
INNOPOLIS UNIVERSITY

МФТИ
Теоретическая и Прикладная Информатика

INNOPOLIS UNIVERSITY

Математика параллельного программирования на разделяемой памяти: Немного о математике XXI века

Заведующий базовой кафедрой ФУПМ МФТИ
Теоретическая и Прикладная Информатика

Ректор Университета Иннополис

д.ф.-м.н., проф. А.Тормасов

2013

О чем эта лекция

- Преставление автора о математике в XXI веке и ее практическом применении
- Взгляд «со стороны» практика
- Пример нового подхода
  - новая проблема
  - новый формализм
  - новый метод доказательства
  - новая интерпретация результатов
  - противоречие в результате

О докладчике

- Александр Геннадьевич Тормасов
- Выпускник МФТИ
- Работаю в Физтехе с момента окончания обучения (25+ лет)
- Ректор нового Университета Иннополис
- Защитил в МФТИ кандидатскую (мат моделирование) и докторскую (computer science) диссертации, профессор, заведующий кафедрой прикладной и теоретический информатики
- Имею опыт контрактного программирования с западными компаниями с 1990 года
- С 1999 года сотрудничаю с SWsoft/Parallels, затем Acronis, Runa Capital и др., сотрудник No 2 компании SWsoft/Parallels
- Автор более 120 патентов США и Евразии
- Руководил 18 защищенных кандидатских
- Не в коем случае не «чистый математик»!

- *Не претендую на «правильность- 3 » - личное мнение*

Коротко

- Математика в XXI веке изменилась – «закопалась»
- Большинство направлений настолько сложны и глубоки, что их понимают 100-200 чел по всему миру
  - Времена «универсалов» - увы, но, похоже, в прошлом
  - Многие математические утверждения даже проверить некому
    - NP vs P доказательством сотрудником HP
    - статьи Перельмана

Новые подходы

- Доказательство с помощью компьютера – средства XX и XXI века
  - Ограниченный перебор – вводим какие-то классы и их перебираем при помощи компьютера
    - Задача 4 красок
- Даже программы и то доказываются на правильность
  - Например, короткие lock-free алгоритмы

Критика математики на Физтехе

- Идея образования Физтеха - энциклопедичность
  - Могу не запомнить всех деталей и доказательств, но знаю что «такое есть» - могу восстановить то, что потребуется, поиском
  - «Когда-то это умел делать руками, но забыл» – все равно могу восстановить
- Но сейчас…
  - Восстанавливать нечего, так как «не слышал»

Критика математики на Физтехе

- Увы, сейчас даже обзорно получить «все нужное» не получается…
- Преподаваемая математика ограничена практически XIX веком
  - От XX века считаные главы, в основном дискретная математика и выч маты
  - Море зубодробительных доказываемых теорем
  - Мало показывается новых подходов
- Что делать?
  - Не знаю. Просто констатирую

Пример нового в математике

- Задача о консенсусе и его невозможности
  - Часть базового цикла кафедры – программирование на разделяемой памяти
- Появилась в самом конце XX века как ответ на актуальную и практическую задачу
- Потребовала (заимствовала) новый формализм, методы доказательства и идеи
- Сначала все шло в русле «теоремы существования» – но потом…
- Вывод оказался несколько парадоксален

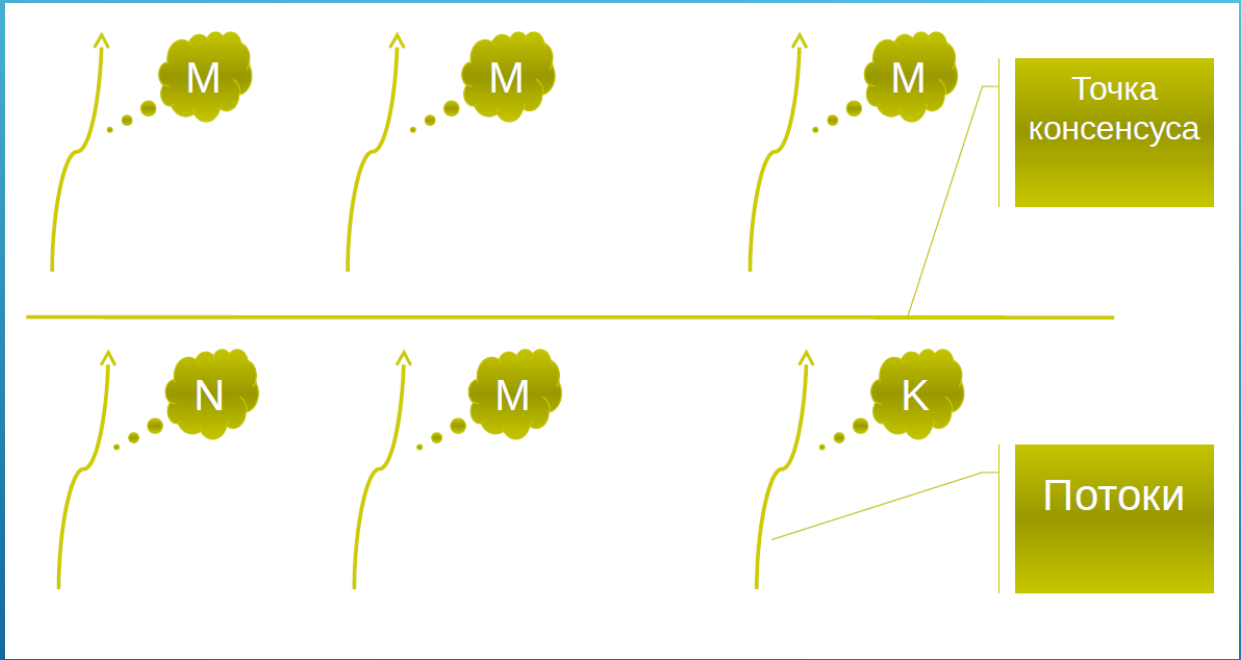
Как...

