Распределенные алгоритмы

ЛЕКТОР: В.А. Захаров

Лекция 12.

Задача сохранения моментального состояния

Алгоритм Чанди-Лампорта

Алгоритм Лаи-Янга

Применение алгоритмов сохранения моментального состояния

Задача сохранения моментального состояния

Следить за вычислениями распределенной системы **изнутри** самой системы — это трудная задача. При проектировании алгоритмов, работающих с системными вычислениями, важным конструктивным блоком служит процедура расчета и сохранения в памяти отдельной конфигурации этого вычисления, так называемого моментального состояния системы (**snapshot**).

Распределенное вычисление — это последовательность конфигураций распределенной системы. Конфигурация системы — это набор конфигураций составляющих ее процессов плюс содержание каналов связи. Каждый процесс может сохранять в памяти свою конфигурацию. Однако не всякий набор конфигураций процессов образует конфигурацию системы.

Как научить процессы запоминать свои конфигурации **согласованно**?

Задача сохранения моментального состояния

Построение моментальных состояний системы используется в ряде приложений.

1) Свойства вычисления можно анализировать в режиме *off-line*, исследуя (фиксированное) моментальное состояние всей системы, а не (изменчивые) текущие состояния процессов.

Свойство P конфигураций называется устойчивым , если справедливо соотношение $P(\gamma) \wedge \gamma \rightsquigarrow \delta \implies P(\delta)$.

Если будет обнаружено, что свойство P выполняется для моментального состояния некоторой конфигурации системы, то отсюда можно сделать вывод о выполнимости P для всех последующих конфигураций. К устойчивым свойствам относятся завершенность вычисления, блокировка (попадание в тупиковую ситуацию), потеря маркера, а также недостижимость объектов в структурах динамической памяти.

Задача сохранения моментального состояния

2) Моментальные состояния системы можно использовать вместо начальных конфигураций, если требуется перезапустить вычисление ввиду отказа какого-либо процесса.
Для этого процесс p следует переустановить в то локальное состояние c_p , которое было запечатлено в моментальном состоянии системы, после чего работа алгоритма будет продолжена.

Задача сохранения моментального состояния

2) Моментальные состояния системы можно использовать вместо начальных конфигураций, если требуется перезапустить вычисление ввиду отказа какого-либо процесса.
Для этого процесс p следует переустановить в то локальное состояние c_p , которое было запечатлено в моментальном состоянии системы, после чего работа алгоритма будет продолжена.

3) Моментальные состояния системы являются полезным средством отладки распределенных программ. Проведя в режиме off-line анализ конфигурации, «выхваченной» из ошибочного выполнения, можно обнаружить причину, по которой программа ведет себя не так, как этого ожидают.

Задача сохранения моментального состояния

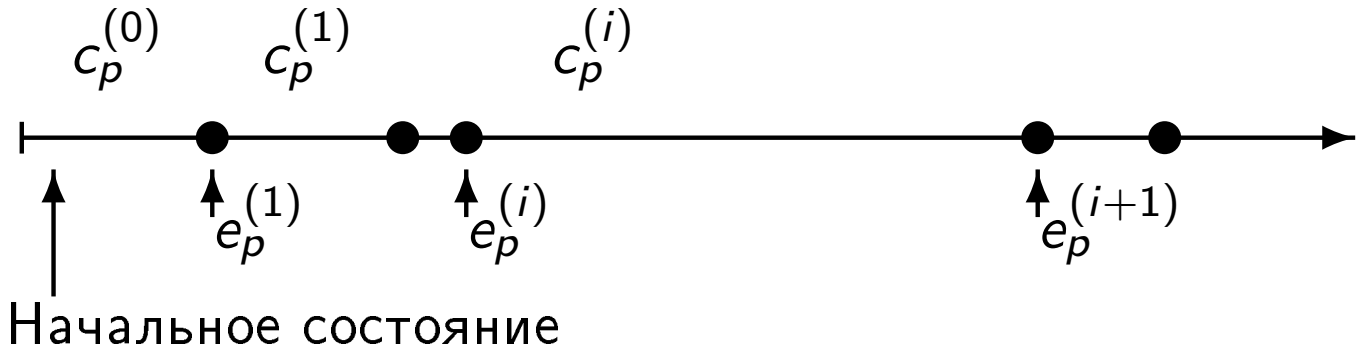
Рассмотрим вычисление C распределенной системы, состоящей из некоторого множества процессов $\mathbb{P}$.

Обозначим символом Ev множество событий этого вычисления. Действует допущение слабой справедливости: каждое сообщение будет доставлено по назначению спустя конечный промежуток времени.

Мы будем полагать, что сеть является сильно связной.

Локальное вычисление процесса p представляет собой последовательность $c_p^{(0)}, c_p^{(1)}, \ldots$ состояний процесса, где $c_p^{(0)}$ — это начальное состояние процесса p . Переход из состояния $c_p^{(i-1)}$ в состояние $c_p^{(i)}$ обусловлен осуществлением события $e_p^{(i)}$ в процессе p . Таким образом, $Ev = \cup_{p \in \mathbb{P}} \{e_p^{(1)}, e_p^{(2)}, \ldots\}$.

