Распределенные алгоритмы

ЛЕКТОР: В.А. Захаров

Лекция 2.

Математическая модель. Системы переходов.
Системы с синхронным и асинхронным обменом сообщениями.
Свойство справедливости выполнений системы.
Зависимые и независимые события.
Причинно-следственный порядок событий.
Эквивалентность выполнений. Вычисления.
Логические часы.

Математическая модель.

При изучении распределенных алгоритмов используются несколько различных моделей распределенной обработки информации. Конкретная модель выбирается зависимости от исследуемой задачи из области распределенных вычислений, а также от представленной разновидности алгоритма (или стремления доказать невозможность построения такового).

Математическая модель.

При изучении распределенных алгоритмов используются несколько различных моделей распределенной обработки информации. Конкретная модель выбирается зависимости от исследуемой задачи из области распределенных вычислений, а также от представленной разновидности алгоритма (или стремления доказать невозможность построения такового).

От модели требуется быть:

Математическая модель.

При изучении распределенных алгоритмов используются несколько различных моделей распределенной обработки информации. Конкретная модель выбирается зависимости от исследуемой задачи из области распределенных вычислений, а также от представленной разновидности алгоритма (или стремления доказать невозможность построения такового).

От модели требуется быть:

- **точной**, чтобы иметь возможность оценивать сложность алгоритмов и получать отрицательные результаты;

Математическая модель.

При изучении распределенных алгоритмов используются несколько различных моделей распределенной обработки информации. Конкретная модель выбирается зависимости от исследуемой задачи из области распределенных вычислений, а также от представленной разновидности алгоритма (или стремления доказать невозможность построения такового).

От модели требуется быть:

- **точной**, чтобы иметь возможность оценивать сложность алгоритмов и получать отрицательные результаты;
- **общей**, чтобы применять ее к целому *классу* родственных распределенных систем;

Математическая модель.

При изучении распределенных алгоритмов используются несколько различных моделей распределенной обработки информации. Конкретная модель выбирается зависимости от исследуемой задачи из области распределенных вычислений, а также от представленной разновидности алгоритма (или стремления доказать невозможность построения такового).

От модели требуется быть:

- **точной**, чтобы иметь возможность оценивать сложность алгоритмов и получать отрицательные результаты;
- **общей**, чтобы применять ее к целому *классу* родственных распределенных систем;
- **компактной**, чтобы ее можно было всесторонне рассмотреть при проведении доказательств.

Математическая модель.

Распределенное вычисление обычно рассматривается в виде упорядоченной совокупности дискретных *событий*, каждое из которых представляет собой небольшое изменение *конфигурации* (глобального состояния всей системы).

Математическая модель.

Распределенное вычисление обычно рассматривается в виде упорядоченной совокупности дискретных *событий*, каждое из которых представляет собой небольшое изменение *конфигурации* (глобального состояния всей системы).

Правила изменения конфигураций задаются в виде *системы переходов*.

Математическая модель.

Распределенное вычисление обычно рассматривается в виде упорядоченной совокупности дискретных *событий*, каждое из которых представляет собой небольшое изменение *конфигурации* (глобального состояния всей системы).

Правила изменения конфигураций задаются в виде *системы переходов*.

Система становится «распределенной» благодаря тому обстоятельству, что при выполнении каждого перехода приходится иметь дело только с частью конфигурации, а именно, с *локальным* состоянием одного процесса (или с локальными состояниями некоторого подмножества взаимодействующих процессов.)

Математическая модель.

Определение 1.

Системой переходов называется тройка $S = (\mathcal{C}, \rightarrow, \mathcal{I})$, в которой

Математическая модель.

Определение 1.

Системой переходов называется тройка $S = (\mathcal{C}, \rightarrow, \mathcal{I})$, в которой

- $\mathcal{C}$ — это множество **конфигураций**,

Математическая модель.

Определение 1.

Системой переходов называется тройка $S = (\mathcal{C}, \to, \mathcal{I})$, в которой

- $\mathcal{C}$ — это множество **конфигураций**,
- $\to$ — это двухместное **отношение переходов** на $\mathcal{C}$, и

Математическая модель.

Определение 1.

Системой переходов называется тройка $S = (\mathcal{C}, \rightarrow, \mathcal{I})$, в которой

- $\mathcal{C}$ — это множество **конфигураций**,
- $\rightarrow$ — это двухместное **отношение переходов** на $\mathcal{C}$, и
- $\mathcal{I}$ — это некоторое подмножество $\mathcal{C}$, элементы которого называются **начальными конфигурациями**.

Математическая модель.

Определение 1.

Системой переходов называется тройка $S = (\mathcal{C}, \to, \mathcal{I})$, в которой

- $\mathcal{C}$ — это множество **конфигураций**,
- $\to$ — это двухместное **отношение переходов** на $\mathcal{C}$, и
- $\mathcal{I}$ — это некоторое подмножество $\mathcal{C}$, элементы которого называются **начальными конфигурациями**.

Определение 2.

Выполнением системы переходов S назовем максимальную последовательность $E = (\gamma_0, \gamma_1, \gamma_2, \ldots)$, в которой

Математическая модель.

Определение 1.

Системой переходов называется тройка $S = (\mathcal{C}, \to, \mathcal{I})$, в которой

- $\mathcal{C}$ — это множество **конфигураций**,
- $\to$ — это двухместное **отношение переходов** на $\mathcal{C}$, и
- $\mathcal{I}$ — это некоторое подмножество $\mathcal{C}$, элементы которого называются **начальными конфигурациями**.

Определение 2.

Выполнением системы переходов S назовем максимальную последовательность $E = (\gamma_0, \gamma_1, \gamma_2, \ldots)$, в которой

- $\gamma_0 \in \mathcal{I}$, и

Математическая модель.

Определение 1.

Системой переходов называется тройка $S = (\mathcal{C}, \rightarrow, \mathcal{I})$, в которой

- $\mathcal{C}$ — это множество **конфигураций**,
- $\rightarrow$ — это двухместное **отношение переходов** на $\mathcal{C}$, и
- $\mathcal{I}$ — это некоторое подмножество $\mathcal{C}$, элементы которого называются **начальными конфигурациями**.

Определение 2.

Выполнением системы переходов S назовем максимальную последовательность $E = (\gamma_0, \gamma_1, \gamma_2, \ldots)$, в которой

- $\gamma_0 \in \mathcal{I}$, и
- для каждого i, $i \geq 0$, выполняется $\gamma_i \rightarrow \gamma_{i+1}$.

Математическая модель.

Определение 3.

Конфигурация γ называется

Математическая модель.

Определение 3.

Конфигурация γ называется

- **заключительной**, если не существует конфигурации δ, удовлетворяющей отношению $\gamma \to \delta$;

Математическая модель.

Определение 3.

Конфигурация γ называется

- **заключительной**, если не существует конфигурации δ, удовлетворяющей отношению $\gamma \to \delta$;
- **достижимой из конфигурации** δ (обозначим это $\delta \rightsquigarrow \gamma$), если существует последовательность конфигураций $\gamma_0, \gamma_1, \gamma_2, \ldots, \gamma_k$, в которой $\delta = \gamma_0$, $\gamma_k = \gamma$, и для каждого i, $0 \leq i < k$, выполняется $\gamma_i \to \gamma_{i+1}$.

Математическая модель.

Определение 3.

Конфигурация γ называется

- **заключительной**, если не существует конфигурации δ, удовлетворяющей отношению $\gamma \to \delta$;
- **достижимой из конфигурации** δ (обозначим это $\delta \rightsquigarrow \gamma$), если существует последовательность конфигураций $\gamma_0, \gamma_1, \gamma_2, \ldots, \gamma_k$, в которой $\delta = \gamma_0$, $\gamma_k = \gamma$, и для каждого i, $0 \leq i < k$, выполняется $\gamma_i \to \gamma_{i+1}$.

Замечание.

Математическая модель.

Определение 3.

Конфигурация γ называется

- **заключительной**, если не существует конфигурации δ, удовлетворяющей отношению $\gamma \to \delta$;
- **достижимой из конфигурации** δ (обозначим это $\delta \rightsquigarrow \gamma$), если существует последовательность конфигураций $\gamma_0, \gamma_1, \gamma_2, \ldots, \gamma_k$, в которой $\delta = \gamma_0$, $\gamma_k = \gamma$, и для каждого i, $0 \leq i < k$, выполняется $\gamma_i \to \gamma_{i+1}$.

Замечание.

1. Выполнение $E = (\gamma_0, \gamma_1, \gamma_2, \ldots)$ будем называть **максимальным**, если оно либо является бесконечным, либо оканчивается заключительной конфигурацией.

Математическая модель.

Определение 3.

Конфигурация γ называется

- **заключительной**, если не существует конфигурации δ, удовлетворяющей отношению $\gamma \to \delta$;
- **достижимой из конфигурации** δ (обозначим это $\delta \rightsquigarrow \gamma$), если существует последовательность конфигураций $\gamma_0, \gamma_1, \gamma_2, \ldots, \gamma_k$, в которой $\delta = \gamma_0$, $\gamma_k = \gamma$, и для каждого i, $0 \leq i < k$, выполняется $\gamma_i \to \gamma_{i+1}$.

Замечание.

1. Выполнение $E = (\gamma_0, \gamma_1, \gamma_2, \ldots)$ будем называть **максимальным**, если оно либо является бесконечным, либо оканчивается заключительной конфигурацией.
2. Всякая конфигурация δ, достижимая из некоторой начальной конфигурации, будет называться просто **достижимой**.

Математическая модель.

Распределенная система состоит из совокупности **процессов** и **коммуникационной подсистемы**. Каждый процесс является системой переходов, которая может взаимодействовать с коммуникационной подсистемой. Мы будем придерживаться следующего соглашения о терминологии:

Математическая модель.

Распределенная система состоит из совокупности **процессов** и **коммуникационной подсистемы**. Каждый процесс является системой переходов, которая может взаимодействовать с коммуникационной подсистемой. Мы будем придерживаться следующего соглашения о терминологии:

- Термины «переход» и «конфигурация» будут использоваться применительно ко всей распределенной системе в целом,

Математическая модель.

Распределенная система состоит из совокупности **процессов** и **коммуникационной подсистемы**. Каждый процесс является системой переходов, которая может взаимодействовать с коммуникационной подсистемой. Мы будем придерживаться следующего соглашения о терминологии:

- Термины «переход» и «конфигурация» будут использоваться применительно ко всей распределенной системе в целом,
- Термины «событие» и «состояние», обозначающие равносильные понятия, будут использоваться, если речь идет о процессах системы.

Математическая модель.

Распределенная система состоит из совокупности **процессов** и **коммуникационной подсистемы**. Каждый процесс является системой переходов, которая может взаимодействовать с коммуникационной подсистемой. Мы будем придерживаться следующего соглашения о терминологии:

- Термины «переход» и «конфигурация» будут использоваться применительно ко всей распределенной системе в целом,
- Термины «событие» и «состояние», обозначающие равносильные понятия, будут использоваться, если речь идет о процессах системы.

Чтобы взаимодействовать с коммуникационной подсистемой, в распоряжении процесса помимо **внутренних событий**, имеются также **события отправления** и **события приема**, которые порождают или изымают сообщения из коммуникационной подсистемы.

Системы с асинхронной связью.

Обозначим символом $\mathcal{M}$ множество всех возможных **сообщений**; запись $\mathbb{M}(\mathcal{M})$ будет обозначать совокупность всех мультимножеств с элементами из $\mathcal{M}$.

Системы с асинхронной связью.

Обозначим символом $\mathcal{M}$ множество всех возможных **сообщений**; запись $\mathbb{M}(\mathcal{M})$ будет обозначать совокупность всех мультимножеств с элементами из $\mathcal{M}$.

Определение 4.

Локальным алгоритмом процесса называется пятерка $(Z, I, \vdash^i, \vdash^s, \vdash^r)$, в которой

Системы с асинхронной связью.

Обозначим символом $\mathcal{M}$ множество всех возможных **сообщений**; запись $\mathbb{M}(\mathcal{M})$ будет обозначать совокупность всех мультимножеств с элементами из $\mathcal{M}$.

Определение 4.

Локальным алгоритмом процесса называется пятерка $(Z, I, \vdash^i, \vdash^s, \vdash^r)$, в которой

- Z — это множество **состояний**,

Системы с асинхронной связью.

Обозначим символом $\mathcal{M}$ множество всех возможных **сообщений**; запись $\mathbb{M}(\mathcal{M})$ будет обозначать совокупность всех мультимножеств с элементами из $\mathcal{M}$.

Определение 4.

Локальным алгоритмом процесса называется пятерка $(Z, I, \vdash^i, \vdash^s, \vdash^r)$, в которой

- Z — это множество **состояний**,
- $I \subseteq Z$ — это подмножество **начальных состояний**,

Системы с асинхронной связью.

Обозначим символом $\mathcal{M}$ множество всех возможных **сообщений**; запись $\mathbb{M}(\mathcal{M})$ будет обозначать совокупность всех мультимножеств с элементами из $\mathcal{M}$.

Определение 4.

Локальным алгоритмом процесса называется пятерка $(Z, I, \vdash^i, \vdash^s, \vdash^r)$, в которой

- Z — это множество **состояний**,
- $I \subseteq Z$ — это подмножество **начальных состояний**,
- $\vdash^i$ — отношение **внутреннего действия** на множестве $Z \times Z$,

Системы с асинхронной связью.

Обозначим символом $\mathcal{M}$ множество всех возможных **сообщений**; запись $\mathbb{M}(\mathcal{M})$ будет обозначать совокупность всех мультимножеств с элементами из $\mathcal{M}$.

Определение 4.

Локальным алгоритмом процесса называется пятерка $(Z, I, \vdash^i, \vdash^s, \vdash^r)$, в которой

- Z — это множество **состояний**,
- $I \subseteq Z$ — это подмножество **начальных состояний**,
- $\vdash^i$ — отношение **внутреннего действия** на множестве $Z \times Z$,
- $\vdash^s$ и $\vdash^r$ отношения **отправления** и **приема** сообщений на множестве $Z \times \mathcal{M} \times Z$.