- Как можно сравнить разные примитивы синхронизации между собой?
- Какие нужно реализовать аппаратные примитивы для нового микропроцессора?
- Можно ли реализовать один примитив из другого?
- Для решения этих вопросов необходимо создание какой-то формы или механизма сравнения, лучше всего в виде формализации, пригодной затем для создания и доказательства каких либо утверждений.

Консенсус

- Задача о консенсусе – это некоторая абстрактная задача, которая признается многими исследователями как удобный способ оценки синхронизационных проблем.

Задача о консенсусе



Задача о консенсусе...

- Пусть есть n потоков
- Все работают с одной разделяемой областью
- Каждый поток пытается поместить туда значение vi
- По достижению консенсуса:
  - существует значение j, когда все участвующие (без исключения) потоки считают, что значение разделяемой области есть vj.
  - Значение j, возможно, не единственное

Задача о консенсусе...

- Объект консенсуса предоставляет 1 метод
  - value decide(value in);
- Каждый поток вызывает его со своим значением
  - decide(vi);
- Метод ВО ВСЕХ потоках возвращает всегда одно и то же значение, с которым поток идет далее

Задача о консенсусе...

- Решение непротиворечиво
  - Метод возвращает одно значение, одинаковое для всех потоков
- Решение корректно
  - Предложено всегда одним из потоков
- Завершенность
  - Каждый корректный процесс когда-нибудь принимает решение

Определение

- Определение

“Протокол консенсуса” есть реализация объекта консенсуса, свободная от ожидания.

- Далее ограничимся детерминированными объектами с последовательной спецификацией
  - Каждый последовательный вызов метода имеет единственный результат

Объект консенсуса

Определение
Класс C решает проблему **консенсуса** для n потоков, если существует протокол консенсуса, использующий для принятия решения любое количество объектов класса C и любое количество атомарных регистров

Определение
Числом консенсуса класса C является максимальное количество потоков n, для которых этот класс решает проблему консенсуса. Число консенсуса может быть бесконечно большим.

Следствие
Если существует реализация объекта C из одного или более объектов класса D и атомарных регистров, то объект D имеет число консенсуса, как минимум, равное числу консенсуса объекта C.

Консенсус – применение

- «Похожа» на то, что решается при обеспечении синхронизации
  - Пронумеруем потоки перед заходом в критическую секцию
  - Работает в ней только тот, чье решение принято
  - Остальные – ждут!
  - Не все потоки обязаны посетить точку консенсуса

Типичная задача

- Бинарный консенсус
  - Два потока предлагают значения 0 или 1
- Каждый поток предпринимает какие то действия (ходы), которые в конечном итоге приводят к решению.
  - Ход есть вызов какого либо метода разделяемого объекта
- Состояние протокола состоит из состояний всех потоков и разделяемых объектов.
- Начальное состояние есть состояние протокола перед ходами всех участвующих потоков.
- Конечное состояние – состояние протокола после того как все ходы всех потоков закончились.
- Решением назовем значение всех потоков в финальном состоянии.

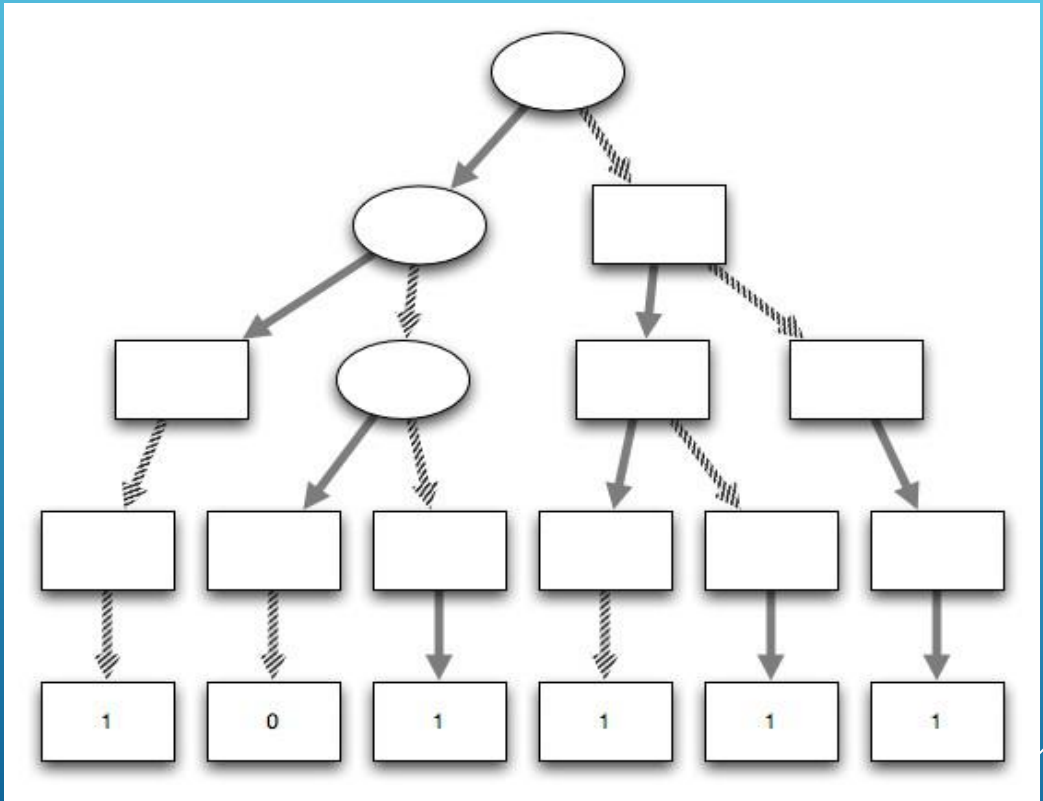
Типичная задача...

