упорядочена, называют **правильной** *(well formed)*.
Обозначим факт предшествования события `response` метода m_0 событию `invocation` метода m_1 как $m_0 \to_H m_1$. Для подистории $H|x$ будем писать $m_0 \to_x m_1$.

Определение 13

Упорядоченную историю H будем называть **легальной** *(legal)*, если для каждого объекта его подистория легальна для объекта.

Линеаризация истории

Определение 14

История H является **линеаризуемой**, если для её расширения H' существует легальная упорядоченная история S такая, что:

$$\begin{aligned} &\mathrm{complete}(H) = S; \\ &\text{если } m_0 \to_H m_1, \text{ то } m_0 \to_S m_1. \end{aligned} \tag{1}$$

Теорема 4 (композиционность линеаризуемости)

Для того, чтобы H была линеаризуемой, необходимо и достаточно, чтобы для каждого объекта x подыстория $H|x$ была линеаризуемой.

Доказательство достаточности: индукция по количеству вызовов в H'.

Теорема 5 (неблокируемость линеаризации)

Если $\langle x \ \mathrm{inv} \ P \rangle$ есть ожидающий вызов тотального метода в линеаризуемой истории H, то существует событие конца вызова $\langle x \ \mathrm{res} \ P \rangle$, такое, что $H\langle \mathrm{res} \ P \rangle$ — линеаризуемая история.

Упражнения

Пусть r — операция чтения из регистра чтения-записи, а w — операция записи.

- Пусть $H|A = \langle x.w(1)A\rangle\langle x.r(1)A\rangle$, $H|B = x.w(2)B$, причём записи соисполняемы, а чтение и запись – нет. Является ли исполнение линеаризуемым?
- Является ли исполнение с историей $H = \langle x.w(1)A\rangle\langle x.r(2)B\rangle$ линеаризуемым, если операции разделены периодом покоя?
- Пусть $H|A = \langle x.w(0)A\rangle\langle x.r(1)A\rangle$, $H|B = \langle x.w(1)B\rangle$. Является ли исполнение упорядоченно согласованным?
- Пусть $H|A = \langle x.w(1)A\rangle\langle x.r(3)A\rangle\langle x.w(3)A\rangle$, причём между первой и второй операцией имеется период покоя; $H|B = \langle x.w(2)B\rangle$, причём эта операция соисполняема с последними двумя операциями потока A. Будет ли такое исполнение согласованным по периодам покоя?