Системы с асинхронной связью.

Обозначим символом $\mathcal{M}$ множество всех возможных **сообщений**; запись $\mathbb{M}(\mathcal{M})$ будет обозначать совокупность всех мультимножеств с элементами из $\mathcal{M}$.

Определение 4.

Локальным алгоритмом процесса называется пятерка $(Z, I, \vdash^i, \vdash^s, \vdash^r)$, в которой

- Z — это множество **состояний**,
- $I \subseteq Z$ — это подмножество **начальных состояний**,
- $\vdash^i$ — отношение **внутреннего действия** на множестве $Z \times Z$,
- $\vdash^s$ и $\vdash^r$ отношения **отправления** и **приема** сообщений на множестве $Z \times \mathcal{M} \times Z$.

Двухместное отношение $\vdash$ на множестве Z определяется соотношением

$$c \vdash d \iff (c, d) \in \vdash^i \ \vee \exists m \in \mathcal{M} \ : \ (c, m, d) \in \vdash^s \cup \vdash^r .$$

Системы с асинхронной связью.

Пример локального алгоритма процесса

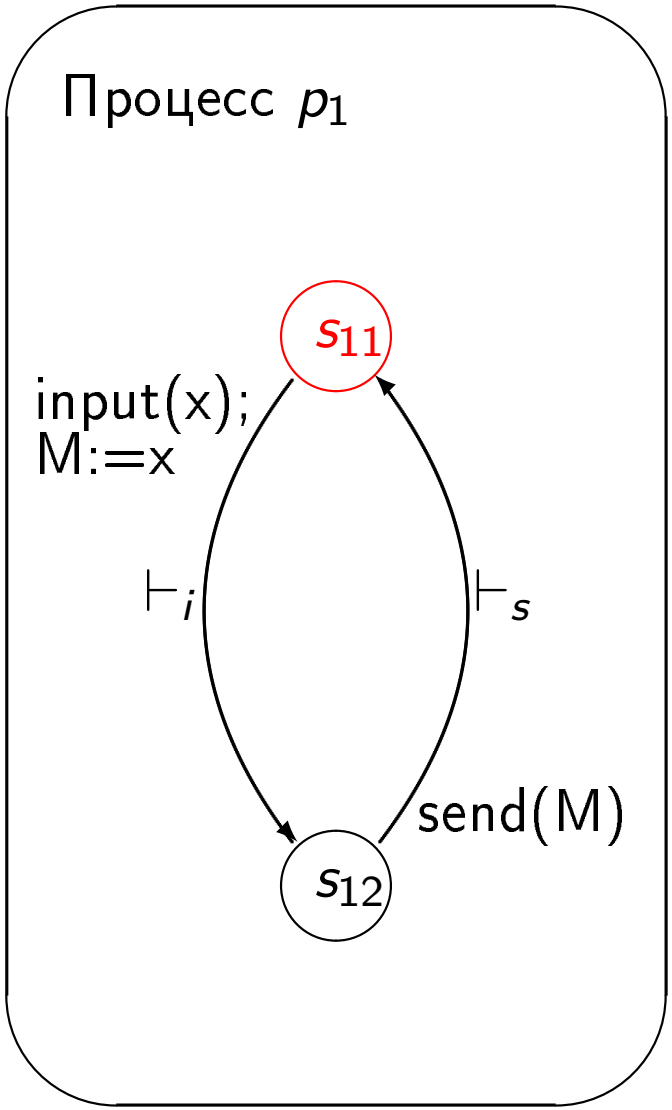
Процесс p_1:

var: x: bool;
var M: bool;

do forever
  begin
    input(x); M:=x;
    send(M)
  end

Системы с асинхронной связью.

Пример локального алгоритма процесса



Системы с асинхронной связью.

Определение 5.

Распределенным алгоритмом для семейства процессов $\mathbb{P} = \{p_1, \ldots, p_N\}$ будет называться совокупность локальных алгоритмов, каждый из которых соответствует в точности одному процессу из $\mathbb{P}$.

Системы с асинхронной связью.

Определение 5.

Распределенным алгоритмом для семейства процессов $\mathbb{P} = \{p_1, \ldots, p_N\}$ будет называться совокупность локальных алгоритмов, каждый из которых соответствует в точности одному процессу из $\mathbb{P}$.

Поведение распределенного алгоритма описывается посредством системы переходов следующего вида. Каждая конфигурация системы определяется состояниями всех процессов, а также совокупностью сообщений, которые еще не достигли адресатов. Переходами являются все события процессов, которые изменяют состояния процессов, оказывают влияние на совокупность сообщений, или претерпевают воздействия этой совокупности сообщений. Начальными конфигурациями считаются все те конфигурации, в которых все процессы пребывают в начальных состояниях, а множество сообщений пусто.

Системы с асинхронной связью.

Определение 6.

Пусть задано семейство процессов $\mathbb{P} = \{p_1, \ldots, p_N\}$, и локальный алгоритм каждого процесса p_i представлен пятеркой $(Z_{p_i}, I_{p_i}, \vdash^i_{p_i}, \vdash^s_{p_i}, \vdash^r_{p_i})$.

Системы с асинхронной связью.

Определение 6.

Пусть задано семейство процессов $\mathbb{P} = \{p_1, \ldots, p_N\}$, и локальный алгоритм каждого процесса p_i представлен пятеркой $(Z_{p_i}, I_{p_i}, \vdash^i_{p_i}, \vdash^s_{p_i}, \vdash^r_{p_i})$. Тогда **система переходов асинхронно взаимосвязанных процессов** $S = (\mathcal{C}, \to, \mathcal{I})$ такова:

Системы с асинхронной связью.

Определение 6.

Пусть задано семейство процессов $\mathbb{P} = \{p_1, \ldots, p_N\}$, и локальный алгоритм каждого процесса p_i представлен пятеркой $(Z_{p_i}, I_{p_i}, \vdash^i_{p_i}, \vdash^s_{p_i}, \vdash^r_{p_i})$. Тогда **система переходов асинхронно взаимосвязанных процессов** $S = (\mathcal{C}, \to, \mathcal{I})$ такова:

1. $\mathcal{C} = \{(c_{p_1}, \ldots, c_{p_N}, M) : (\forall p \in \mathbb{P} : c_p \in Z_p)$ и $M \in \mathbb{M}(\mathcal{M})\}$.

Системы с асинхронной связью.

Определение 6.

Пусть задано семейство процессов $\mathbb{P} = \{p_1, \ldots, p_N\}$, и локальный алгоритм каждого процесса p_i представлен пятеркой $(Z_{p_i}, I_{p_i}, \vdash^i_{p_i}, \vdash^s_{p_i}, \vdash^r_{p_i})$. Тогда **система переходов асинхронно взаимосвязанных процессов** $S = (\mathcal{C}, \to, \mathcal{I})$ такова:

1. $\mathcal{C} = \{(c_{p_1}, \ldots, c_{p_N}, M) : (\forall p \in \mathbb{P} : c_p \in Z_p)$ и $M \in \mathbb{M}(\mathcal{M})\}$.
2. $\to = (\cup_{p \in \mathbb{P}} \to_p)$, где $\to_p$ обозначает переходы процесса p; т. е. $\to_{p_i}$ — это множество всех пар вида $(c_{p_1}, \ldots, c_{p_i}, \ldots, c_{p_N}, M_1)$, $(c_{p_1}, \ldots, c'_{p_i}, \ldots, c_{p_N}, M_2)$ для которых выполняется одно из трех условий:

Системы с асинхронной связью.

Определение 6.

Пусть задано семейство процессов $\mathbb{P} = \{p_1, \ldots, p_N\}$, и локальный алгоритм каждого процесса p_i представлен пятеркой $(Z_{p_i}, I_{p_i}, \vdash^i_{p_i}, \vdash^s_{p_i}, \vdash^r_{p_i})$. Тогда **система переходов асинхронно взаимосвязанных процессов** $S = (\mathcal{C}, \to, \mathcal{I})$ такова:

1. $\mathcal{C} = \{(c_{p_1}, \ldots, c_{p_N}, M) : (\forall p \in \mathbb{P} : c_p \in Z_p)$ и $M \in \mathbb{M}(\mathcal{M})\}$.
2. $\to = (\cup_{p \in \mathbb{P}} \to_p)$, где $\to_p$ обозначает переходы процесса p; т. е. $\to_{p_i}$ — это множество всех пар вида $(c_{p_1}, \ldots, c_{p_i}, \ldots, c_{p_N}, M_1)$, $(c_{p_1}, \ldots, c'_{p_i}, \ldots, c_{p_N}, M_2)$ для которых выполняется одно из трех условий:
   - $(c_{p_i}, c'_{p_i}) \in \vdash^i_{p_i}$ и $M_1 = M_2$;

Системы с асинхронной связью.

Определение 6.

Пусть задано семейство процессов $\mathbb{P} = \{p_1, \ldots, p_N\}$, и локальный алгоритм каждого процесса p_i представлен пятеркой $(Z_{p_i}, I_{p_i}, \vdash^i_{p_i}, \vdash^s_{p_i}, \vdash^r_{p_i})$. Тогда **система переходов асинхронно взаимосвязанных процессов** $S = (\mathcal{C}, \to, \mathcal{I})$ такова:

1. $\mathcal{C} = \{(c_{p_1}, \ldots, c_{p_N}, M) : (\forall p \in \mathbb{P} : c_p \in Z_p)$ и $M \in \mathbb{M}(\mathcal{M})\}$.
2. $\to = (\cup_{p \in \mathbb{P}} \to_p)$, где $\to_p$ обозначает переходы процесса p; т. е. $\to_{p_i}$ — это множество всех пар вида $(c_{p_1}, \ldots, c_{p_i}, \ldots, c_{p_N}, M_1)$, $(c_{p_1}, \ldots, c'_{p_i}, \ldots, c_{p_N}, M_2)$ для которых выполняется одно из трех условий:
   - $(c_{p_i}, c'_{p_i}) \in \vdash^i_{p_i}$ и $M_1 = M_2$;
   - для некоторого сообщения $m \in \mathcal{M}$ имеется событие $(c_{p_i}, m, c'_{p_i}) \in \vdash^s_{p_i}$, и при этом $M_2 = M_1 \cup \{m\}$;

Системы с асинхронной связью.

Определение 6.

Пусть задано семейство процессов $\mathbb{P} = \{p_1, \ldots, p_N\}$, и локальный алгоритм каждого процесса p_i представлен пятеркой $(Z_{p_i}, I_{p_i}, \vdash^i_{p_i}, \vdash^s_{p_i}, \vdash^r_{p_i})$. Тогда **система переходов асинхронно взаимосвязанных процессов** $S = (\mathcal{C}, \to, \mathcal{I})$ такова:

1. $\mathcal{C} = \{(c_{p_1}, \ldots, c_{p_N}, M) : (\forall p \in \mathbb{P} : c_p \in Z_p)$ и $M \in \mathbb{M}(\mathcal{M})\}$.
2. $\to = (\cup_{p \in \mathbb{P}} \to_p)$, где $\to_p$ обозначает переходы процесса p; т. е. $\to_{p_i}$ — это множество всех пар вида $(c_{p_1}, \ldots, c_{p_i}, \ldots, c_{p_N}, M_1)$, $(c_{p_1}, \ldots, c'_{p_i}, \ldots, c_{p_N}, M_2)$ для которых выполняется одно из трех условий:
   - $(c_{p_i}, c'_{p_i}) \in \vdash^i_{p_i}$ и $M_1 = M_2$;
   - для некоторого сообщения $m \in \mathcal{M}$ имеется событие $(c_{p_i}, m, c'_{p_i}) \in \vdash^s_{p_i}$, и при этом $M_2 = M_1 \cup \{m\}$;
   - для некоторого сообщения $m \in \mathcal{M}$ имеется событие $(c_{p_i}, m, c'_{p_i}) \in \vdash^r_{p_i}$, и при этом $M_1 = M_2 \cup \{m\}$.

Системы с асинхронной связью.

Определение 6.

Пусть задано семейство процессов $\mathbb{P} = \{p_1, \ldots, p_N\}$, и локальный алгоритм каждого процесса p_i представлен пятеркой $(Z_{p_i}, I_{p_i}, \vdash^i_{p_i}, \vdash^s_{p_i}, \vdash^r_{p_i})$. Тогда **система переходов асинхронно взаимосвязанных процессов** $S = (\mathcal{C}, \to, \mathcal{I})$ такова:

1. $\mathcal{C} = \{(c_{p_1}, \ldots, c_{p_N}, M) : (\forall p \in \mathbb{P} : c_p \in Z_p)$ и $M \in \mathbb{M}(\mathcal{M})\}$.
2. $\to = (\cup_{p \in \mathbb{P}} \to_p)$, где $\to_p$ обозначает переходы процесса p; т. е. $\to_{p_i}$ — это множество всех пар вида $(c_{p_1}, \ldots, c_{p_i}, \ldots, c_{p_N}, M_1)$, $(c_{p_1}, \ldots, c'_{p_i}, \ldots, c_{p_N}, M_2)$ для которых выполняется одно из трех условий:
   - $(c_{p_i}, c'_{p_i}) \in \vdash^i_{p_i}$ и $M_1 = M_2$;
   - для некоторого сообщения $m \in \mathcal{M}$ имеется событие $(c_{p_i}, m, c'_{p_i}) \in \vdash^s_{p_i}$, и при этом $M_2 = M_1 \cup \{m\}$;
   - для некоторого сообщения $m \in \mathcal{M}$ имеется событие $(c_{p_i}, m, c'_{p_i}) \in \vdash^r_{p_i}$, и при этом $M_1 = M_2 \cup \{m\}$.
3. $\mathcal{I} = \{(c_{p_1}, \ldots, c_{p_N}, M) : (\forall p \in \mathbb{P} : c_p \in I_p) \wedge M = \varnothing\}$.