- Дерево состояний
  - Узел (вершина) – состояние
  - Ребро – ход из одного состояния в другое
  - Лист (решение) помечается принятым значением
- Для бинарного дерева
  - 2 ребра для каждого узла
  - Листья помечаются только 0 или 1
  - Ребро соединяет предшествующие узлы (и состояния)

Валентность

- Возможность попасть в какое то состояние
- Состояние является бивалентным, если двигаясь из него, можно попасть в листы, помеченные разными значениями
- Состояние является унивалентным, если все листы помечены одним значением
  - 1-валентное
  - 0-валентное
- Пунктирные и сплошные линии на рисунке – ходы разных потоков

Валентность...



Число консенсуса регистров

Теорема.

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

Следствие:

– Число консенсуса атомарных чтения/записи: 1

Рассматриваются ТОЛЬКО детерминированные объекты и системы!

Доказательство

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

Новые идеи

- Для доказательства используется факт наличия знания
- Появляется «локальное время» и вообще время как основа математической модели!
  - Один поток не знает о другом – значит, они не могут принять «общее решение»
  - Мы можем умозрительно «переставить» последовательность действий разных потоков по абсолютному времени
- Более того, по сути компьютера событие не существует, пока оно не наблюдается – нет понятия «на самом деле»

  Аналог теоремы Белла - значение, ассоциируемое нами со значением ячейки, начинает существовать только после выполнения соответствующей команды считывания ячейки, а до того оно как бы «размазано» между разными кэшами, шиной данных, собственно памятью

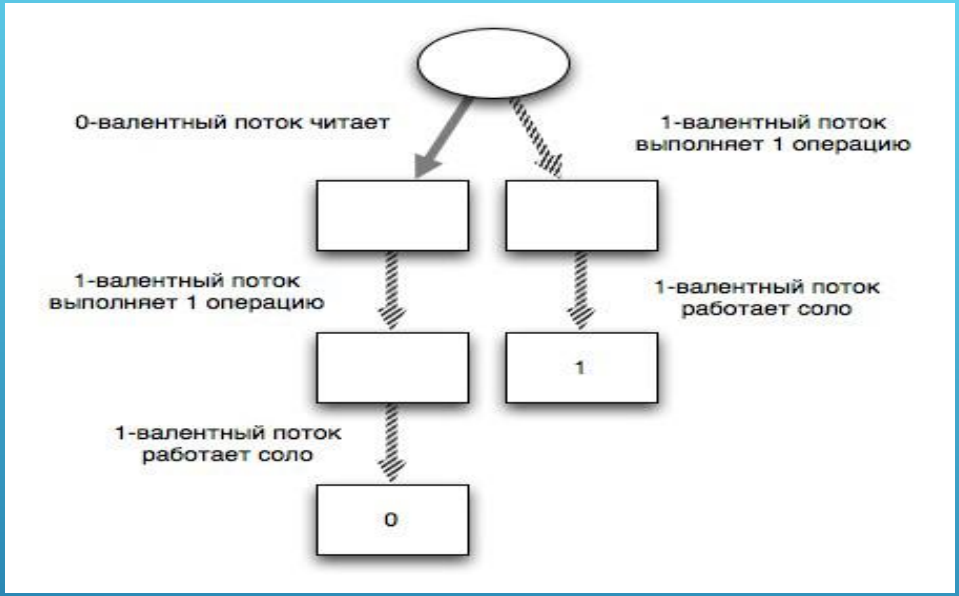
Доказательство

- По лемме будет существовать критическое состояние (после которого состояния унивалентны).