Задача сохранения моментального состояния



Задача сохранения моментального состояния

На множестве событий процесса p введем причинно-следственный порядок $\preceq_p$ при помощи соотношения

$$e_p^{(i)} \preceq_p e_p^{(j)} \iff i \leq j.$$

Каждое событие может быть событием отправления сообщения , приема сообщения или внутренним событием .

Для упрощения описания алгоритмов и формулировки теорем мы будем придерживаться допущения о том, что вся история обмена информацией с процессом определяется его состоянием.

Для канала связи, ведущего из p в q , состояние $c_p^{(i)}$ процесса p включает в себя список $sent_{pq}^{(i)}$ сообщений, которые процесс p отправил процессу q при осуществлении событий начиная с $e_p^{(1)}$ и оканчивая $e_p^{(i)}$.

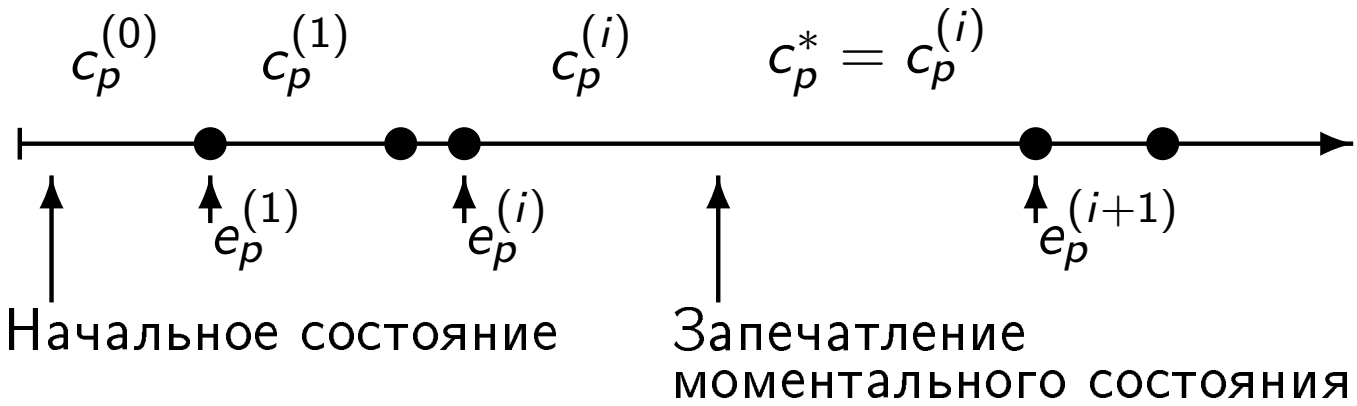
Состояние $c_q^{(i)}$ процесса q также содержит список $rcvd_{pq}^{(i)}$ сообщений, которые процесс q получил от процесса p при осуществлении событий, начиная с $e_q^{(1)}$ и оканчивая $e_q^{(i)}$.

Задача сохранения моментального состояния

Алгоритм построения моментального состояния системы предназначен для реконструкции глобальной конфигурации, которая образуется из локальных состояний (моментальных состояний) всех процессов.
Процесс p запечатлевает свое локальное состояние , записывая в память одно локальное состояние c_p^* , которое называется моментальным локальным состоянием процесса p .

Если моментальным локальным состоянием процесса является $c_p^{(i)}$, т. е. p проводит моментальный снимок состояния в промежутке между осуществлением событий $e_p^{(i)}$ и $e_p^{(i+1)}$, то события $e_p^{(j)}$, для которых $j \leq i$, называются предмоментальными событиями процесса p , а те события, для которых $j > i$, называются постмоментальными событиями процесса p .

Задача сохранения моментального состояния



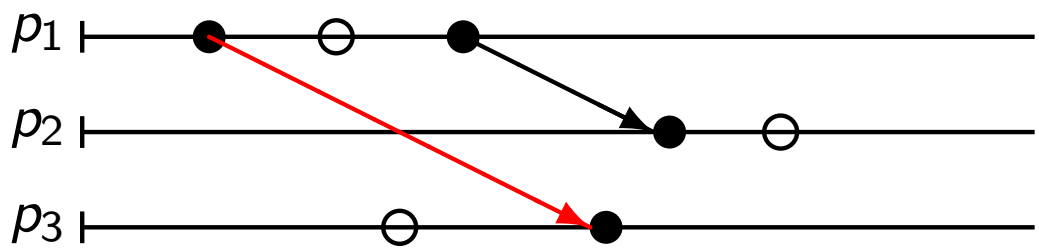
Задача сохранения моментального состояния

Глобальное моментальное состояние S^* образуется из моментальных состояний c_p^* всех процессов p , входящих в множество $\mathbb{P}$. Для обозначения этого мы будем использовать запись $S^* = (c_{p_1}^*, \ldots, c_{p_N}^*)$.