Системы с асинхронной связью.

Замечания

Пары $(c, d) \in \vdash_p^i$ назовем **внутренними событиями** процесса p,

Системы с асинхронной связью.

Замечания

Пары $(c, d) \in \vdash_p^i$ назовем **внутренними событиями** процесса p, а тройки, из которых состоят отношения $\vdash_p^s$ и $\vdash_p^r$, назовем **событиями отправления** и **приема** .

Системы с асинхронной связью.

Замечания

Пары $(c, d) \in \vdash_p^i$ назовем **внутренними событиями** процесса p, а тройки, из которых состоят отношения $\vdash_p^s$ и $\vdash_p^r$, назовем **событиями отправления** и **приема** .

Пример распределенной системы

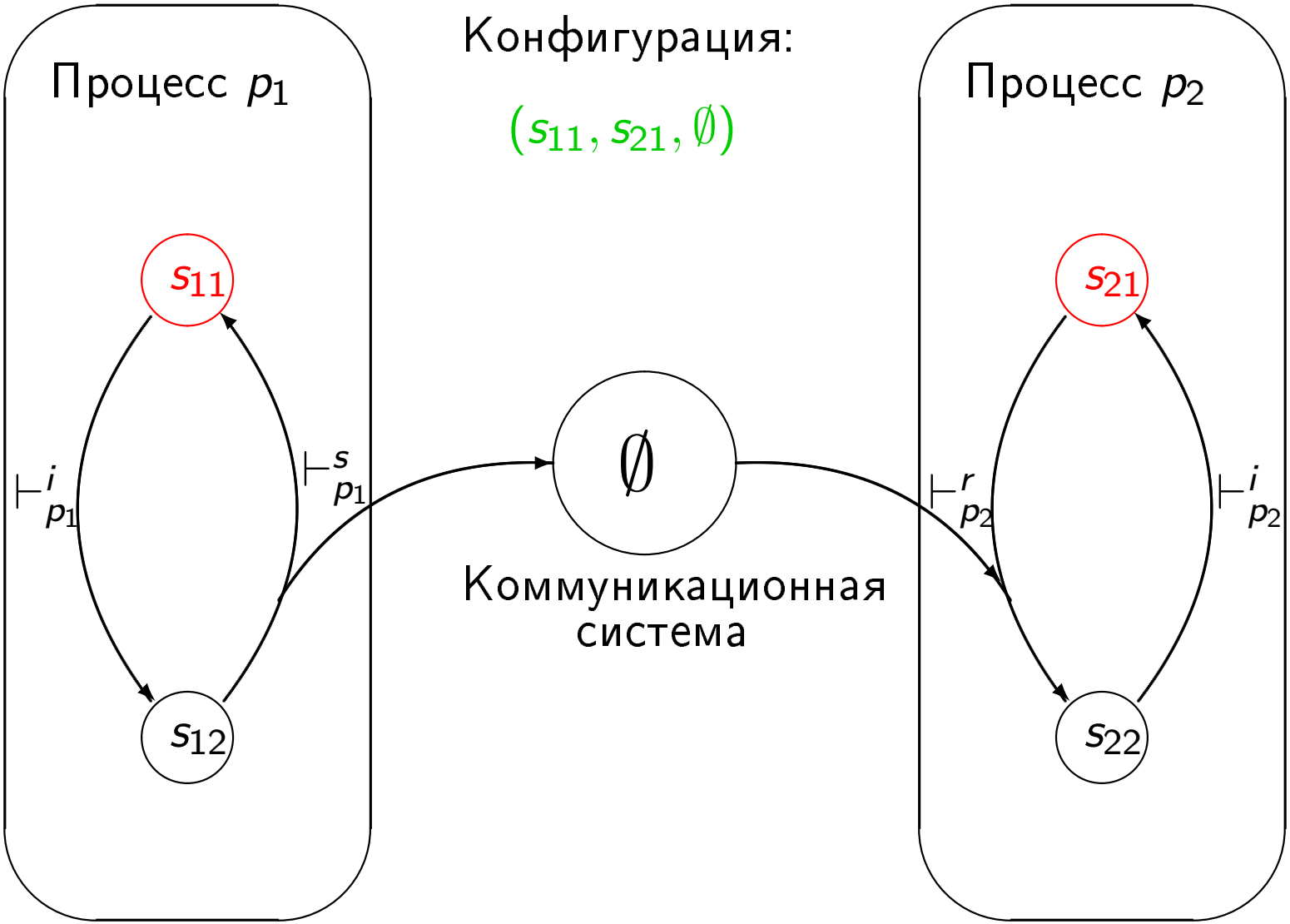
Процесс p_1:
var: x: bool;
var M: bool;

do forever
  begin
    input(x); M:=x;
    send(M)
  end

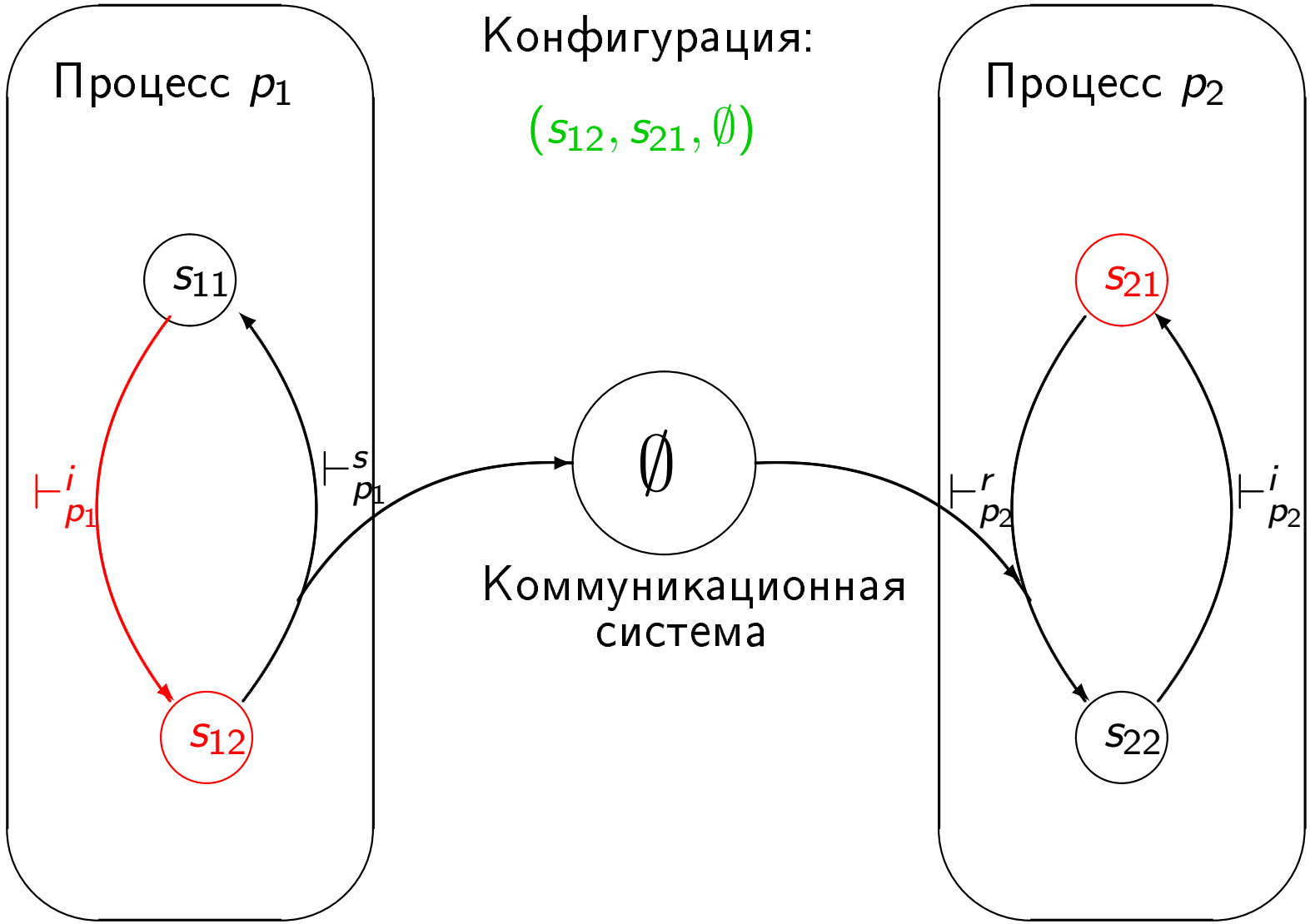
Процесс p_2:
var: y: bool;
var m: bool;

do forever
  begin
    receive(m);
    y:=m; output(y)
  end

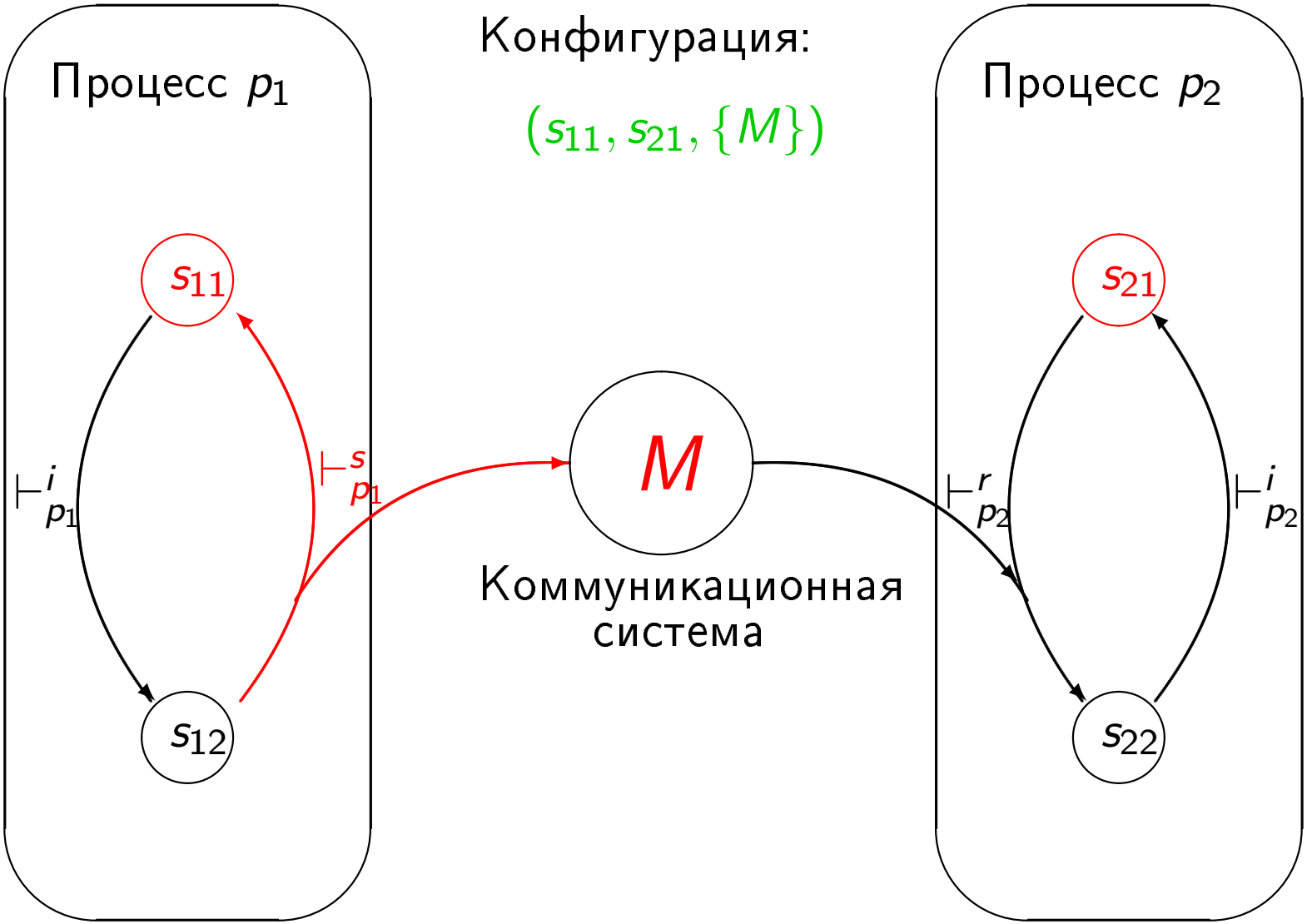
Системы с асинхронной связью.