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

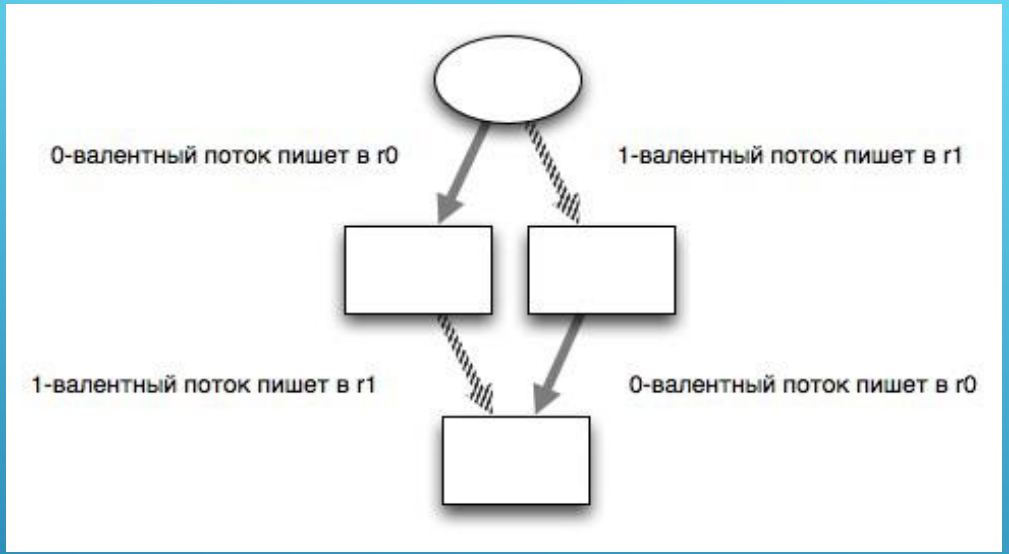


Поток 0 читает

(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!



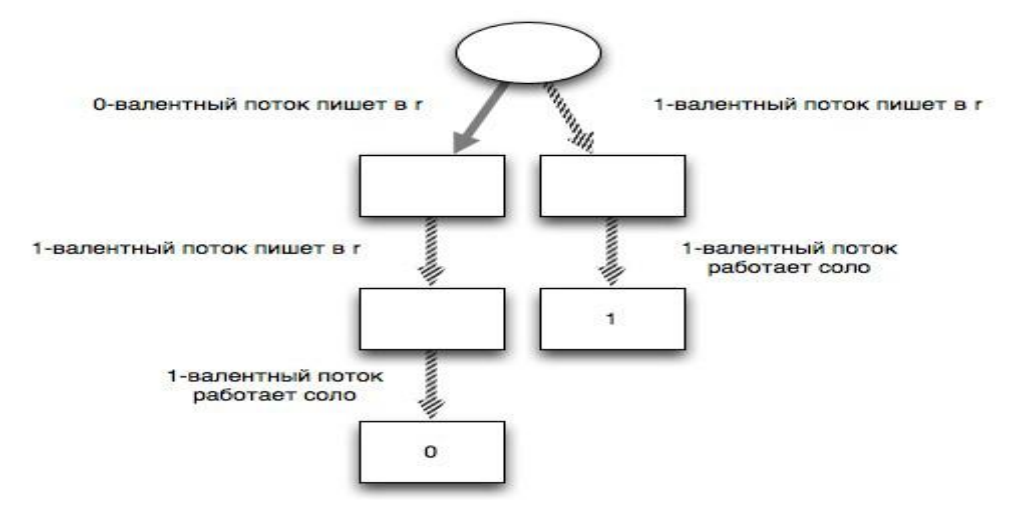
Потоки пишут в разные регистры

(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.

Соглашение для k-набора

- Пусть есть n потоков
- Все работают с одной разделяемой областью
- Каждый поток пытается поместить туда значение vi
- По достижению консенсуса:
  - существует набор не более чем k значений {j1,…,jk}, когда все участвующие (без исключения) потоки считают, что значение разделяемой области есть vj, где j принадлежит {j1,…,jk}.
- Каждый поток принимает одно из не более чем k значений

Соглашение для k-набора

Теорема

создание свободного от ожидания алгоритма, использующего только атомарные регистры чтения-записи для любого k<n, невозможно.

Следствие

число консенсуса **не является** исчерпывающей характеристикой вычислительной мощности объекта

Так как, согласно теореме, регистрами, у которых число консенсуса 1, решить задачу для k=2 нельзя, но и решить задачу о консенсусе для 2 процессов этим объектом тоже нельзя (это легко показать –просьба проверить самостоятельно), то условное число консенсуса для объекта соглашения для k-набора лежит между 1 и 2!

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, из-за несоответствия требованию свободы от ожидания), но решающие практические задачи.