Так как локальные состояния включают в себя истории обменов информацией, моментальное состояние S^* определяет конфигурацию γ^* , если состоянием канала pq считать множество сообщений, отправленных по этому каналу процессом p (согласно состоянию c_p^*), но еще не доставленных процессу q (согласно состоянию c_q^*). Иными словами, состояние канала pq в моментальном состоянии S^* определяется множеством сообщений $sent_{pq}^* \setminus rcvd_{pq}^*$.

Конфигурацию, состоящую из моментальных состояний процессов и определенных таким образом состояний каналов, обозначим символом γ^* .

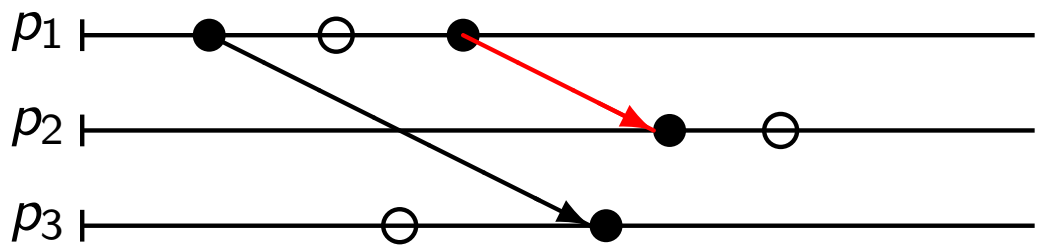
Задача сохранения моментального состояния



При построении конфигурации γ^* появляются некоторые отклонения, если множество $rcvd^*_{pq}$ не является подмножеством $sent^*_{pq}$. Судя по локальному состоянию $c^*_{p_1}$ в формируемом моментальном состоянии системы, сообщение было отправлено процессом p_1 процессу p_3 , но, как отмечено в локальном состоянии $c^*_{p_3}$, никакого сообщения от p_1 не было получено.

Таким образом, в данном моментальном состоянии системы канал $p_1 p_3$ содержит одно сообщение; говорят, что в этом моментальном состоянии системы указанное сообщение пребывает «на этапе пересылки».

Задача сохранения моментального состояния



Обратимся теперь к сообщению, которое процесс p_1 отправил процессу p_2 .

Отправление этого сообщения относится к числу постмоментальных событий, тогда как его получение является предмоментальным событием.

Значит, в соответствии с состоянием $c^*_{p_1}$ никакие сообщения по каналу $p_1 p_2$ не отправлялись, тогда как в состоянии $c^*_{p_2}$ отмечено, что по этому каналу было получено некоторое сообщение. Поскольку $rcvd^*_{p_1 p_2} \not\subseteq sent^*_{p_1 p_2}$, никакого осмысленного состояния каналу $p_1 p_2$ придать нельзя.

Задача сохранения моментального состояния

Определение 1.

Моментальное состояние S^* называется осуществимым , если для всякой пары соседних процессов p и q выполняется включение $rcvd^*_{pq} \subseteq sent^*_{pq}$.

Осуществимость моментального состояния означает, что при построении соответствующей конфигурации в $rcvd^*_{pq}$ не останется ни одного сообщения, которое нельзя удалить из $sent^*_{pq}$. Сообщение, отправление которого является предмоментальным (постмоментальным) событием, будем называть предмоментальным (соответственно, постмоментальным) сообщением.

Задача сохранения моментального состояния

Существует взаимнооднозначное соответствие между моментальными состояниями и конечными сечениями на совокупности событий вычисления. Сечением называется совокупность событий, замкнутая влево относительно отношения локальной причинно-следственной зависимости.

Определение 2.

Сечением на множестве Ev называется такое подмножество $L \subseteq Ev$, для которого выполняется соотношение

$$e \in L \wedge e' \preceq_p e \quad \Longrightarrow \quad e' \in L.$$

Сечение L_2 считается **более поздним**, чем сечение L_1 , если верно включение $L_1 \subseteq L_2$.

Согласованным сечечнием на множестве событий Ev называется такое подмножество $L \subseteq Ev$, для которого выполняется соотношение

Задача сохранения моментального состояния

Нетрудно видеть, что для каждого моментального состояния S^* совокупность L всех предмоментальных событий является конечным сечением.

Рассмотрим теперь произвольное конечное сечение L . Для каждого процесса p либо ни одно событие, произошедшее в p , не входит в L (в таком случае будем полагать $m_p = 0$), либо в L существует наибольшее событие $e_p^{(m_p)}$ и при этом все события $e \preceq_p e_p^{(m_p)}$ также содержатся в L . Поэтому L представляет собой в точности множество предмоментальных событий моментального состояния $S^* = (c_{p_1}^{(m_{p_1})}, \ldots, c_{p_N}^{(m_{p_N})})$.

Задача сохранения моментального состояния

Моментальное состояние будет использоваться для извлечения информации о том вычислении, которое оно запечатлевает, но произвольно выбранное моментальное состояние системы дает очень мало информации о таком вычислении. Лучше, чтобы алгоритм построения моментального состояния реконструировал конфигурацию, которая и «в самом деле» может возникнуть при вычислении.