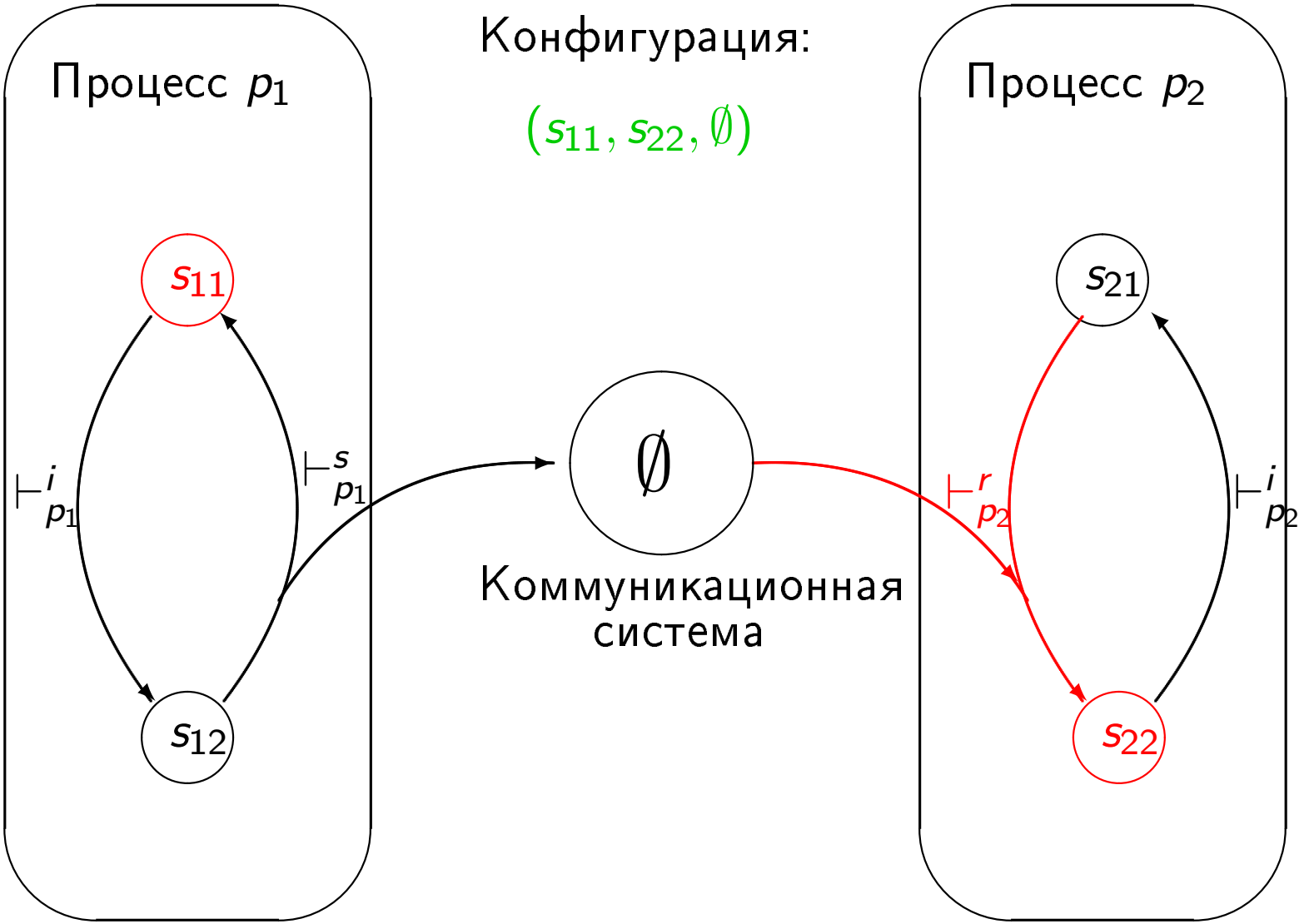
Системы с асинхронной связью.



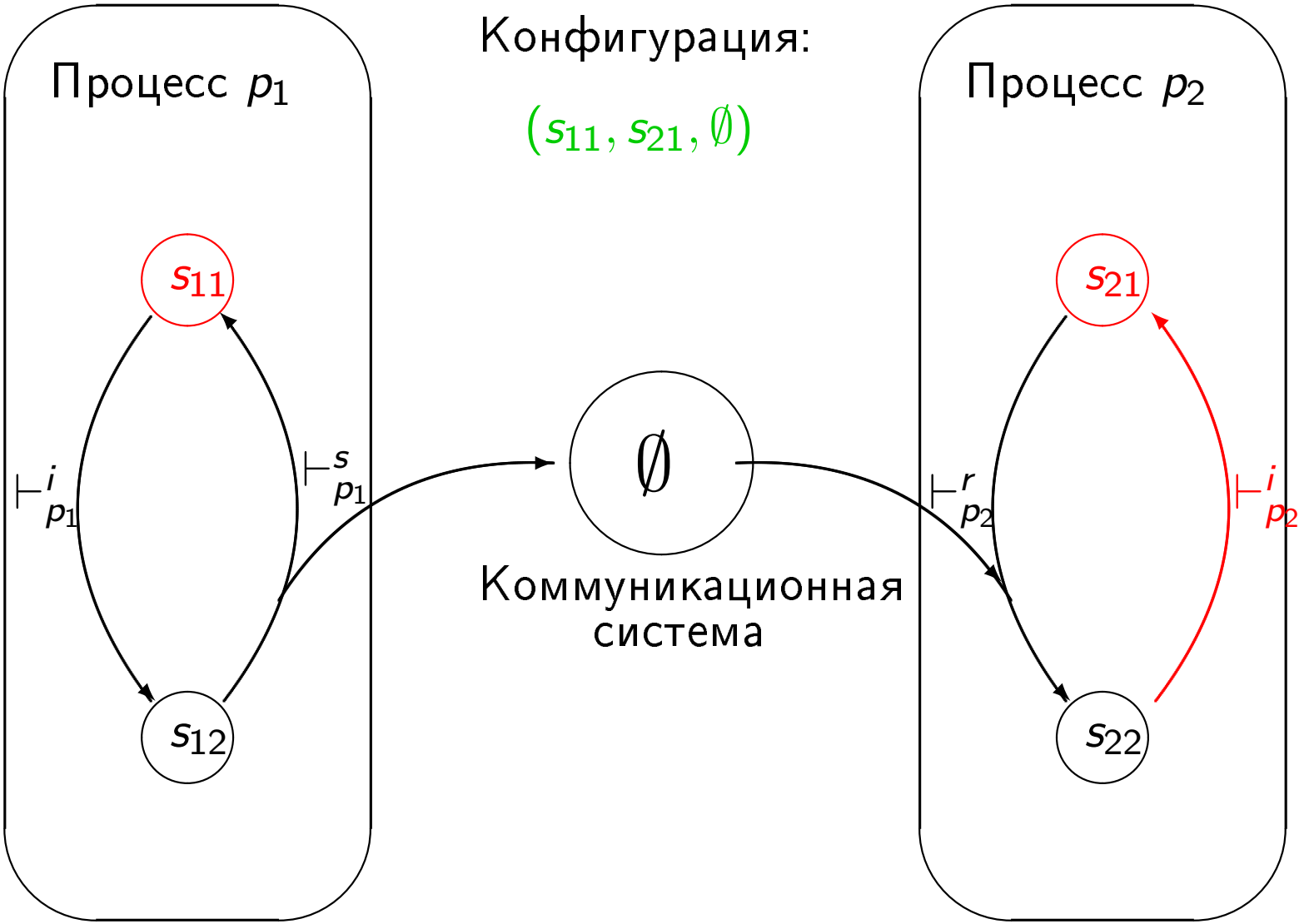
Системы с асинхронной связью.



Системы с асинхронной связью.



Системы с асинхронной связью.



Системы с асинхронной связью.

Вопросы

1. Какие еще выполнения возможны для этой распределенной системы?
2. Если другие выполнения есть, то насколько значительно они отличаются от друг от друга?

Системы с асинхронной связью.

Замечания

- Внутреннее событие $e = (c, d)$ процесса p **допустимо** в конфигурации $\gamma = (c_{p_1}, \ldots, c_p, \ldots, c_{p_N}, M)$, если $c_p = c$. Тогда $e(\gamma) = (c_{p_1}, \ldots, d, \ldots, c_{p_N}, M)$.

Системы с асинхронной связью.

Замечания

- Внутреннее событие $e = (c,\ d)$ процесса p **допустимо** в конфигурации $\gamma = (c_{p_1},\ \ldots,\ c_p,\ \ldots,\ c_{p_N},\ M)$, если $c_p = c$. Тогда $e(\gamma) = (c_{p_1},\ \ldots,\ d,\ \ldots,\ c_{p_N},\ M)$.
- Событие $e = (c,\ m,\ d)$ отправления сообщения процессом p **допустимо** в конфигурации $\gamma = (c_{p_1},\ \ldots,\ c_p,\ \ldots,\ c_{p_N},\ M)$, если $c_p = c$. Тогда $e(\gamma) = (c_{p_1},\ \ldots,\ d,\ \ldots,\ c_{p_N},\ M \cup \{m\})$.

Системы с асинхронной связью.

Замечания

- Внутреннее событие $e = (c,\ d)$ процесса p **допустимо** в конфигурации $\gamma = (c_{p_1},\ \ldots,\ c_p,\ \ldots,\ c_{p_N},\ M)$, если $c_p = c$. Тогда $e(\gamma) = (c_{p_1},\ \ldots,\ d,\ \ldots,\ c_{p_N},\ M)$.
- Событие $e = (c,\ m,\ d)$ отправления сообщения процессом p **допустимо** в конфигурации $\gamma = (c_{p_1},\ \ldots,\ c_p,\ \ldots,\ c_{p_N},\ M)$, если $c_p = c$. Тогда $e(\gamma) = (c_{p_1},\ \ldots,\ d,\ \ldots,\ c_{p_N},\ M \cup \{m\})$.
- Событие $e = (c,\ m,\ d)$ приема сообщения процессом p **допустимо** в конфигурации $\gamma = (c_{p_1},\ \ldots,\ c_p,\ \ldots,\ c_{p_N},\ M)$, если $c_p = c$, $m \in M$. Тогда $e(\gamma) = (c_{p_1}, \ldots, d, \ldots, c_{p_N}, M \setminus \{m\})$.

Системы с асинхронной связью.

Замечания

- Внутреннее событие $e = (c, d)$ процесса p **допустимо** в конфигурации $\gamma = (c_{p_1}, \ldots, c_p, \ldots, c_{p_N}, M)$, если $c_p = c$. Тогда $e(\gamma) = (c_{p_1}, \ldots, d, \ldots, c_{p_N}, M)$.
- Событие $e = (c, m, d)$ отправления сообщения процессом p **допустимо** в конфигурации $\gamma = (c_{p_1}, \ldots, c_p, \ldots, c_{p_N}, M)$, если $c_p = c$. Тогда $e(\gamma) = (c_{p_1}, \ldots, d, \ldots, c_{p_N}, M \cup \{m\})$.
- Событие $e = (c, m, d)$ приема сообщения процессом p **допустимо** в конфигурации $\gamma = (c_{p_1}, \ldots, c_p, \ldots, c_{p_N}, M)$, если $c_p = c$, $m \in M$. Тогда $e(\gamma) = (c_{p_1}, \ldots, d, \ldots, c_{p_N}, M \setminus \{m\})$.

Соглашение: для каждого сообщения имеется единственный процесс, который может получать это сообщение. Этот процесс будем называть **адресатом** данного сообщения.

Системы с синхронной связью.

Обмен сообщениями называется **синхронным** , если событие отправления сообщения и соответствующее ему событие приема сообщения согласованы так, что образуют единый переход в системе. Иначе говоря, процессу не дозволено отправлять сообщение до тех пор, пока адресат этого сообщения не будет готов к его приему.

Системы с синхронной связью.

Обмен сообщениями называется **синхронным** , если событие отправления сообщения и соответствующее ему событие приема сообщения согласованы так, что образуют единый переход в системе. Иначе говоря, процессу не дозволено отправлять сообщение до тех пор, пока адресат этого сообщения не будет готов к его приему.
В соответствии с этим переходы в системе разделяются на два типа: переходы первого типа связаны с изменением внутренних состояний процесса, а переходы второго типа связаны с комбинированным осуществлением событий отправления-приема сообщения двумя процессами.

Системы с синхронной связью.

Определение 7.

Пусть задано семейство процессов $\mathbb{P} = \{p_1, \ldots, p_N\}$. Тогда **система переходов синхронно связанных процессов** $S = (\mathcal{C}, \rightarrow, \mathcal{I})$ такова:

Системы с синхронной связью.

Определение 7.

Пусть задано семейство процессов $\mathbb{P} = \{p_1, \ldots, p_N\}$. Тогда **система переходов синхронно связанных процессов** $S = (\mathcal{C}, \to, \mathcal{I})$ такова:

1. $\mathcal{C} = \{(c_{p_1}, \ldots, c_{p_N}) : \forall p \in \mathbb{P} : c_p \in Z_p\}$.

Системы с синхронной связью.

Определение 7.

Пусть задано семейство процессов $\mathbb{P} = \{p_1, \ldots, p_N\}$. Тогда **система переходов синхронно связанных процессов** $S = (\mathcal{C}, \to, \mathcal{I})$ такова:

1. $\mathcal{C} = \{(c_{p_1}, \ldots, c_{p_N}) : \forall p \in \mathbb{P} : c_p \in Z_p\}$.
2. $\to = \left(\cup_{p \in \mathbb{P}} \to_p\right) \cup \left(\cup_{p,q \in \mathbb{P}:\, p \neq q} \to_{pq}\right)$, где

Системы с синхронной связью.

Определение 7.

Пусть задано семейство процессов $\mathbb{P} = \{p_1, \ldots, p_N\}$. Тогда **система переходов синхронно связанных процессов** $S = (\mathcal{C}, \rightarrow, \mathcal{I})$ такова:

1. $\mathcal{C} = \{(c_{p_1}, \ldots, c_{p_N}) : \forall p \in \mathbb{P} : c_p \in Z_p\}$.
2. $\rightarrow = (\cup_{p \in \mathbb{P}} \rightarrow_p) \cup (\cup_{p,q \in \mathbb{P}:\, p \neq q} \rightarrow_{pq})$, где
   - $\rightarrow_{p_i}$ представляет собой множество пар $(c_{p_1}, \ldots, c_{p_i}, \ldots, c_{p_N})$, $(c_{p_1}, \ldots, c'_{p_i}, \ldots, c_{p_N})$, для которых $(c_{p_i}, c'_{p_i}) \in \vdash^i_{p_i}$;

Системы с синхронной связью.

Определение 7.