Тема

- Консенсус в системе со сбоями
- Один из самых фундаментальных и достаточно неожиданных теоретических результатов, полученных в современной науке о вычислительных системах.
- Теорема Фишера, Линч и Патерсона о невозможности консенсуса в системе со сбоями (FLP impossibility)
- Следствия теоремы

Свобода от ожидания

- Определение подразумевает «устойчивость ко сбоям»
  - Алгоритм, вне зависимости от поведения соседей, должен завершиться за конечное число шагов
  - Даже если сосед замедлился, застрял или умер
- Рассмотрим утверждение, касающееся работы таких систем «со сбоями» – в обобщенном смысле

Асинхронная система

- Полностью асинхронная система

– нет никаких предположений о последовательности операций
– рассматривается в работе Fischer, Lynch, и Paterson

Асинхронная система

- Потоки обмениваются сообщениями через буфер
  - send(p,m) – помещает сообщение m, адресованное p, в буфер
    - обычно помещается пара (p,m) – полная характеристика *события*
  - receive(p) – удаляет из буфера сообщение для p, и возвращает m
    - может вернуть предопределенное значение null – нет ничего
- Нет обязательств системы доставки сообщения по порядку их доставки
- Нет обязательств даже по их присутствию в буфере
  - может вернуться null даже если пара (p,m) присутствует в буфере

FLP impossibility

Теорема

Система соисполняемых потоков, обменивающихся сообщениями, в которой потоки могут быть неограниченно задержаны (сбоить), или сообщения могут быть переставлены, не может достигнуть консенсуса.

Идея

Если среди коммуницирующих объектов один завис на неопределенное время (по любой причине, например, из-за сбоя канала, собственного сбоя и т.д.), то у нас нет никакого основания думать, что все участвующие в консенсусе «дождутся» неработающего объекта, чтобы принять действительно правильное решение за ограниченное время.

FLP impossibility

Доказательство

оригинальное доказательство доказывает существование последовательность событий (обмена сообщениями), которое никогда не ведет к консенсусу. То есть, из некого начального бивалентного состояния можно сделать ход в другое бивалентное состояние, и всегда может существовать новое бивалентное состояние, куда мы можем перейти (и, собственно, доказывается что оно всегда существует).

Есть более конструктивное доказательство, которое показывает, КАК можно найти тот самый «плохой» путь, «ломающий» асинхронную систему (предложено в 2004 году Фольцером).

Выводы

- Рассмотрели консенсус в системе с непредсказуемыми событиями
- Доказали его невозможность в классической постановке
- Практическое использование базовых определений сложно, и часто используются всевозможные «послабления»
  - Любое изменение требований обычно выводит задачу в область, где иерархия может быть ненадежной
    - особенно, связанное с внесением асинхронности в систему

Выводы

- Есть много систем, практически использующих неблокирующие примитивы для построения реально работающих систем общего назначения
  - Анализ таких систем требует аккуратности в рассмотрении требований и условий

- Можно показать, что рассмотренная последовательность событий «редка» - т.е. можно получить консенсус в плохой системе со сколь угодно близкой к 1 вероятностью

- То есть, мы доказали невозможность консенсуса, но, «заодно», получили практическое решение «нерешаемой задачи» – в полном противоречии с «чисто математической постановкой задачи»

- Мне кажется, что это демонстрация практичности чистой математики

Гомоморфное шифрование для облачных систем

Может ли компьютер вычислять то, что он не понимает?

- Как сделать так, чтоб процессор, проводя операции, не знал (и не мог вычислить) операндов?
- Облачные системы требуют этого!
  - Одна из основных проблем – мои секретные данные в руках администратора облака, который имеет физический доступ к компьютеру и может все подсмотреть!
- Нетривиальные операции с базами данных
  - Среднее между конкурентами – как вычислить не разглашая?
- Полный набор операций требует подобного подхода
  - Целочисленные арифметические
  - Булевы
  - Плавающие
- Отчасти сводится к «булеву сложению и умножению»

МАТЕМАТИЧЕСКОЕ МОДЕЛИРОВАНИЕ АЛГОРИТМОВ, РАБОТАЮЩИХ НА РАЗДЕЛЯЕМОЙ ПАМЯТИ

Задачи

- Моделирование многопоточных алгоритмов, работающих на разделяемой памяти.

- Разработка критериев наличия состояний конкурентного доступа к памяти, приводящих к некорректной работе алгоритма.

- Разработка методики автоматизированного тестирования, определяющей наличие состояний конкурентного доступа к памяти, приводящих к некорректной работе многопоточной программы.

- Оценка сложности методики автоматизированного тестирования в различных случаях.

Анализ инструкций потоков

Классификация инструкций:

- Операции чтения. Поток считывает значение одной из общих ячеек памяти. Обозначение – R.
- Операции записи. Поток записывает значение в одну из общих ячеек памяти. Обозначение – W.
- Другие операции. Операции, не входящие ни в один из вышеописанных классов. Обозначение – X.

Обозначение O_j^i, где i – номер общей ячейки памяти, j – номер потока исполнения (например, R_1^3)

Рассмотрим две операции в разных потоках A_x^j и B_y^k

Лемма. Состояние исполнения *S* после выполнения операций может зависеть от их порядка, если j=k