К сожалению, множество реализуемых конфигураций не является инвариантом отношения эквивалентности, и поэтому оно не определяется вычислением однозначно. Таким образом, мы будем считать разумным всякий результат работы алгоритма, если он приводит к построению какой-либо конфигурации, которая окажется возможной для заданного вычисления (т. е. может случится в некоторой реализации этого вычисления).

Задача сохранения моментального состояния

Определение 3.

Моментальное состояние S^* называется значимым в вычислении C, если существует такая реализация $E \in C$, что γ^* является конфигурацией E.

Мы будем требовать от алгоритма построения моментального состояния такой координации регистрации локальных моментальных состояний, чтобы полученное в результате этого глобальное моментальное состояние оказалось значимым.

Задача сохранения моментального состояния

Теорема 1.

Пусть S^* — моментальное состояние системы и L — сечение, порожденное S^*. Тогда равносильны следующие три утверждения:

1) S^* осуществимо;
2) L согласовано;
3) S^* значимо.

Задача сохранения моментального состояния

Теорема 1.

Пусть S^* — моментальное состояние системы и L — сечение, порожденное S^*. Тогда равносильны следующие три утверждения:

1) S^* осуществимо;
2) L согласовано;
3) S^* значимо.

Доказательство.

Покажем справедливость соотношений

$$(1) \Longrightarrow (2) \Longrightarrow (3) \Longrightarrow (3)$$

Задача сохранения моментального состояния

Доказательство. (1) $\Rightarrow$ (2)

Допустим, что состояние S^* осуществимо. Чтобы убедиться в том, что L согласовано, рассмотрим $e \in L$ и $e' \preceq e$. По определению отношения $\preceq$ достаточно показать, что включение $e' \in L$ выполняется в следующих двух случаях.

Задача сохранения моментального состояния

Доказательство. (1) $\Rightarrow$ (2)

Допустим, что состояние S^* осуществимо. Чтобы убедиться в том, что L согласовано, рассмотрим $e \in L$ и $e' \preceq e$. По определению отношения $\preceq$ достаточно показать, что включение $e' \in L$ выполняется в следующих двух случаях.

1. Выполняется соотношение $e' \preceq_p e$ (где p — это процесс, в котором происходят e' и e). В таком случае включение $e' \in L$ выполняется благодаря тому, что L — сечение.

Задача сохранения моментального состояния

Доказательство. (1) $\Rightarrow$ (2)

Допустим, что состояние S^* осуществимо. Чтобы убедиться в том, что L согласовано, рассмотрим $e \in L$ и $e' \preceq e$. По определению отношения $\preceq$ достаточно показать, что включение $e' \in L$ выполняется в следующих двух случаях.

1. Выполняется соотношение $e' \preceq_p e$ (где p — это процесс, в котором происходят e' и e). В таком случае включение $e' \in L$ выполняется благодаря тому, что L — сечение.

2. Событие e' — это событие отправления сообщения, а событие e — соответствующее ему событие приема сообщения.

Задача сохранения моментального состояния

Доказательство. (1) $\Rightarrow$ (2)

Допустим, что состояние S^* осуществимо. Чтобы убедиться в том, что L согласовано, рассмотрим $e \in L$ и $e' \preceq e$. По определению отношения $\preceq$ достаточно показать, что включение $e' \in L$ выполняется в следующих двух случаях.

1. Выполняется соотношение $e' \preceq_p e$ (где p — это процесс, в котором происходят e' и e). В таком случае включение $e' \in L$ выполняется благодаря тому, что L — сечение.

2. Событие e' — это событие отправления сообщения, а событие e — соответствующее ему событие приема сообщения.Рассмотрим процесс p , в котором происходит событие e' , процесс q , в котором происходит e , и сообщение m , которым обмениваются процессы. Тогда

Задача сохранения моментального состояния

Доказательство. (1) $\Rightarrow$ (2)

Допустим, что состояние S^* осуществимо. Чтобы убедиться в том, что L согласовано, рассмотрим $e \in L$ и $e' \preceq e$. По определению отношения $\preceq$ достаточно показать, что включение $e' \in L$ выполняется в следующих двух случаях.

1. Выполняется соотношение $e' \preceq_p e$ (где p — это процесс, в котором происходят e' и e). В таком случае включение $e' \in L$ выполняется благодаря тому, что L — сечение.

2. Событие e' — это событие отправления сообщения, а событие e — соответствующее ему событие приема сообщения.Рассмотрим процесс p , в котором происходит событие e' , процесс q , в котором происходит e , и сообщение m , которым обмениваются процессы. Тогда

$$e \in L \implies m \in rcvd^*_{pq}, \quad \text{т.к. } e \text{ — предмоментальное событие}$$

Задача сохранения моментального состояния

Доказательство. (1) $\Rightarrow$ (2)

Допустим, что состояние S^* осуществимо. Чтобы убедиться в том, что L согласовано, рассмотрим $e \in L$ и $e' \preceq e$. По определению отношения $\preceq$ достаточно показать, что включение $e' \in L$ выполняется в следующих двух случаях.