Пусть задано семейство процессов $\mathbb{P} = \{p_1, \ldots, p_N\}$. Тогда **система переходов синхронно связанных процессов** $S = (\mathcal{C}, \to, \mathcal{I})$ такова:

1. $\mathcal{C} = \{(c_{p_1}, \ldots, c_{p_N}) : \forall p \in \mathbb{P} : c_p \in Z_p\}$.
2. $\to = (\cup_{p \in \mathbb{P}} \to_p) \cup (\cup_{p,q \in \mathbb{P}:\, p \neq q} \to_{pq})$, где
   - $\to_{p_i}$ представляет собой множество пар $(c_{p_1}, \ldots, c_{p_i}, \ldots, c_{p_N})$, $(c_{p_1}, \ldots, c'_{p_i}, \ldots, c_{p_N})$, для которых $(c_{p_i}, c'_{p_i}) \in \vdash^i_{p_i}$;
   - $\to_{p_i p_j}$ представляет собой множество пар $(\ldots, c_{p_i}, \ldots, c_{p_j}, \ldots)$, $(\ldots, c'_{p_i}, \ldots, c'_{p_j}, \ldots)$, для которых есть такое сообщение $m \in \mathcal{M}$, что $(c_{p_i}, m, c'_{p_i}) \in \vdash^s_{p_i}$ и $(c_{p_j}, m, c'_{p_j}) \in \vdash^r_{p_j}$.

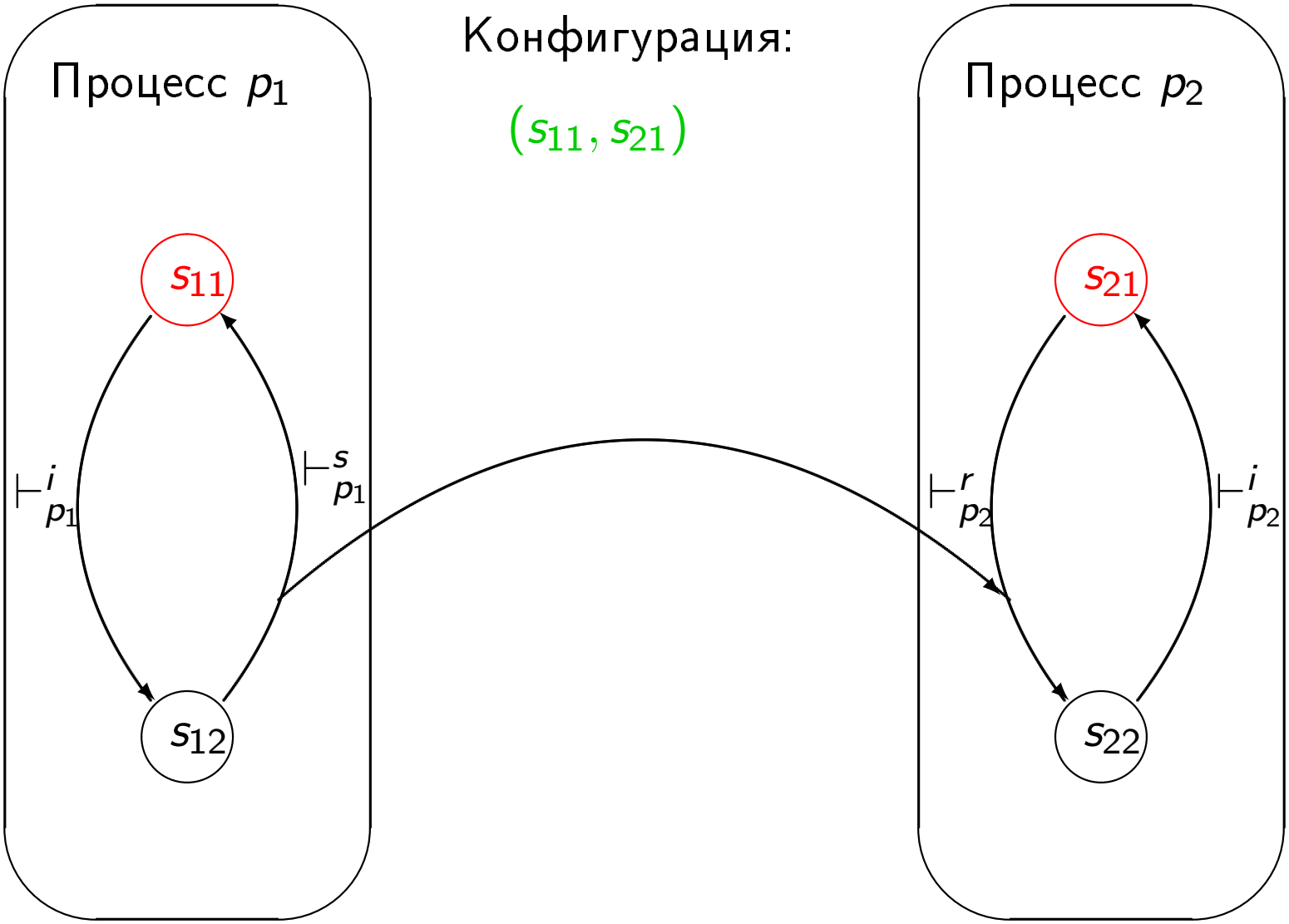
Системы с синхронной связью.

Определение 7.

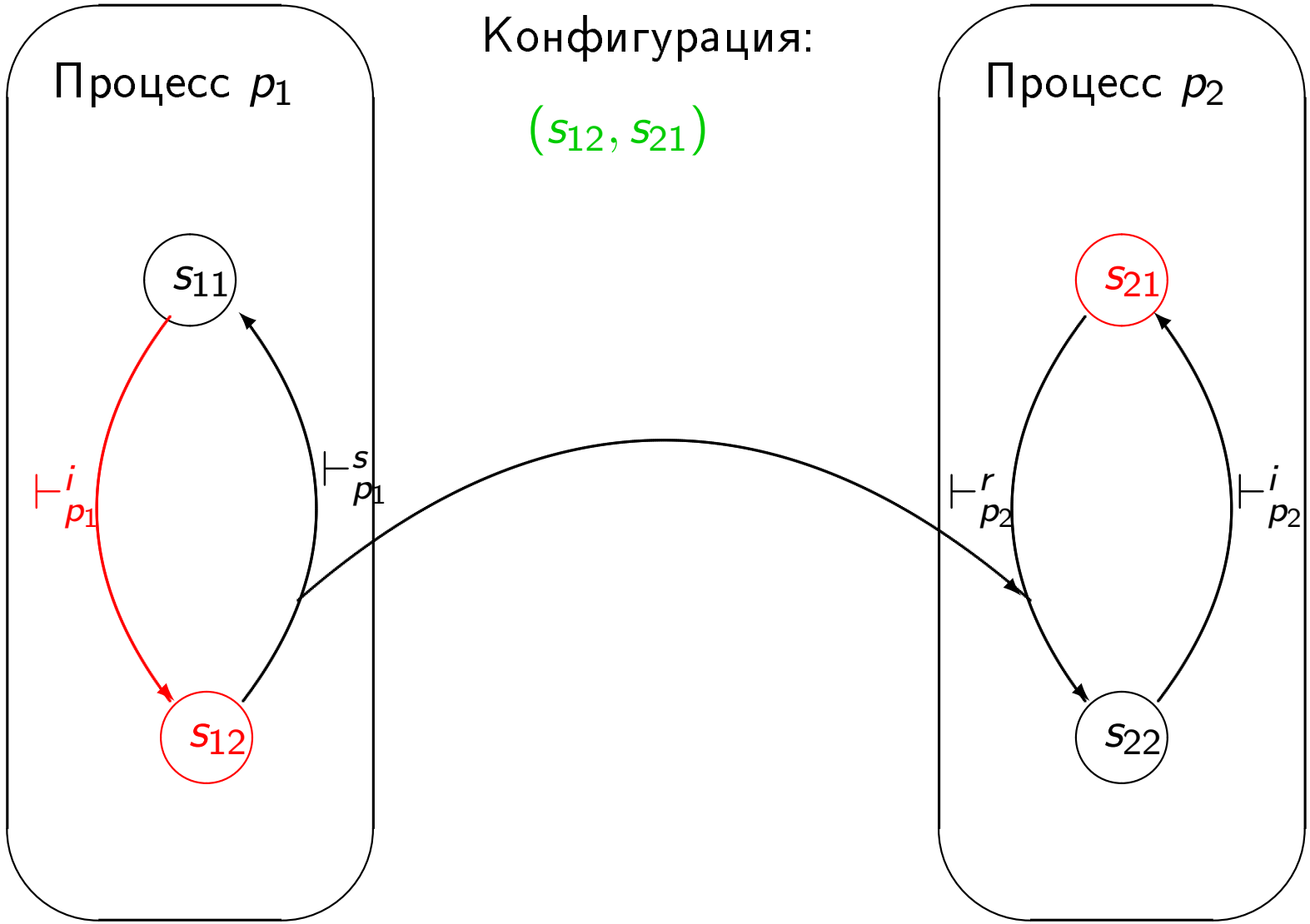
Пусть задано семейство процессов $\mathbb{P} = \{p_1, \ldots, p_N\}$. Тогда **система переходов синхронно связанных процессов** $S = (\mathcal{C}, \to, \mathcal{I})$ такова:

1. $\mathcal{C} = \{(c_{p_1}, \ldots, c_{p_N}) : \forall p \in \mathbb{P} : c_p \in Z_p\}$.
2. $\to = (\cup_{p \in \mathbb{P}} \to_p) \cup (\cup_{p,q \in \mathbb{P}:\, p \neq q} \to_{pq})$, где
   - $\to_{p_i}$ представляет собой множество пар $(c_{p_1}, \ldots, c_{p_i}, \ldots, c_{p_N})$, $(c_{p_1}, \ldots, c'_{p_i}, \ldots, c_{p_N})$, для которых $(c_{p_i}, c'_{p_i}) \in \vdash^i_{p_i}$;
   - $\to_{p_i p_j}$ представляет собой множество пар $(\ldots, c_{p_i}, \ldots, c_{p_j}, \ldots)$, $(\ldots, c'_{p_i}, \ldots, c'_{p_j}, \ldots)$, для которых есть такое сообщение $m \in \mathcal{M}$, что $(c_{p_i}, m, c'_{p_i}) \in \vdash^s_{p_i}$ и $(c_{p_j}, m, c'_{p_j}) \in \vdash^r_{p_j}$.
3. $\mathcal{I} = \{(c_{p_1}, \ldots, c_{p_N}) : (\forall p \in \mathbb{P} : c_p \in I_p)\}$.

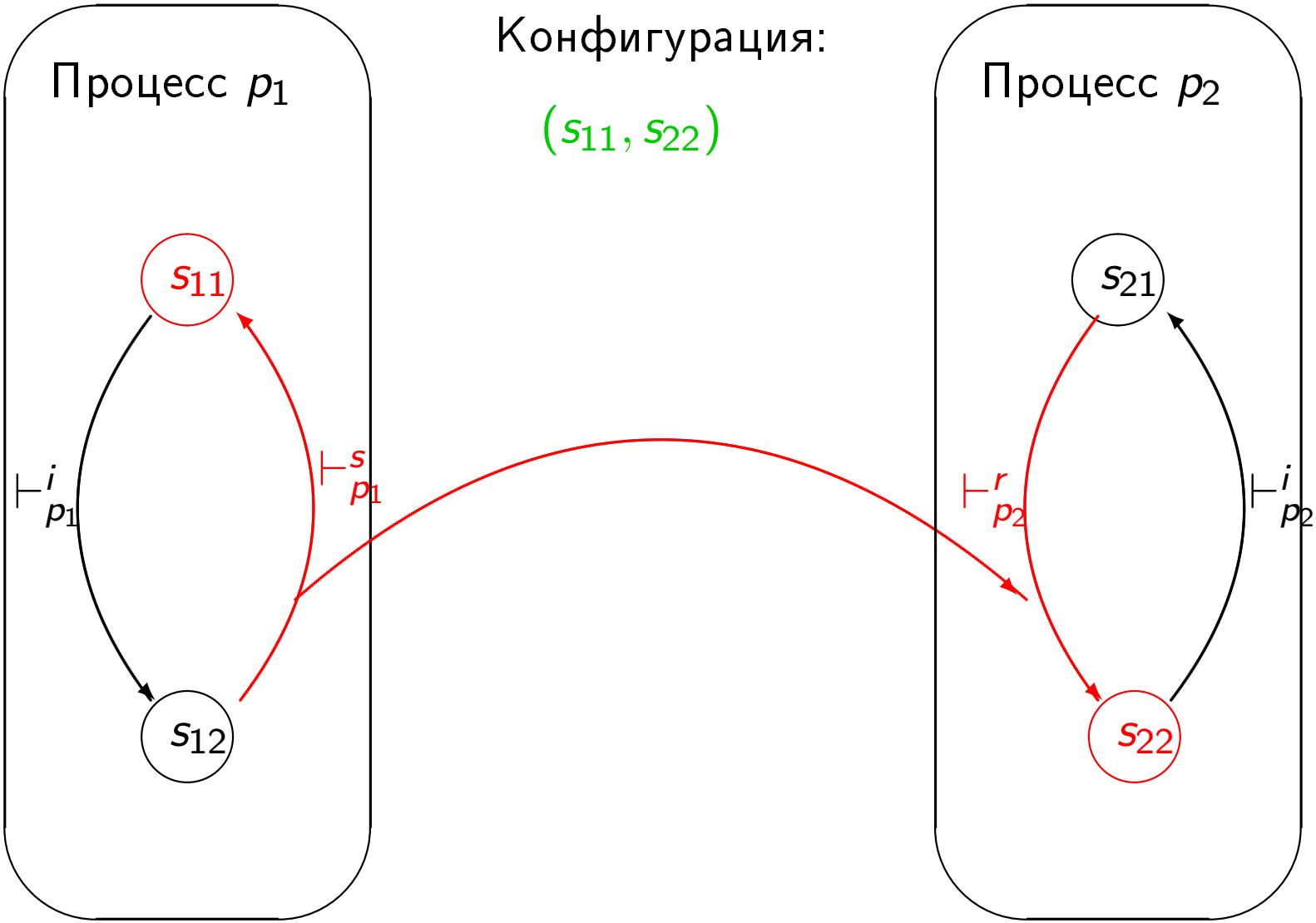
Системы с синхронной связью.



