ТЕОРИЯ И ПРАКТИКА МНОГОПОТОЧНОГО ПРОГРАММИРОВАНИЯ

Тема 11

Свойства примитивов по отношению к числам консенсуса

Д.ф.-м.н., профессор А.Г. Тормасов
Базовая кафедра «Теоретическая и Прикладная Информатика», МФТИ

Тема

- Теорема об эквивалентности примитивов с одинаковым числом консенсуса.
- Теорема о числе консенсуса атомарных операций чтения и записи и невозможности синхронизации при использовании только их.
- «тонкие отличия» числа консенсуса и решения задач синхронизации
- Теорема о неулучшаемости числа консенсуса путем комбинирования.

Универсальность консенсуса

- Обсудили что нельзя сделать из других примитивов
- А что можно? А что такое одинаковость числа консенсуса?

Теорема.

Любой объект с числом консенсуса n в системе из n потоков можно реализовать, используя

- один или более экземпляров любого другого объекта с тем же числом консенсуса n
- несколько экземпляров атомарных регистров типа чтение-запись (опционально)

Универсальность консенсуса

- **произвольный примитив** синхронизации можно реализовать через **любой другой примитив** синхронизации одинаковой с ним силы!

Доказательство

создадим алгоритм:

- реализует объект (выбранного примитива)
- реализация является корректной
- реализация является ограниченной от ожидания

Конструируемый объект

- последовательный объект в его начальном состоянии
- «журнал операций» объекта
  - последовательность вызовов в виде односвязного списка, каждый элемент описывает вызов метода объекта с параметрами
- Read-only
- Исполнение в потоке
  - Добавляет описание метода который надо исполнить в голову
  - Создает приватную копию данных и исполняет все методы из списка
  - Возвращает результат исполнения последней операции

Конструируемый объект

- Сделаем сконструированный объект соисполняемым для вызова любого потока
  - Создадим элемент списка для выполнения метода
  - Решим задачу о консенсусе n потоков (по условию всегда можно) используя ЛЮБОЙ примитив с числом n (!)
  - Победитель делает свою копию и добавляет элемент в вершину и вычисляет ответ
  - Это соисполнимо, так как все операции – только чтение, модификаций ни объектов ни списков нет
- Полученное, по конструкции, свободно от ожидания

Число консенсуса регистров

Теорема.

Используя только атомарные операции чтения и записи, невозможно решить задачу консенсуса для двух и более потоков

- Число консенсуса атомарных чтения/записи: 1
- ТОЛЬКО детерминированные объекты и системы!

Доказательство

от «противного» - пусть существуют 2 процесса, решающих задачу консенсуса для 2 потоков 0 и 1

Доказательство

- По лемме будет существовать критическое состояние. Выберем нумерацию потоков так, что после критического
  - все ходы потока 0 ведут в 0-валентное состояние
  - все ходы потока 1 ведут в 1-валентное состояние

- Рассмотрим все возможные варианты
  - один поток читает (а второй делает что угодно)
    - Поток 0 собирается читать нашу ячейку
    - Поток 1 делает что угодно (читает или пишет)
  - оба потока пишут в разные регистры (r0 и r1 соответственно)
  - оба потока пишут в один и тот же регистр r