1. Выполняется соотношение $e' \preceq_p e$ (где p — это процесс, в котором происходят e' и e). В таком случае включение $e' \in L$ выполняется благодаря тому, что L — сечение.

2. Событие e' — это событие отправления сообщения, а событие e — соответствующее ему событие приема сообщения.Рассмотрим процесс p , в котором происходит событие e' , процесс q , в котором происходит e , и сообщение m , которым обмениваются процессы. Тогда

$$
\begin{aligned}
e \in L \quad &\Longrightarrow \quad m \in rcvd^*_{pq}, && \text{т.к. } e \text{ — предмоментальное событие} \\
&\Longrightarrow \quad m \in sent^*_{pq}, && \text{т.к. } S^* \text{ осуществимо}
\end{aligned}
$$

Задача сохранения моментального состояния

Доказательство. (1) $\Rightarrow$ (2)

Допустим, что состояние S^* осуществимо. Чтобы убедиться в том, что L согласовано, рассмотрим $e \in L$ и $e' \preceq e$. По определению отношения $\preceq$ достаточно показать, что включение $e' \in L$ выполняется в следующих двух случаях.

1. Выполняется соотношение $e' \preceq_p e$ (где p — это процесс, в котором происходят e' и e). В таком случае включение $e' \in L$ выполняется благодаря тому, что L — сечение.

2. Событие e' — это событие отправления сообщения, а событие e — соответствующее ему событие приема сообщения.Рассмотрим процесс p , в котором происходит событие e' , процесс q , в котором происходит e , и сообщение m , которым обмениваются процессы. Тогда

$$
\begin{aligned}
e \in L \quad &\Longrightarrow \quad m \in rcvd^*_{pq}, && \text{т.к. } e \text{ — предмоментальное событие} \\
&\Longrightarrow \quad m \in sent^*_{pq}, && \text{т.к. } S^* \text{ осуществимо} \\
&\Longrightarrow \quad e' \in L.
\end{aligned}
$$

Задача сохранения моментального состояния

Доказательство. (2) $\Rightarrow$ (3)

Задача сохранения моментального состояния

Доказательство. (2) $\Rightarrow$ (3)

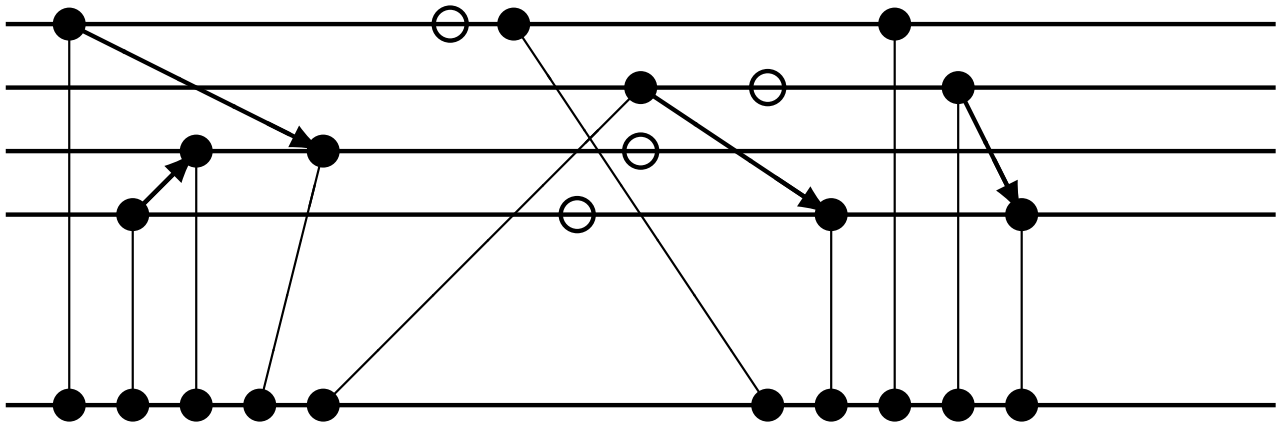
Построим такое выполнение вычисления, в котором все предмоментальные события происходят ранее любого постмоментального события.

Введем следующую нумерацию $f = (f_0, f_1, \ldots)$ событий множества Ev .

Вначале перечислим все предмоментальные события из Ev в произвольном порядке, согласованном с отношением $\preceq$, а далее перечислим все постмоментальные события в произвольном порядке, согласованном с отношением $\preceq$.

Задача сохранения моментального состояния

Пространственно-временная диаграмма C



Последовательность f

Задача сохранения моментального состояния

Доказательство. (2) $\Rightarrow$ (3)

Построим такое выполнение вычисления, в котором все предмоментальные события происходят ранее любого постмоментального события.

Введем следующую нумерацию $f = (f_0, f_1, \ldots)$ событий множества Ev .

Вначале перечислим все предмоментальные события из Ev в произвольном порядке, согласованном с отношением $\preceq$, а далее перечислим все постмоментальные события в произвольном порядке, согласованном с отношением $\preceq$.

Воспользуемся теоремой о вычислениях.