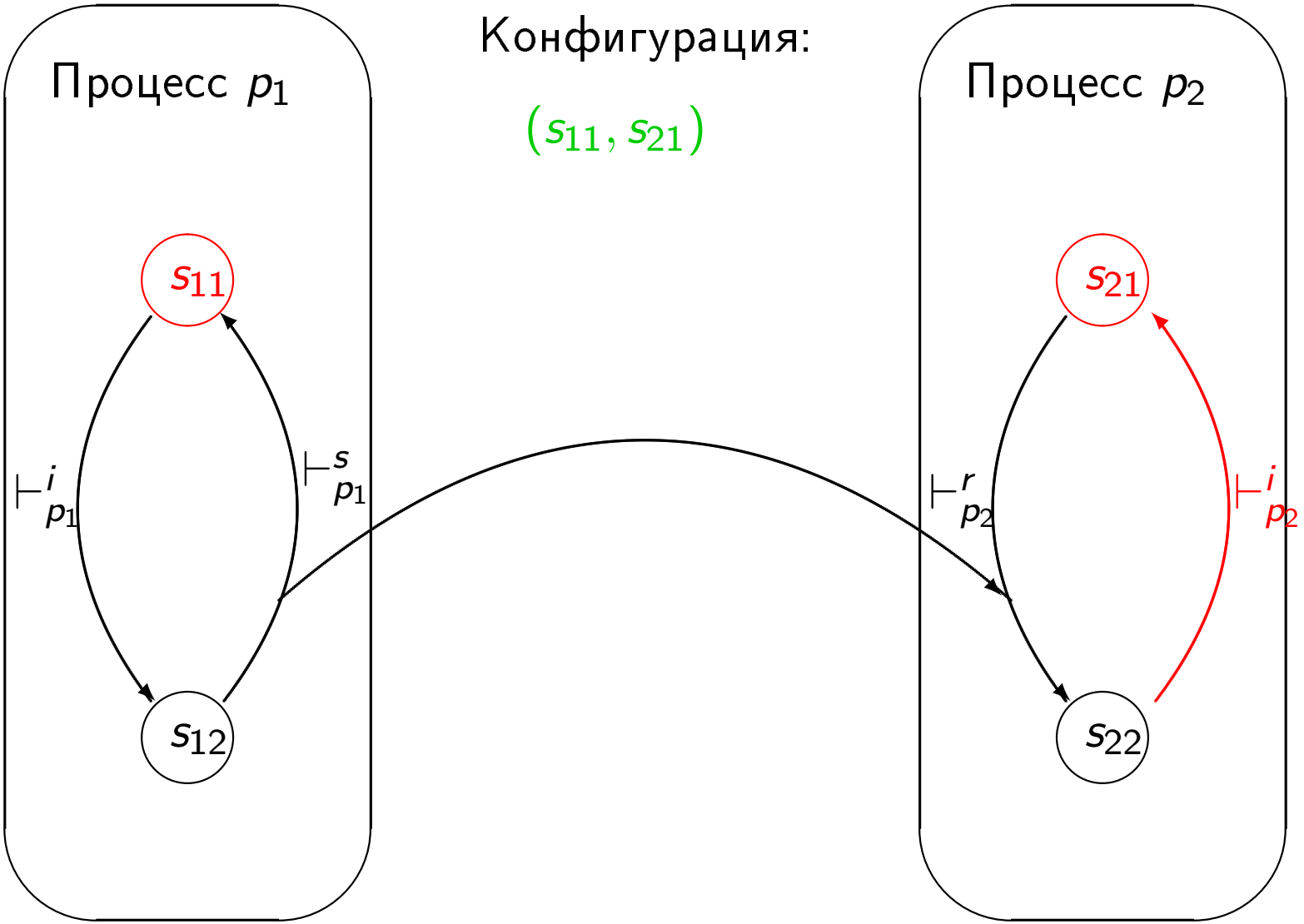
Системы с синхронной связью.



Системы с синхронной связью.



Системы с синхронной связью.



Справедливость.

Иногда при исследовании поведения систем возникает необходимость ограничиться рассмотрением только так называемых **справедливых** выполнений. Условия справедливости позволяют исключить из рассмотрения такие выполнения, в которых некоторые события оказываются допустимыми всегда (или бесконечно часто), но при этом ни разу не осуществляются в виде переходов (из-за того, что выполнение продолжается всякий раз за счет осуществления других событий).

Справедливость.

Иногда при исследовании поведения систем возникает необходимость ограничиться рассмотрением только так называемых **справедливых** выполнений. Условия справедливости позволяют исключить из рассмотрения такие выполнения, в которых некоторые события оказываются допустимыми всегда (или бесконечно часто), но при этом ни разу не осуществляются в виде переходов (из-за того, что выполнение продолжается всякий раз за счет осуществления других событий).

Имеются две основные формы справедливости: **слабая справедливость** и **сильная справедливость**.

Справедливость.

Определение 8.

1. Каждое выполнение, оканчивающееся заключительной конфигурацией, является справедливым (как в слабом, так и в сильном смысле).

Справедливость.

Определение 8.

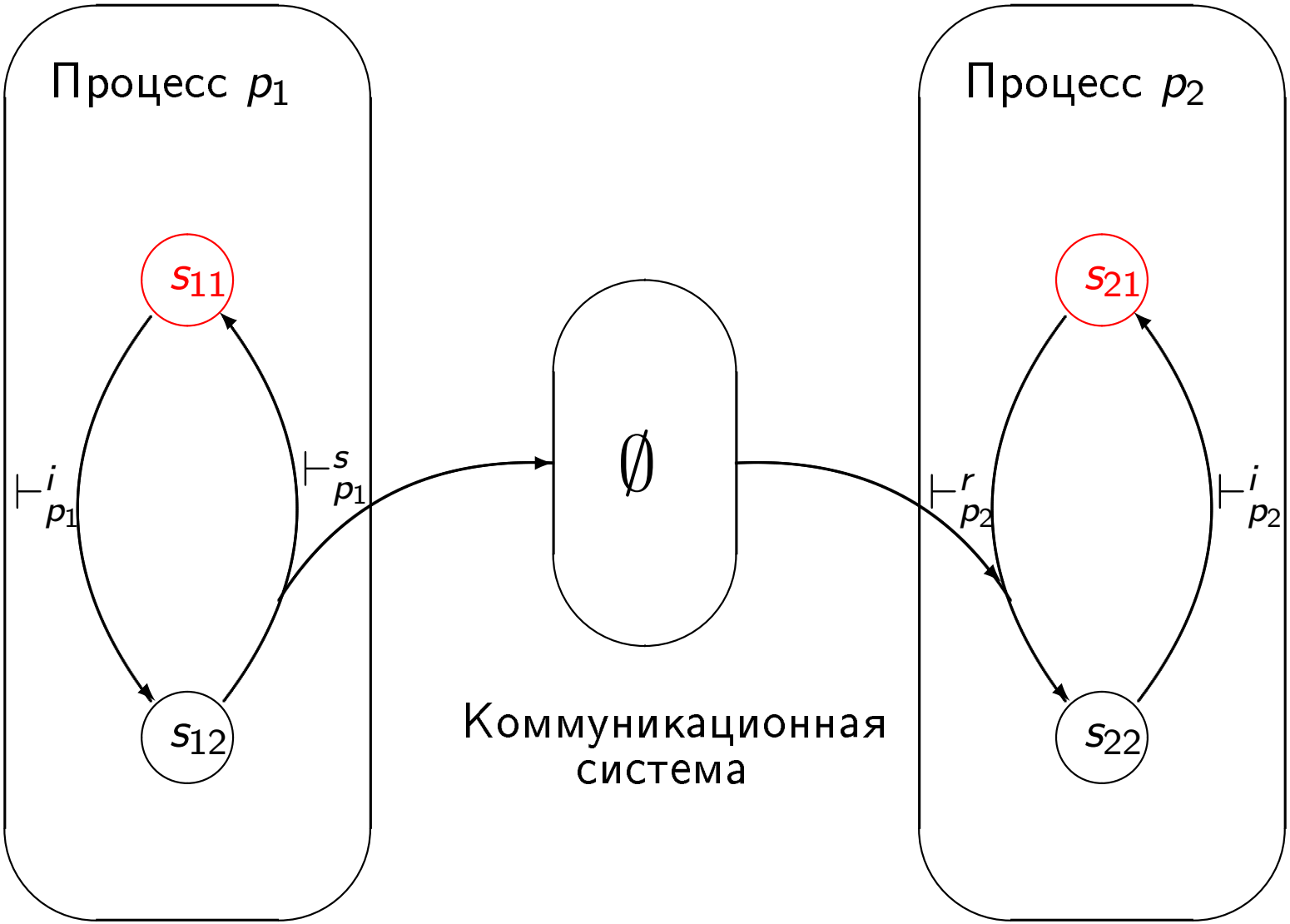
1. Каждое выполнение, оканчивающееся заключительной конфигурацией, является справедливым (как в слабом, так и в сильном смысле).
2. Бесконечное выполнение $E = (\gamma_0, \gamma_1, \gamma_2, \ldots)$ считается **слабо справедливым**, если не существует такого натурального числа n, $n > 0$, и такого события e, что для любого i, $i \geq n$, событие e допустимо в конфигурации γ_i, но при этом $\gamma_{i+1} \neq e(\gamma_i)$.

Справедливость.

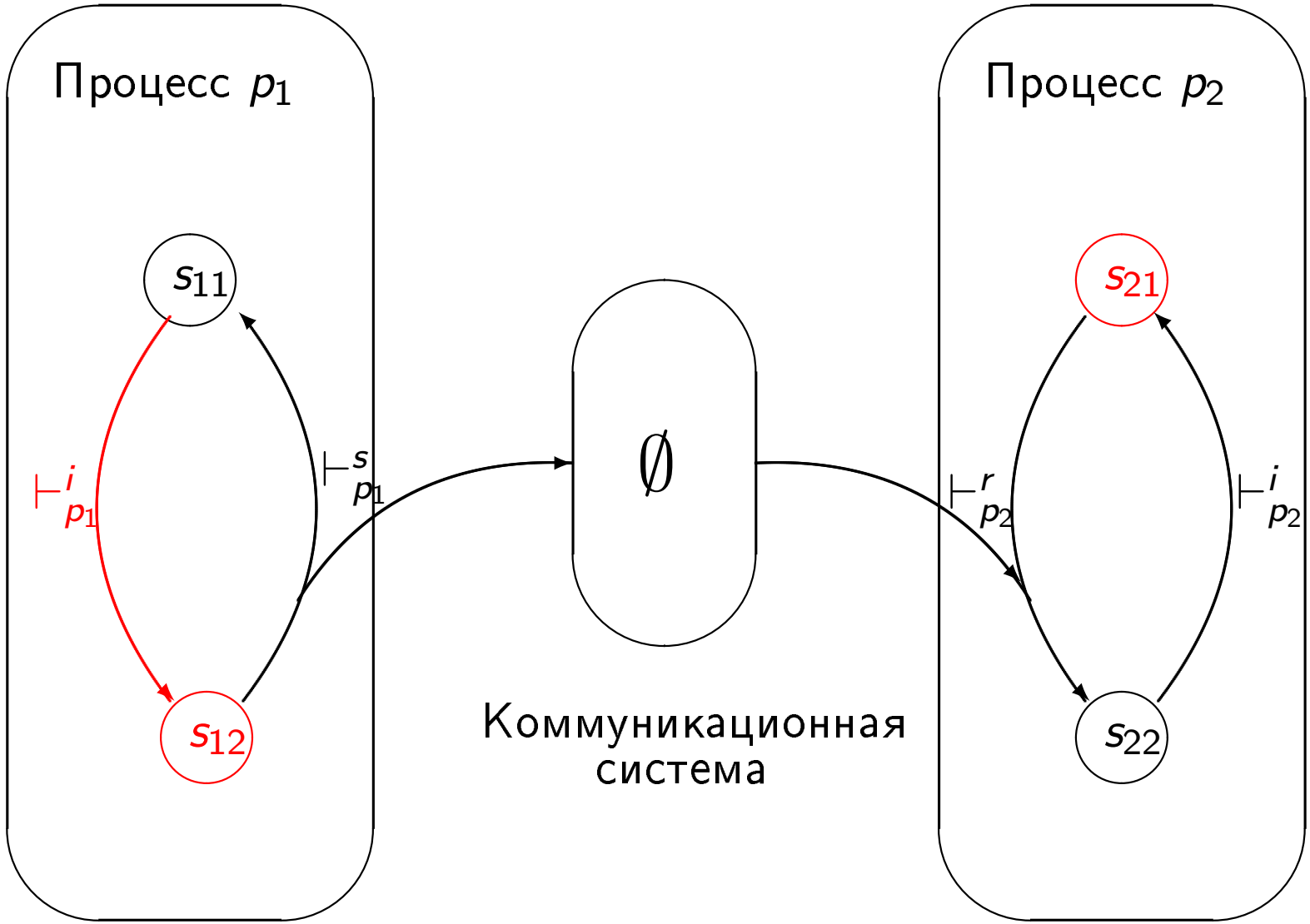
Определение 8.

1. Каждое выполнение, оканчивающееся заключительной конфигурацией, является справедливым (как в слабом, так и в сильном смысле).
2. Бесконечное выполнение $E = (\gamma_0, \gamma_1, \gamma_2, \ldots)$ считается **слабо справедливым**, если не существует такого натурального числа n, $n > 0$, и такого события e, что для любого i, $i \geq n$, событие e допустимо в конфигурации γ_i, но при этом $\gamma_{i+1} \neq e(\gamma_i)$.
3. Бесконечное выполнение $E = (\gamma_0, \gamma_1, \gamma_2, \ldots)$ считается **сильно справедливым**, если не существует такого натурального числа n, $n > 0$, и такого события e, что для любого i, $i \geq n$, событие e допустимо в некоторой конфигурации γ_j, где $j \geq i$, но при этом $\gamma_{i+1} \neq e(\gamma_i)$.