- A=R, B=W
- A=W, B=R
- A=W, B=W

Такие операции – ***некоммутирующие*.**

Моделирование исполнения двух потоков

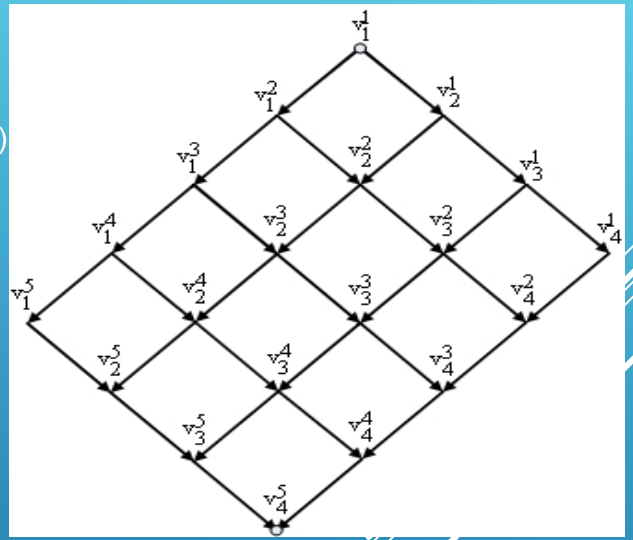
Два потока исполнения $T_1 = \{o_1^1, ..., o_k^1\}, T_2 = \{o_1^2, ..., o_n^2\}, k = z_1, n = z_2$

Функция корректности задана в виде $p = p(S^*)$, где S^* - состояние

исполнения после завершения работы потоков (результирующее состояние исполнения)

Граф *совместного исполнения потоков* – ориентированный граф

$$G:=(V,A), V = \bigcup_{\substack{i=1,k+1 \\ j=1,n+1}} v_j^i, A = \bigcup_{\substack{i=1,k \\ j=1,n+1}} (v_j^i, v_j^{i+1}) \cup \bigcup_{\substack{i=1,k+1 \\ j=1,n}} (v_j^i, v_{j+1}^i)$$



v_1^1 - *начальная* вершина, v_{n+1}^{k+1} - *конечная*. *Уровень вершины* v_j^i - i+j.

Ориентированный путь из начальной вершины в конечную - *полный путь*. Всего таких путей C_{n+k}^k , а $C_{2n}^n \approx \frac{2^{2n}}{\sqrt{n}}$

Лемма. Существует взаимно-однозначное соответствие между вариантами совместного исполнения потоков и полными путями в соответствующем графе совместного исполнения потоков.

Дуги (v_j^i, v_j^{i+1}) , где j=1,…,n+1 - *представляют* i-ую операцию, выполняемую первым потоком.

Дуги (v_j^i, v_{j+1}^i) , где i=1,…,k+1 - *представляют* j-ую операцию, выполняемую вторым потоком.

Полный путь *представляет* соответствующий ему вариант совместного исполнения потоков.

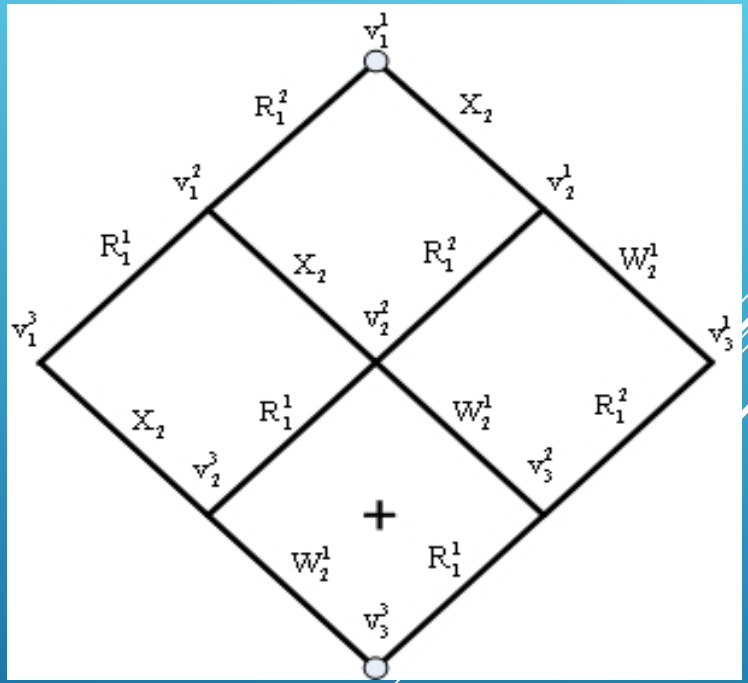
Классы эквивалентности путей на графе

Для любых двух некоммутирующих операций i и j поставим знак + в области, ограниченной дугами (v_j^i, v_j^{i+1}), $(v_j^{i+1}, v_{j+1}^{i+1})$, $(v_{j+1}^i, v_{j+1}^{i+1})$ и (v_j^i, v_{j+1}^i).

Число *некоммутирующих операций на t-ом уровне* p_t - количество вершин v_j^i уровня t, таких что операции, представляемые дугами (v_j^i, v_j^{i+1}) и (v_j^i, v_{j+1}^i) являются некоммутирующими.

Два различных пути P_1 и P_2 из v_y^x в v_{x+1}^{k+1}. Будем двигаться по дугам графа вдоль каждого из путей и преобразовывать состояние исполнения потоков *S* согласно операциям, которые эти дуги представляют. Если мы можем гарантировать, что дойдя до v_{x+1}^{k+1}, мы получим одинаковые состояния исполнения потоков *S*, то будем считать эти пути *эквивалентными*.

Утверждение. Два пути из v_y^x в v_{x+1}^{k+1}, отличающиеся только парой дуг (v_j^i, v_j^{i+1}), $(v_j^{i+1}, v_{j+1}^{i+1})$ в одном и (v_j^i, v_{j+1}^i), $(v_{j+1}^i, v_{j+1}^{i+1})$ в другом, принадлежат одному классу эквивалентности, если i-я операция первого потока и j-я операция второго коммутируют между собой.