Пусть $f = (f_0, f_1, f_2, \ldots)$ — перестановка событий выполнения E , сохраняющая причинно-следственный порядок событий выполнения . Тогда f определяет единственное выполнение F , начинающееся с той же конфигурации, что и выполнение E .

Задача сохранения моментального состояния

Чтобы воспользоваться этой теоремой, покажем, что вся последовательность f согласована с отношением $\preceq$. Допустим, что $f_i \preceq f_j$.

Задача сохранения моментального состояния

Чтобы воспользоваться этой теоремой, покажем, что вся последовательность f согласована с отношением $\preceq$. Допустим, что $f_i \preceq f_j$.

Если оба события f_i и f_j предмоментальные, то $i \leq j$, поскольку в f предмоментальные события перечислены в порядке, согласованном с отношением $\preceq$.

Задача сохранения моментального состояния

Чтобы воспользоваться этой теоремой, покажем, что вся последовательность f согласована с отношением $\preceq$. Допустим, что $f_i \preceq f_j$.

Если оба события f_i и f_j предмоментальные, то $i \leq j$, поскольку в f предмоментальные события перечислены в порядке, согласованном с отношением $\preceq$.

По тем же причинам это неравенство выполняется, когда оба события f_i и f_j являются постмоментальными.

Задача сохранения моментального состояния

Чтобы воспользоваться этой теоремой, покажем, что вся последовательность f согласована с отношением $\preceq$. Допустим, что $f_i \preceq f_j$.

Если оба события f_i и f_j предмоментальные, то $i \leq j$, поскольку в f предмоментальные события перечислены в порядке, согласованном с отношением $\preceq$.

По тем же причинам это неравенство выполняется, когда оба события f_i и f_j являются постмоментальными.

Если f_i — предмоментальное событие, а f_j — постмоментальное, то неравенство $i \leq j$ соблюдается т.к. в f все предмоментальные события предшествуют постмоментальным.

Задача сохранения моментального состояния

Чтобы воспользоваться этой теоремой, покажем, что вся последовательность f согласована с отношением $\preceq$. Допустим, что $f_i \preceq f_j$.

Если оба события f_i и f_j предмоментальные, то $i \leq j$, поскольку в f предмоментальные события перечислены в порядке, согласованном с отношением $\preceq$.

По тем же причинам это неравенство выполняется, когда оба события f_i и f_j являются постмоментальными.

Если f_i — предмоментальное событие, а f_j — постмоментальное, то неравенство $i \leq j$ соблюдается т.к. в f все предмоментальные события предшествуют постмоментальным.

Случай, когда f_j — предмоментальное событие, а f_i — постмоментальное, исключен, ввиду того что сечение L по предположению является согласованным.

Задача сохранения моментального состояния

Итак, последовательность f согласована с отношением $\preceq$. Значит, по теореме о вычислениях существует выполнение вычисления F , состоящее из всех событий множества Ev , осуществляемых в том порядке, который представлен последовательностью f . Выполнение F содержит конфигурацию γ^* , которая возникает сразу же после осуществления всех предмоментальных событий.

Доказательство. (3) $\Rightarrow$ (1)

Задача сохранения моментального состояния

Итак, последовательность f согласована с отношением $\preceq$. Значит, по теореме о вычислениях существует выполнение вычисления F , состоящее из всех событий множества Ev , осуществляемых в том порядке, который представлен последовательностью f . Выполнение F содержит конфигурацию γ^* , которая возникает сразу же после осуществления всех предмоментальных событий.

Доказательство. (3) $\Rightarrow$ (1)

Если моментальное состояние S^* является значимым, то конфигурация γ^* возникает при выполнении вычисления C . В каждом выполнении всякое сообщение не может быть доставлено по назначению, прежде чем оно будет отправлено. Отсюда следует, что для каждой пары процессов p и q имеет место соотношение $rcvd^*_{pq} \subseteq sent^*_{pq}$. Поэтому состояние S^* осуществимо.

Алгоритм Чанди–Лампорта

По теореме 1 достаточно скоординировать локальные моментальные состояния, чтобы выполнялись следующие два свойства:

Алгоритм Чанди–Лампорта

По теореме 1 достаточно скоординировать локальные моментальные состояния, чтобы выполнялись следующие два свойства:

1. выбранные в каждом процессе локальные моментальные состояния должны быть реализованы.

Алгоритм Чанди–Лампорта

По теореме 1 достаточно скоординировать локальные моментальные состояния, чтобы выполнялись следующие два свойства:

1. выбранные в каждом процессе локальные моментальные состояния должны быть реализованы.
2. получение постмоментального сообщения не может составлять предмоментальное событие.

Алгоритм Чанди–Лампорта

По теореме 1 достаточно скоординировать локальные моментальные состояния, чтобы выполнялись следующие два свойства:

1. выбранные в каждом процессе локальные моментальные состояния должны быть реализованы.
2. получение постмоментального сообщения не может составлять предмоментальное событие.

Во всех алгоритмах построения моментального состояния всякий процесс запечатлевает свое локальное моментальное состояние, до того как получит постмоментальное сообщение.