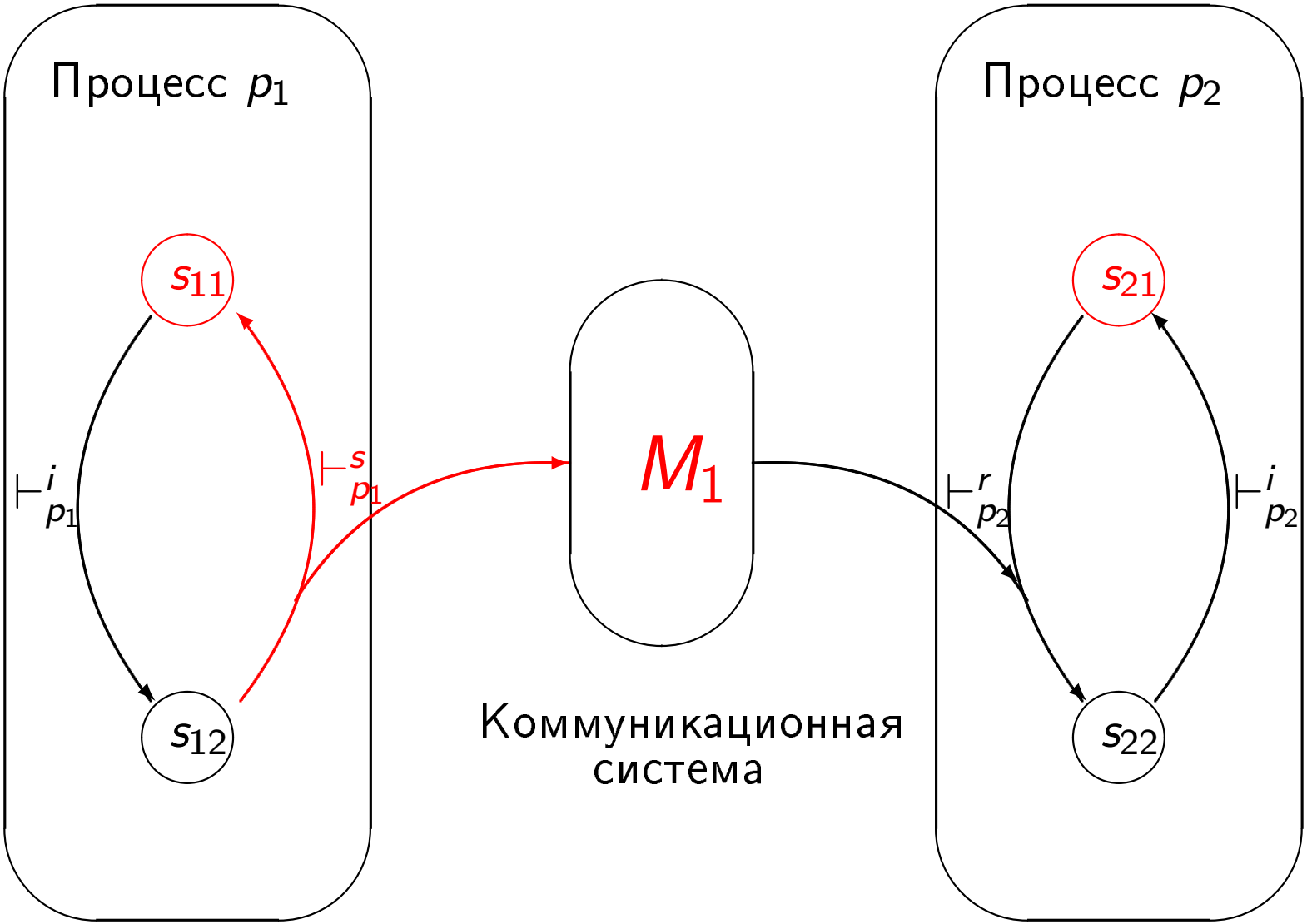
Справедливость.



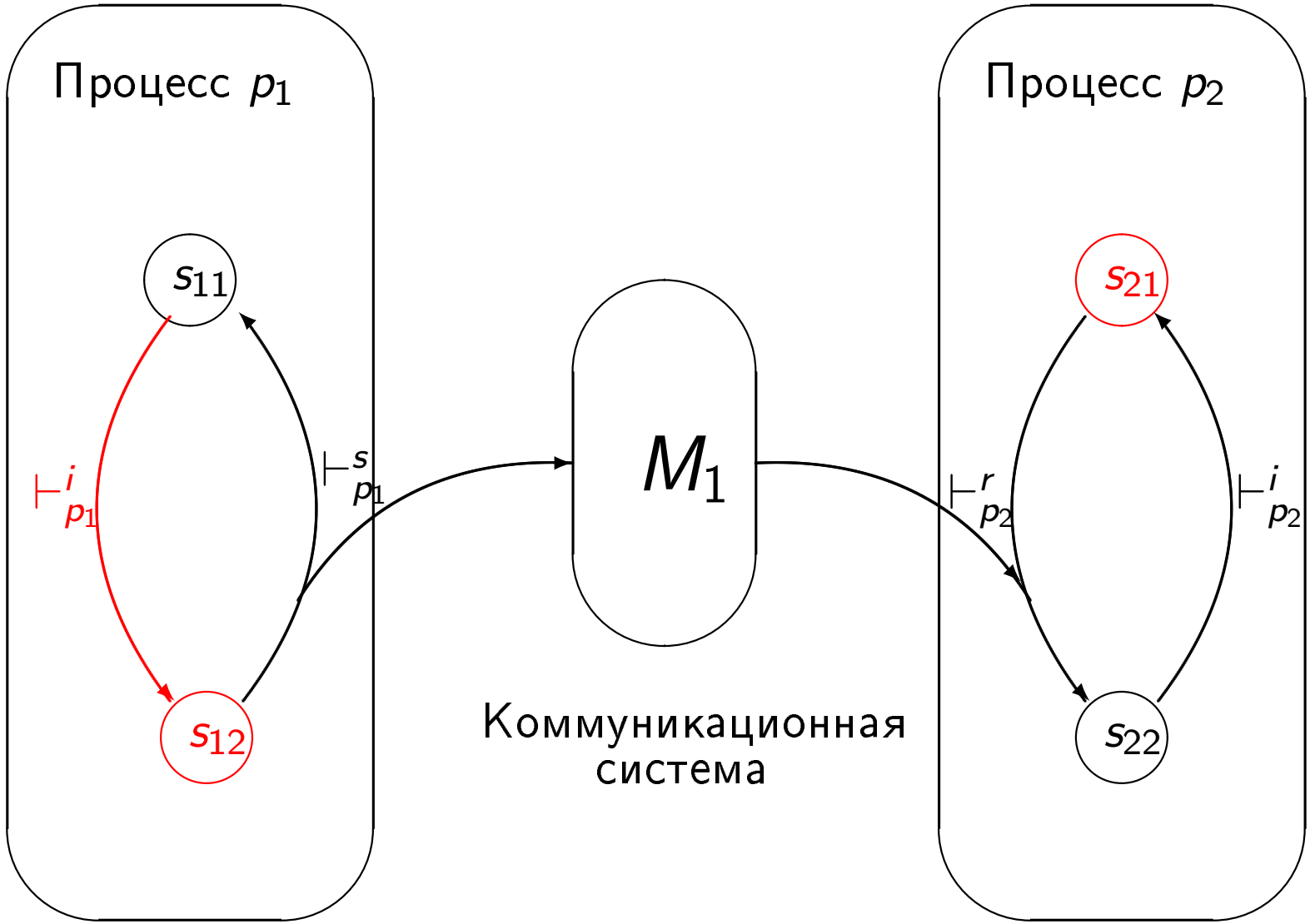
Справедливость.



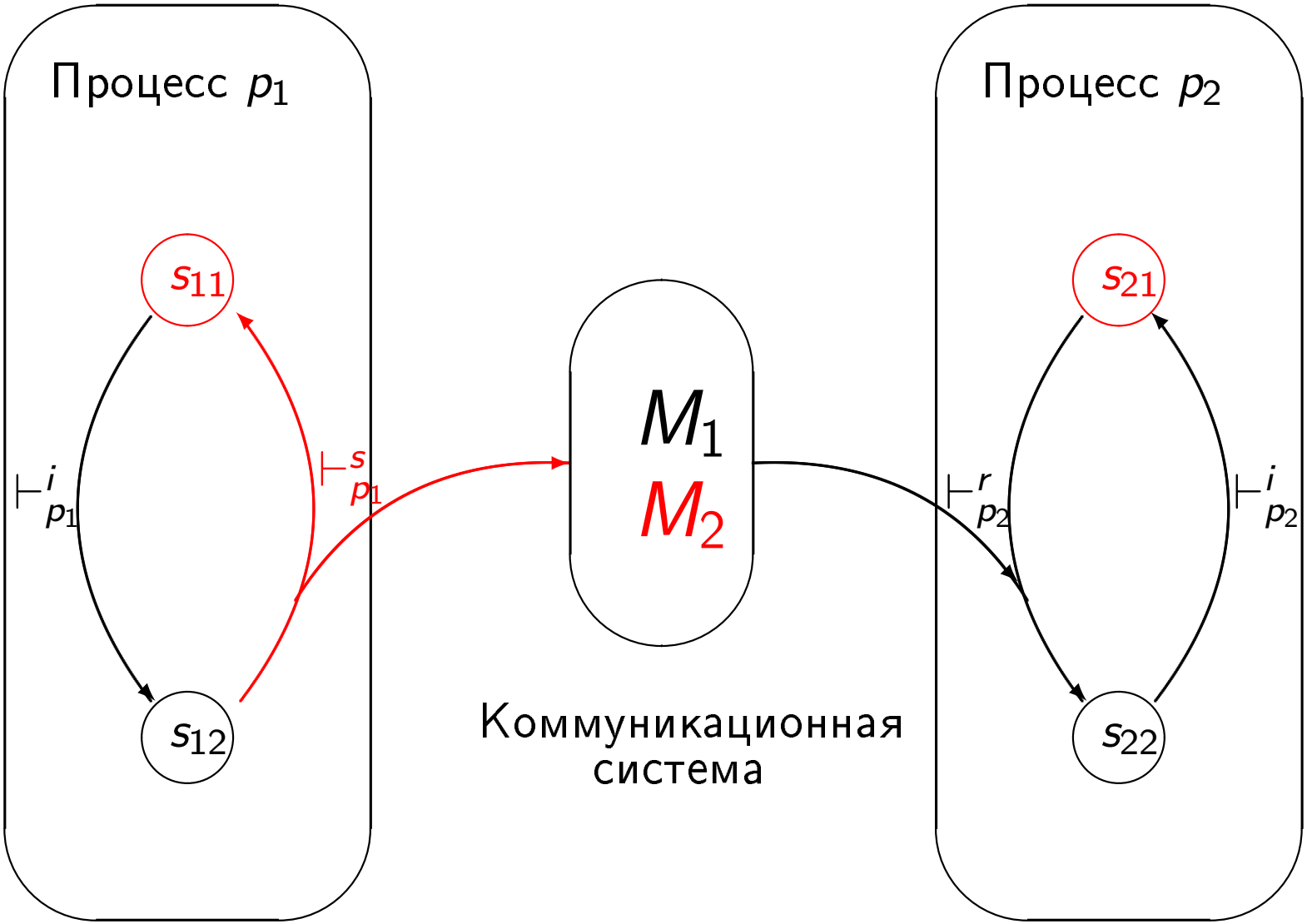
Справедливость.



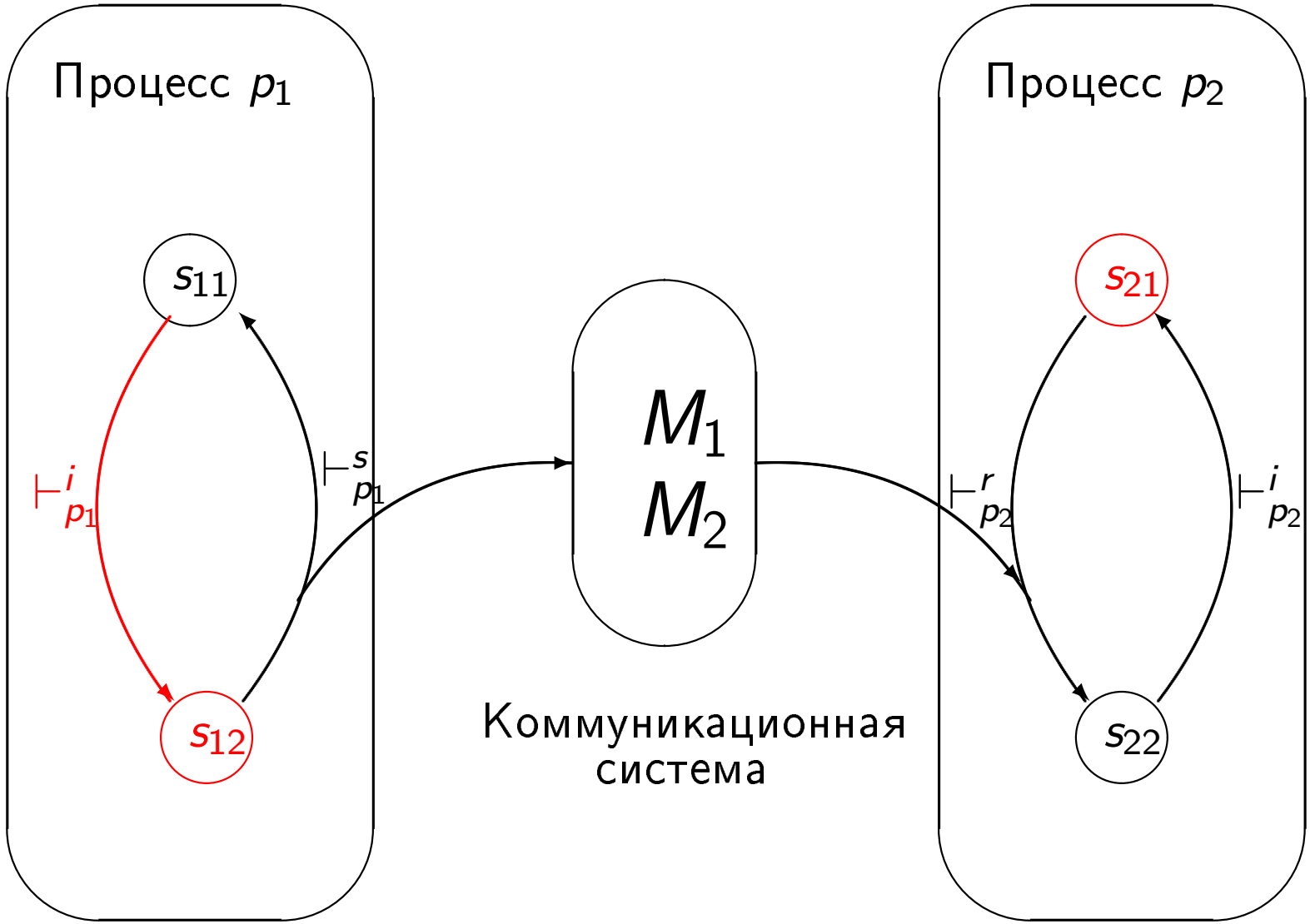
Справедливость.