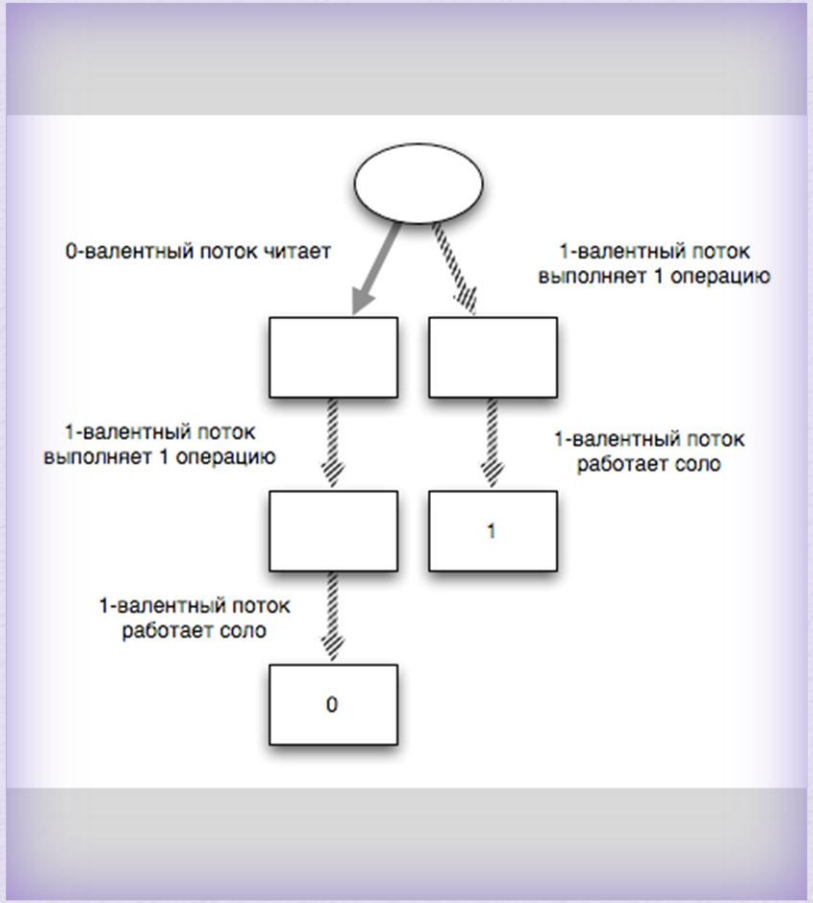
Поток 0 читает

(Поток 1 может читать или писать в тот же или другой регистр)

Левая ветка
поток 1 делает 1 ход, переводя в 1-валентное состояние, и работает соло, оканчивая решением 1.

Правая ветка
поток 0 делает ход, переводя в 0-валентное состояние, и оканчивает решением 0. поток 1 также, после первого хода, работает соло и должен решить что 0.

Но для потока 1 оба состояния неотличимы – при одинаковых условиях он обязан получить два разных решения!



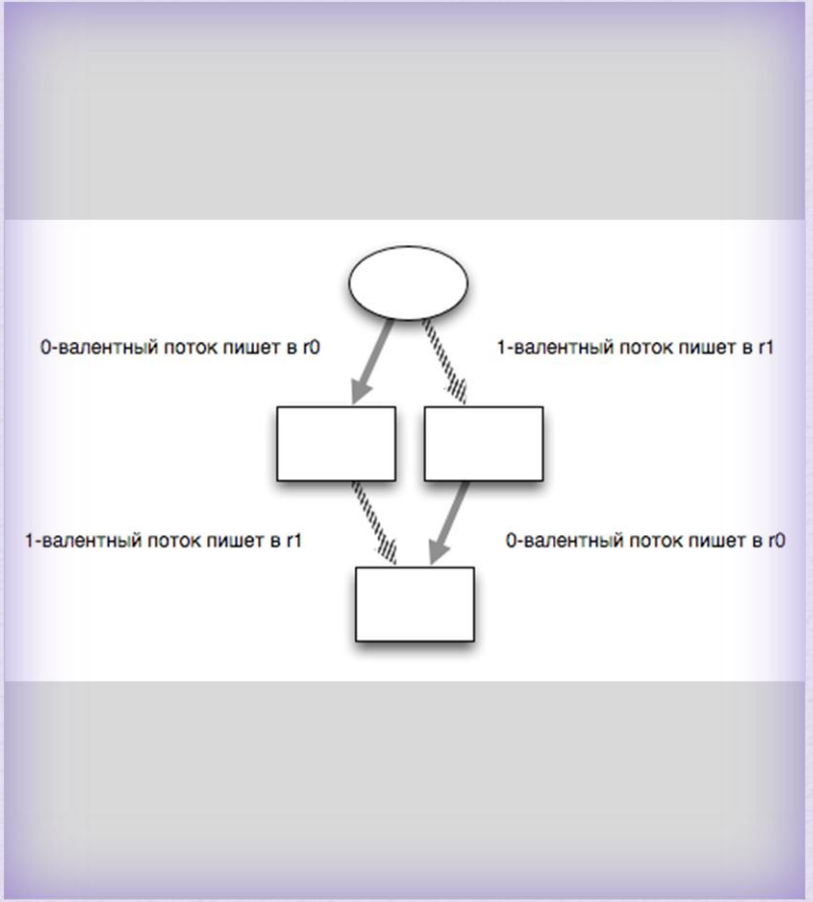
Потоки пишут в разные регистры

(Поток 0 пишет в r0 и поток 1 пишет в r1)

Левая ветка
поток 0 сначала пишет в r0, переводя в 0-валентное состояние, и поток 1 пишет в r1, оканчивая решением 0.