Чтобы отличать сообщения в алгоритме построения моментального состояния от сообщений в самом вычислении, мы будем называть сообщения первого вида контрольными сообщениями , а сообщения второго вида — базовыми сообщениями .

Алгоритм Чанди–Лампорта

Алгоритм применим к сильно связной сети с произвольной топологией, однонаправленными каналами, в которых поддерживается очередность сообщений.

Алгоритм Чанди–Лампорта

Алгоритм применим к сильно связной сети с произвольной топологией, однонаправленными каналами, в которых поддерживается очередность сообщений.

Идея алгоритма.

В алгоритме Ченди—Лэмпорта процессы сообщают друг другу о построении моментальных состояний путем отправления специальных сообщений (**маркеров**) $\langle \mathbf{mkr} \rangle$ по каждому каналу.

Каждый процесс, запечатлевая свое локальное состояние, отправляет маркер в точности один раз по каждому примыкающему к нему каналу; маркеры рассматриваются как контрольные сообщения.

Получение сообщения $\langle \mathbf{mkr} \rangle$ каким-либо процессом, который еще не запечатлел свое моментальное состояние, приводит к тому, что этот процесс запечатлевает свое моментальное состояние и также отправляет маркер $\langle \mathbf{mkr} \rangle$, который выполняется параллельно с вычислением C .

Алгоритм Чанди–Лампорта

var $taken_p$: **bool** **init** *false* ;

Для запуска алгоритма:
begin запомнить локальное состояние ; $taken_p := true$;
forall $q \in Neigh_p$ **do** send $\langle$**mkr**$\rangle$ to q
end

Если был доставлен маркер:
begin receive $\langle$**mkr**$\rangle$;
if not $taken_p$ **then**
begin запомнить локальное состояние ; $taken_p := true$;
forall $q \in Neigh_p$ **do** send $\langle$**mkr**$\rangle$ to q
end
end

Алгоритм Чанди–Лампорта

Лемма.
Если хотя бы один процесс запустит алгоритм Чанди–Лампорта, то спустя конечный промежуток времени все процессы запечатлеют свое локальное моментальное состояние.

Алгоритм Чанди–Лампорта

Лемма.
Если хотя бы один процесс запустит алгоритм Чанди–Лампорта, то спустя конечный промежуток времени все процессы запечатлеют свое локальное моментальное состояние.

Доказательство.
Самостоятельно

Алгоритм Чанди–Лампорта

Теорема 2.

Алгоритм Ченди—Лэмпорта вычисляет значимое моментальное состояние системы.

Алгоритм Чанди–Лампорта

Теорема 2.

Алгоритм Ченди—Лэмпорта вычисляет значимое моментальное состояние системы.

Доказательство.

Согласно лемме алгоритму нужно конечное время для вычисления моментального состояния системы. Покажем, что построенное моментальное состояние осуществимо.

Алгоритм Чанди–Лампорта

Теорема 2.

Алгоритм Ченди—Лэмпорта вычисляет значимое моментальное состояние системы.

Доказательство.

Согласно лемме алгоритму нужно конечное время для вычисления моментального состояния системы. Покажем, что построенное моментальное состояние осуществимо.

Рассмотрим постмоментальное сообщение m , отправленное процессом p процессу q . Перед отправлением m , процесс p запечатлел свое моментальное локальное состояние и отправил сообщение $\langle \mathbf{mkr} \rangle$ всем своим соседям, включая процесс q . Т.к. в каналах поддерживается очередность сообщений, процесс q получил сообщение $\langle \mathbf{mkr} \rangle$, до того как получил m , и поэтому запечатлел свое моментальное состояние сразу по получении этого сообщения или еще ранее того. Следовательно, получение сообщения m относится к числу постмоментальных событий.

Алгоритм Лаи–Янга

Алгоритм Лая—Янга не опирается на свойство поддержания очередности сообщений в каналах. Поэтому отделить предмоментальные события от постмоментальных при помощи маркеров, как это сделано в алгоритме Ченди—Лэмпорта, уже невозможно.

Алгоритм Лаи–Янга

Алгоритм Лая—Янга не опирается на свойство поддержания очередности сообщений в каналах. Поэтому отделить предмоментальные события от постмоментальных при помощи маркеров, как это сделано в алгоритме Ченди—Лэмпорта, уже невозможно.

Идея алгоритма.

Каждое базовое сообщение снабжается пометкой, которая позволяет понять, является ли это сообщение предмоментальным или постмоментальным.

Для этого процесс p , отправляя сообщение по ходу вычисления C , добавляет к нему значение $taken_p$. Так как содержание сообщений, относящихся к вычислению C , не играет здеь никакой роли, мы будем обозначать такие сообщения просто $\langle \mathbf{mes}, c \rangle$, где c — это значение переменной, включенное в состав сообщения процессом-отправителем.