Построение представителей классов

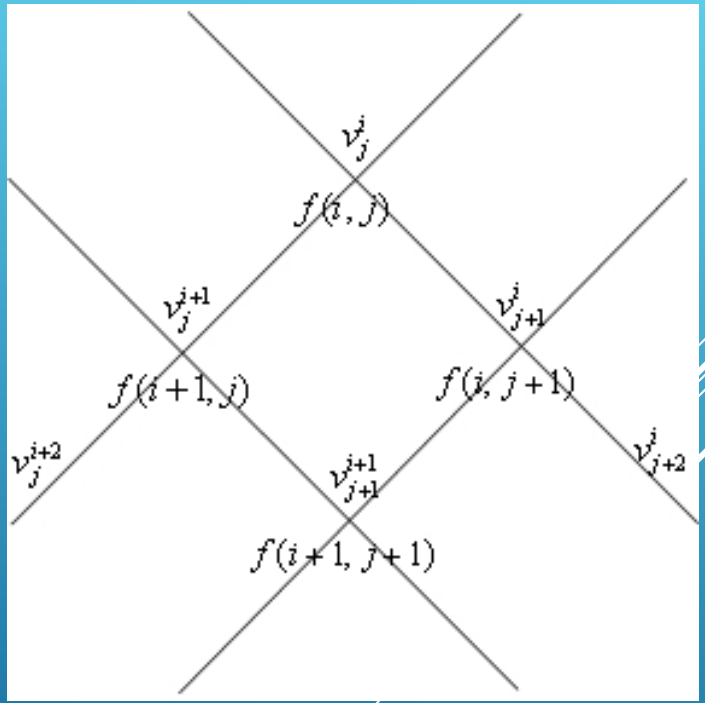
Определим функцию $f(i,j), 1 \le i \le k+1, 1 \le j \le n+1$ ..

1. $f(i,j) = 1$ при $i = k+1, j = 1,..,n+1$
2. $f(i,j) = 1$ при $j = n+1, i = 1,..,k$
3. $f(i,j) = f(i+1,j) + f(i,j+1)$, если i-я операция первого потока и j-я операция второго потока являются некоммутирующими, в противном случае $f(i,j) = f(i+1,j) + f(i,j+1) - f(i+1,j+1)$

Теорема. Число классов эквивалентности полных путей равно $f(1,1)$ и может быть вычислено за O(k*n) операций.

Предположим, что классы эквивалентности занумерованы числами от 1 до $f(1,1)$. В работе представлен алгоритм построения пути, принадлежащего классу эквивалентности с заданным номером. Находясь в вершине v_y^x, принимается решение о включении в выстраиваемый путь вершины v_y^{x+1}, либо вершины v_{y+1}^x, в зависимости от значений $f(x,y)$, $f(x+1,y)$ и $f(x,y+1)$.

Теорема. Алгоритм позволяет построить представителей всех классов эквивалентности, требуя при этом O(k+n) действий для нахождения одного пути.



Анализ графа совместного исполнения

- Для анализа достаточно вычислить результирующие состояния исполнения *S* только для одного представителя каждого из классов.
- Удалив из графа дуги, которые не входят в полные пути, соответствующие выбранным представителям классов, получим *редуцированный граф* совместного исполнения потоков.
- Результирующее состояние исполнения потоков может быть вычислено в неопределенных коэффициентах, двигаясь от начальной вершины графа к конечной.
- Чтобы определить, возможно ли возникновение неразрешенного состояния гонки, остается проанализировать полученное состояние *S* с помощью функции корректности *p*.

Новая идея – уменьшение перебора

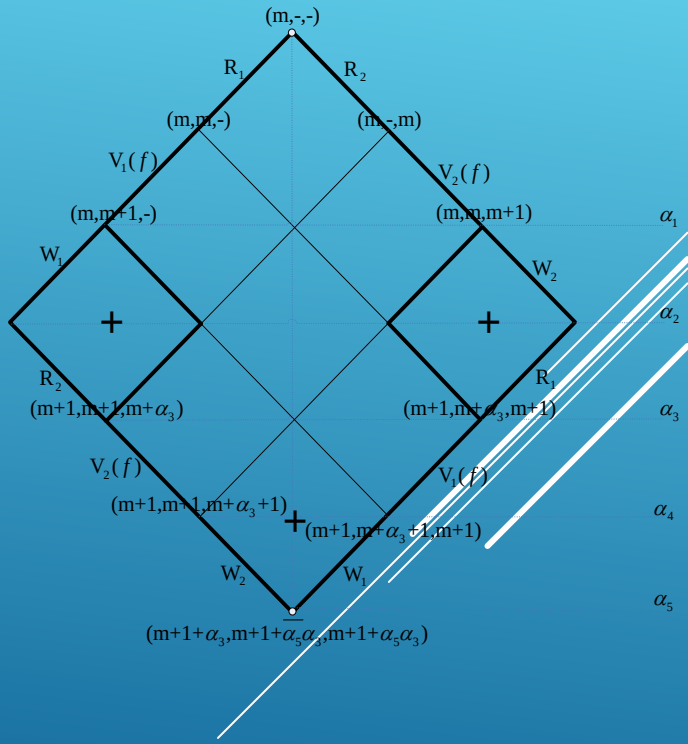
- Задача «плоха» – полный перебор всех путей на направленном графе (NP?)