Правая ветка
поток 1 сначала пишет в r1, переводя в 1-валентное состояние, и поток 0 пишет в r0, оканчивая решением 1.

Но для потоков оба состояния неотличимы, так как они не знают кто сделал первый ход – при одинаковых условиях опять обязаны получиться два разных решения!



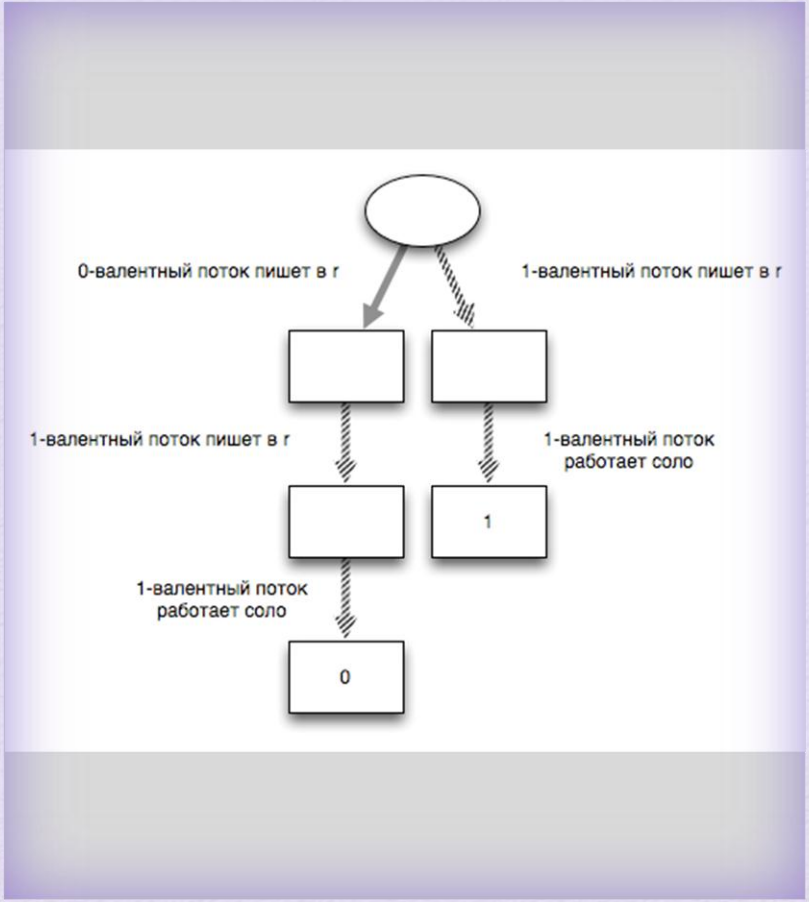
Потоки пишут в один регистр

(Оба потока пишут в регистр r)

Левая ветка
поток 0 сначала пишет в r, переводя в 0-валентное состояние, и затем поток 1 пишет в r, оканчивая решением 0.

Правая ветка
поток 1 сначала пишет в r, переводя в 1-валентное состояние, и затем поток работает соло, оканчивая решением 1.

Но для потока 1 оба состояния неотличимы, так как содержание регистра r затерто после записи и он не знает, делал ли поток 0 первый ход – при разных начальных условиях обязаны получиться одинаковые решения!



Число консенсуса регистров

Следствие.

Построение любого свободного от ожидания объекта с числом консенсуса более 1, которое использует только операции атомарного чтения и записи, **невозможно**.

- Комбинируя операции чтения и записи, построить, например, FIFO очередь, нельзя

Неулучшаемость консенсуса

Теорема.

Если X имеет число консенсуса n, и Y имеет число консенсуса m, и $m < n$, то не существует свободной от блокировок реализации X через Y в системе с более чем m потоками.

Доказательство

Пусть X и Y являются объектами в традиционном ООП смысле (имеют методы, и данные в виде регистров, которыми пользуются методы)

Неулучшаемость консенсуса

Мы имеем

- набор вызовов для объекта X {Fx}
- набор вызовов для объекта Y {Fy}
- причем X может быть реализовано через Y
- есть некоторая композиция вызовов {Fx Fy}, которая реализует протокол, и использует данные объектов Y!