Алгоритм построения моментального состояния проверяет поступающие сообщения и записывает моментальное

Алгоритм Лаи–Янга

var $taken_p$: **bool** **init** *false* ;

Для запуска алгоритма:
begin запомнить локальное состояние ; $taken_p := true$ **end**

Для отправления сообщения по ходу вычисления C:
send $\langle \textbf{mes}, taken_p \rangle$

Если поступило сообщение $\langle \textbf{mes}, c \rangle$;
begin receive $\langle \textbf{mes}, c \rangle$;
if c **and not** $taken_p$ **then**
begin запомнить локальное состояние ; $taken := true$ **end**;
изменить состояние на то, в котором пребывал процесс
при получении сообщения в вычислении C
end

Алгоритм Лаи–Янга

В алгоритме Лаи–Янга нет обмена контрольными сообщениями, и он не может гарантировать того, что каждый процесс рано или поздно запишет свое состояние. Рассмотрим процесс p , не являющийся инициатором алгоритма построения моментального состояния, и предположим, что ни один из соседей процесса p не отправляет процессу p сообщений, после того как этот сосед запечатлел свое моментальное локальное состояние. В таком случае p никогда не проведет запись своего состояния, и алгоритм построения моментального состояния завершит работу, имея неполное моментальное состояние.

Решение этой проблемы зависит от того, что нам известно о вычислении C ; если мы имеем гарантию, что связь с каждым процессом рано или поздно осуществится, то и полное моментальное состояние всегда можно будет запечатлеть. В противном случае алгоритм можно снабдить блоком, проводящим первоначальный обмен контрольными сообщениями между всеми процессами.

Алгоритм Лаи–Янга

Теорема 3.

Алгоритм Лая—Янга вычисляет только значимые моментальные состояния системы.

Алгоритм Лаи–Янга

Теорема 3.

Алгоритм Лая—Янга вычисляет только значимые моментальные состояния системы.

Доказательство.

Рассмотрим моментальное состяние, вычисленное алгоритмом, и пусть $m = \langle \mathbf{mes}, c \rangle$ — это постмоментальное сообщение, отправленное процессом p процессу q . Это значит, что $c = true$, и поэтому q запечатлевает свое состояние таким, каким оно было до получения сообщения m .

Таким образом, в состоянии, которое записал процесс q , не учитывается поступление m , и прием сообщения m относится к числу постмоментальных событий.

(Существенно здесь то, какое состояние записать, а не то, когда провести эту запись; в нашем случае запись состояния проводится одновременно с получением первого постмоментального события.)

Применение алгоритмов сохранения моментального состояния

Рассмотрим устойчивое свойство P конфигураций, которое в случае выполнения будет продолжать выполняться и для каждой последующей конфигурации.

Применение алгоритмов сохранения моментального состояния

Рассмотрим устойчивое свойство P конфигураций, которое в случае выполнения будет продолжать выполняться и для каждой последующей конфигурации.

1. **Завершение вычисления.** Если γ — это заключительная конфигурация и $\gamma \rightsquigarrow \delta$, то $\gamma = \delta$, и поэтому конфигурация δ также является заключительной. Следовательно, задача обнаружения завершения может быть решена за счет вычисления моментального состояния системы и анализа активных процессов и базовых сообщений в этом моментальном состоянии.

Применение алгоритмов сохранения моментального состояния

Рассмотрим устойчивое свойство P конфигураций, которое в случае выполнения будет продолжать выполняться и для каждой последующей конфигурации.

1. **Завершение вычисления.** Если γ — это заключительная конфигурация и $\gamma \rightsquigarrow \delta$, то $\gamma = \delta$, и поэтому конфигурация δ также является заключительной. Следовательно, задача обнаружения завершения может быть решена за счет вычисления моментального состояния системы и анализа активных процессов и базовых сообщений в этом моментальном состоянии.
2. **Наличие тупиков.** Если в конфигурации γ некоторое множество процессов S оказывается заблокированными, из-за того что все процессы из S пребывают в ожидании поступления сообщений от других процессов из S, то такая же ситуация повторится и в последующих конфигурациях, хотя процессы вне множества S могут изменять свои состояния.

Применение алгоритмов сохранения моментального состояния

3. **Потеря маркеров.** Рассмотрим алгоритм, в котором маркеры циркулируют между процессами, а сами процессы могут поглощать маркеры. Свойство вида «Имеется не более k маркеров» является устойчивым, так как маркеры могут поглощаться, но не могут порождаться.

Применение алгоритмов сохранения моментального состояния

3. **Потеря маркеров.** Рассмотрим алгоритм, в котором маркеры циркулируют между процессами, а сами процессы могут поглощать маркеры. Свойство вида «Имеется не более k маркеров» является устойчивым, так как маркеры могут поглощаться, но не могут порождаться.
4. **Наличие «мусора».** В объектно-ориентированной среде программирования создается совокупность объектов, каждый из которых может содержать *ссылку* на другие объекты. Объект считается *доступным*, если можно отыскать путь, ведущий по ссылкам к данному объекту от некоторого предписанного объекта; в противном случае объект объявляется *«мусором»*. Ссылки могут появляться и исчезать, но ссылки, ведущие к «мусорным» объектам, никогда не возникают. Поэтому, как только объект становится «мусором», он будет во веки вечные оставаться таковым.

КОНЕЦ ЛЕКЦИИ 12.