- Не ищем «приблизительных» решений – нужно «точное»
  - это специфика предметной области

- Для введения эквивалентности задаем искусственно понятие функции корректности, «сегментирующее» область
  - Есть «понятное» визуальное представление задачи – полный перебор всех разных путей «вокруг» выделенных клеток на матрице – все равно полный перебор, но…

- Не решаем «в лоб», а просто уменьшаем перебор, выбрасывая все пути, кроме одного, из каждого класса эквивалентности

- Для решения проводим анализ одного представителя каждого из классов эквивалентности
  - Большинство практических задач имеет небольшие вполне «перебираемые» графы



Методика анализа

- Построение графа совместного исполнения потоков
- Поиск эквивалентных путей на графе
- Анализ представителя каждого из классов эквивалентности с помощью функции корректности *p*
- Ответ на вопрос о правильной работе многопоточного алгоритма

Изменение ячейки в двух потоках

Представление состояния исполнения потоков S в виде совокупности векторов (a,b,c), где

- a – это значение ячейки памяти,
- b – считанное значение ячейки, которым оперирует первый поток,
- c – считанное значение ячейки, которым оперирует второй поток.

$V_i(f)$ - модификация считанного значения, с помощью функции *f.*

Первый поток выполняет $R_1\ V_1(f)\ W_1,\ f(x)=x+1$

Второй поток выполняет $R_2\ V_2(f)\ W_2,\ f(x)=x+1$

Начальное состояние исполнения (m,-,-).

Функция корректности *p* – требованиеконстантности результирующего состояния исполнения потоков.

Число вариантов исполнения 20, анализируемых 4.

innopolis university

Новая идея – метод неопр. коэфф.

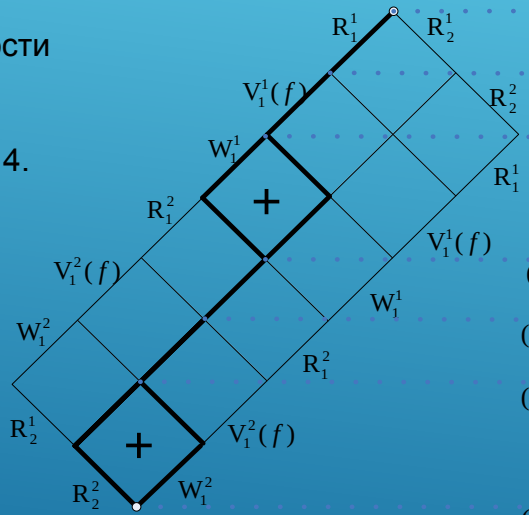
- Идея из вычислительной математики (методика построения схем А.С. Холодова)
- При анализе графа мы не знаем, «куда пойдем» - то есть, надо рассматривать все возможные решения
- Вводим неопределенные коэффициенты в каждой точке возможного ветвления алгоритма
- Совокупность значений каждого коэффициента описывает один конкретный путь – «что могло случиться»
- Описаны все пути, то есть, для анализа на условия гонки надо подобрать пару противоречивых путей (совокупностей значений коэффициентов)
- Значения ячеек в некоторых случаях легко выразить через значения неопределенных коэффициентов
- Наличие неопределенных коэффициентов в значениях ячеек могут (не обязаны) служить признаком наличия условия гонки.

Транзакционное изменение двух ячеек

R V (*f*) W , R V (*g*) W1 , *f* (*x*) = *x* - *k*, *g*(*x*) = *x* + *k*

Функция корректности *p* – требование константности результирующего состояния исполнения потоков.

Число вариантов исполнения 28, анализируемых 4.



$(m_1,-,-)(m_2,-,-)$

$(m_1,m_1,-)(m_2,-,-)$

$(m_1,m_1-k,-)(m_2,-,-)$

α_4

$(m_1-k,m_1-k,m_1-\alpha_4 k)(m_2,-,-)$

$(m_1-k,m_1-k,m_1-\alpha_4 k)(m_2,m_2,-)$

$(m_1-k,m_1-k,m_1-\alpha_4 k)(m_2,m_2+k,-)$

α_8

$(m_1-k,m_1-k,m_1-\alpha_4 k)(m_2+k,m_2+k,m_2+\alpha_8 k)$

Общие выводы

- Современная математика меняется - «излишняя» сложность пытается компенсироваться новыми подходами

- Представлено несколько примеров новой математики XXI века, возникших как ответ на реальные проблемы

- Новая математика потребовала новых подходов,«заимствующих» отчасти «чужие» идеи – динамику доказываемого, использование понятия «знания» не так, как это делает теория информации и тд – вплоть до идей квантовой механики!

- Не всегда то, что мы доказываем, является «истиной в последней инстанции» – практика дает возможность получения практически пригодных решений нерешаемых проблем, причем не только приблизительных!

Спасибо за внимание!

© А. Тормасов, 2013 г
tor@phystech.edu