- Итого, у нас есть новый объект, в каком то смысле эквивалентный Y - та же память, тоже свободный от ожидания, с другим набором вызовов, и решает задачу консенсуса для n>m! Но Y по определению ее решить не может. Противоречие.

Консенсус – панацея?

- Является ли решение задачи консенсуса синонимом возможности решения задач организации корректного синхронного доступа к данным, и наоборот?
- Задача о соглашении для k-набора продемонстрировала, что число консенсуса не полностью характеризует вычислительную мощность объекта.
- Существует понятие «надежности» иерархии, согласно которому нельзя сконструировать объект из более высокого уровня иерархии через низкоуровневые (См теорему о невозможности). Иерархия консенсуса не является надежной для недетерминистических (асинхронных) объектов.
- Более того, есть задачи и их решения, не попадающие под определение консенсуса (в основном, изза несоответствия требованию свободы от ожидания), но решающие практические задачи.

Алгоритм булочной

```
// Пример реализации алгоритма Лампорта
// использует лексикографическое сравнение пар
// (num[i],i) << (num[j],j) тогда и только тогда
// num[i]<num[j] или num[i] == num[j] и i<j

// данные
boolean choosing[n];
int num[n];

// инициализация
for (j=0; j < n; j++)
{
    num[j] = 0;
}

// выход для процесса i
void Leave(int i)
{
    // сбрасываем билет
    num[i] = 0;
}
```

```
// вход для процесса i
void Entry(int i)
{
    // выбор билета – "тамбур"
    choosing[i] = TRUE;
    num[i] = max(num[0], ..., num[n-1]) + 1;
    choosing[i] = FALSE; //конец тамбура
    // для всех остальных процессов
    for (j=0; j < n; j++)
    {
        // ждем если процесс выбран
        while (choosing[j])
            ;
        // ждем если билет процесса раньше нас
        if ((num[j] > 0) &&
          ((num[j] < num[i]) ||
          (num[j] == num[i]) && (j < i)))
        {
            while (num[j] > 0)
                ;
        }
    }
}
```

Алгоритм булочной

Лемма

Алгоритм булочной является свободным от взаимной блокировки

Доказательство: ожидающий поток имеет уникальную пару значений (num[i],i), то есть не ждет другого потока

Лемма

Алгоритм булочной является «честным» (обслуживает в порядке прихода)

Доказательство: если «тамбур» (см текст) i предшествует j, то билет i меньше, так как

$write_i(num[i]) \rightarrow read_j(num[i]) \rightarrow write_j(num[j]) \rightarrow read_j(choosing[i])$

то есть j заблокировано пока choosing[i] равно TRUE.

NB Любой алгоритм, являющийся свободным от взаимной блокировки и являющийся «честным», также является свободным от зависаний.

Лемма

Алгоритм булочной удовлетворяет условиям взаимного исключения в критической секции.

Доказательство (самостоятельно).

Алгоритм булочной

- Прост, элегантен, обладает честностью, обладает свободой от взаимной блокировки, обладает уникальным свойством работать корректно даже с «небезопасными» регистрами при параллельном чтении-записи (!)
- Но – требует n раздельных ячеек, где n – максимальное значение соисполнимых потоков!
- Можно доказать, что не существует другого алгоритма использующего только чтение-запись, со свободой от взаимной блокировки, требующего меньшего размера памяти!
- Но, как обсуждалось, число консенсуса чтения-записи 1, так как же быть с алгоритмом булочной Лампорта??? (для самостоятельного обдумывания)

Выводы

- Рассмотрели отношения примитивов друг к другу при помощи анализа их на число консенсуса
- Все примитивы одного уровня одинаковы, их не улучшить!
- Доказали число консенсуса для чтения-записи – основополагающее утверждение для всей теории
- Начали анализировать «тонкие отличия» числа консенсуса и решения задач синхронизации в общем
  - Невозможность решения задачи о консенсусе не есть невозможность организации синхронизации
  - Практические ограничения – не только сложность решения задачи, но и объем памяти, нужной для работы алгоритма
  - Часто легче сделать менее «свободный» но более эффективный алгоритм

(c) А. Тормасов, 2010-11 г.

Базовая кафедра «Теоретическая и Прикладная Информатика» ФУПМ МФТИ
tor@ crec .mipt .ru_