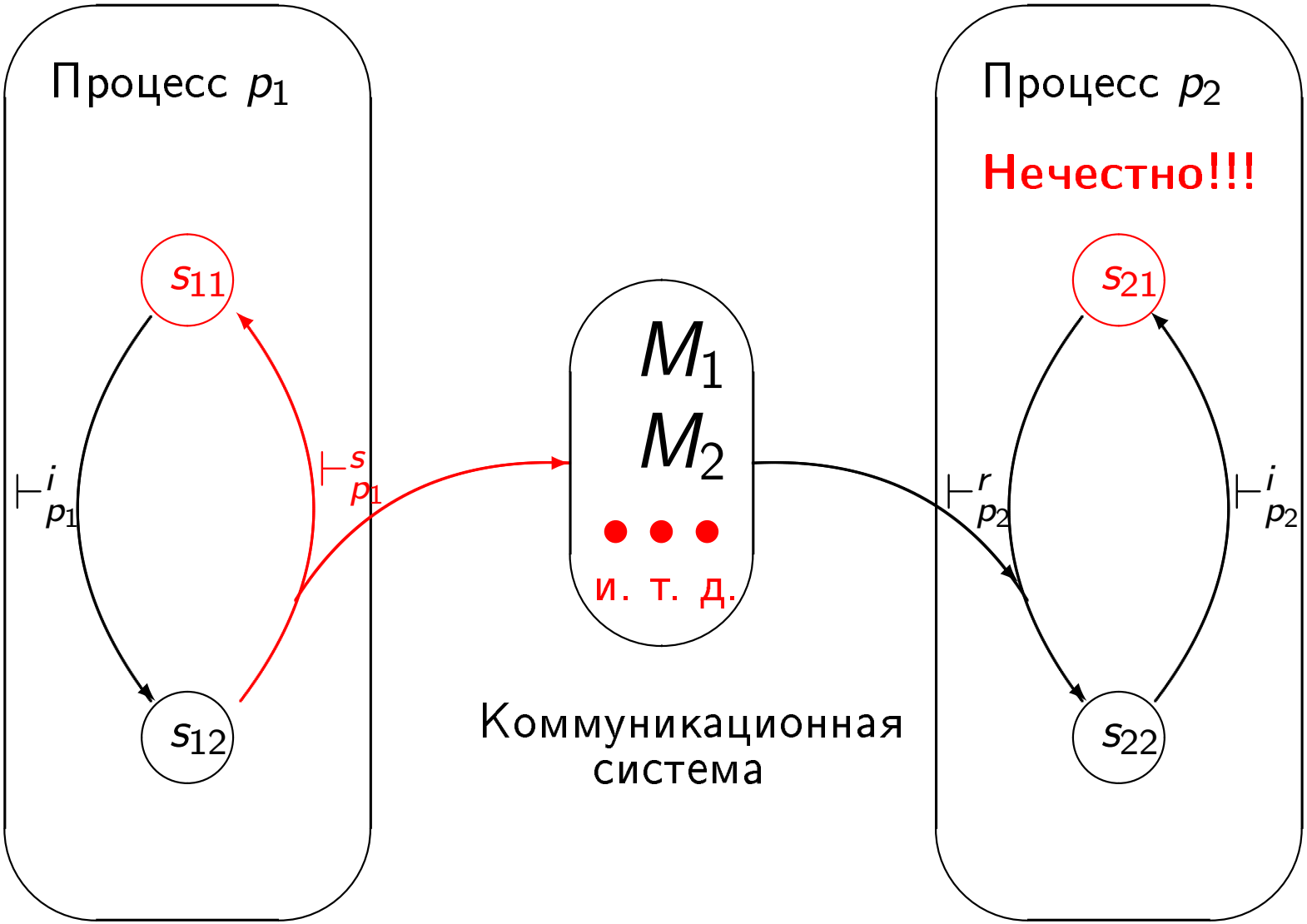
Справедливость.



Справедливость.



Справедливость.



Причинно-следственный порядок событий.

Сопоставим выполнению $E = (\gamma_0, \gamma_1, \ldots)$ связанную с ним последовательность событий $\overline{\mathbf{E}} = (e_0, e_1, \ldots)$, где e_i обозначает событие преобразования конфигурации γ_i в конфигурацию γ_{i+1}. Для визуализации выполнения можно использовать *пространственно–временные диаграммы*.

Причинно-следственный порядок событий.

Сопоставим выполнению $E = (\gamma_0, \gamma_1, \ldots)$ связанную с ним последовательность событий $\overline{\mathbf{E}} = (e_0, e_1, \ldots)$, где e_i обозначает событие преобразования конфигурации γ_i в конфигурацию γ_{i+1}. Для визуализации выполнения можно использовать *пространственно–временные диаграммы*.

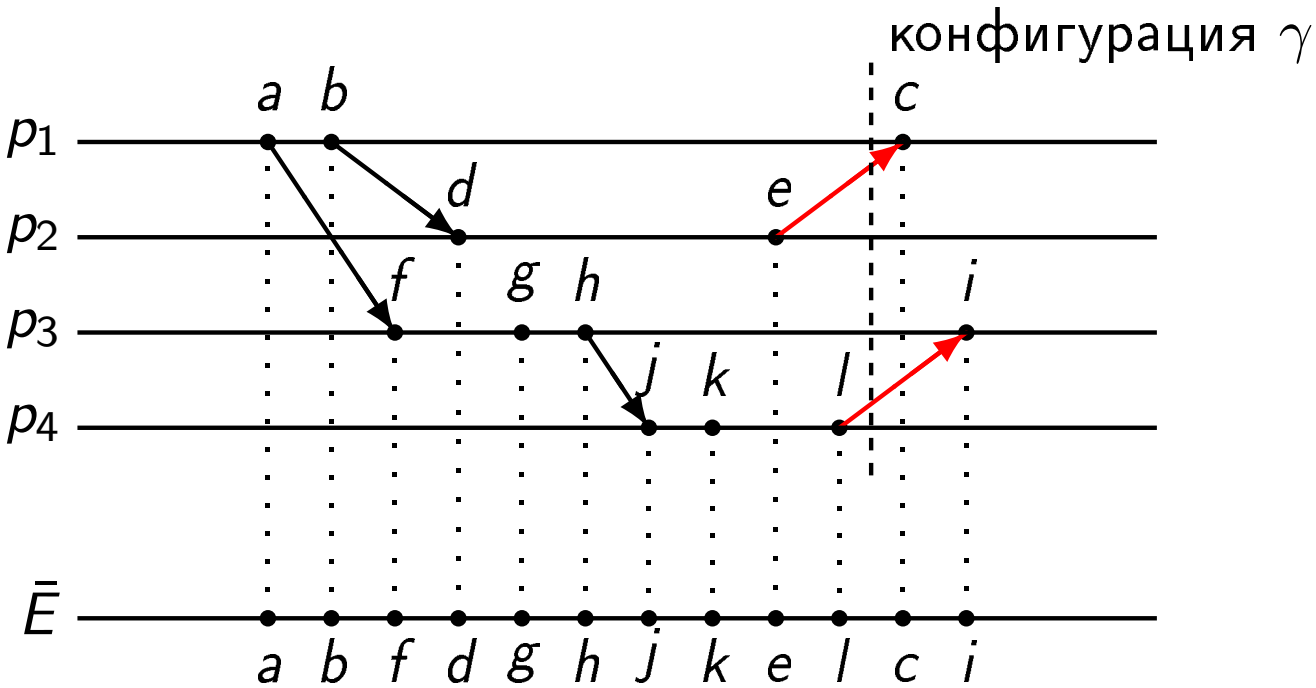


Рис.: Пример пространственно-временной диаграммы.

Причинно-следственный порядок событий.

Теорема 1.

Пусть γ — конфигурация распределенной системы (с асинхронным обменом сообщениями), и пусть e_p и e_q — события, которые происходят в разных процессах p и q, и при этом оба события допустимы в конфигурации γ.

Причинно-следственный порядок событий.

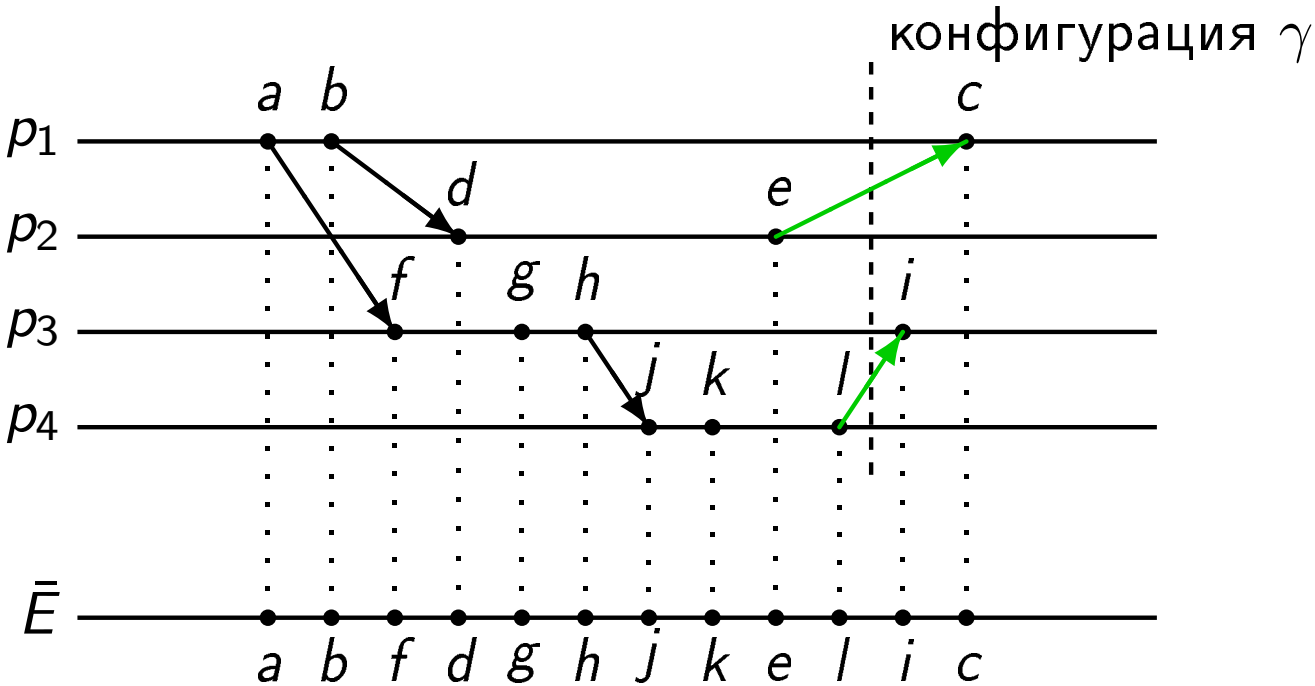
Теорема 1.

Пусть γ — конфигурация распределенной системы (с асинхронным обменом сообщениями), и пусть e_p и e_q — события, которые происходят в разных процессах p и q, и при этом оба события допустимы в конфигурации γ. Тогда событие e_p допустимо в конфигурации $e_q(\gamma)$, а событие e_q — в конфигурации $e_p(\gamma)$, и при этом $e_p(e_q(\gamma)) = e_q(e_p(\gamma))$.

Причинно-следственный порядок событий.

Теорема 1.

Пусть γ — конфигурация распределенной системы (с асинхронным обменом сообщениями), и пусть e_p и e_q — события, которые происходят в разных процессах p и q, и при этом оба события допустимы в конфигурации γ. Тогда событие e_p допустимо в конфигурации $e_q(\gamma)$, а событие e_q — в конфигурации $e_p(\gamma)$, и при этом $e_p(e_q(\gamma)) = e_q(e_p(\gamma))$.



Причинно-следственный порядок событий.

Доказательство.

Причинно-следственный порядок событий.

Доказательство.

Придумайте его сами! Это несложно.

Причинно-следственный порядок событий.

Доказательство.

Придумайте его сами! Это несложно.

Нужно лишь перебрать все пары типов событий, которые могут удовлетворять условиям теоремы, и воспользоваться определением выполнения.

Причинно-следственный порядок событий.

Определение 9.

Пусть задано выполнение E. Отношением **причинно-следственного порядка** $\prec$ называется наименьшее отношение, удовлетворяющее следующим условиям

Причинно-следственный порядок событий.

Определение 9.

Пусть задано выполнение E. Отношением **причинно-следственного порядка** $\prec$ называется наименьшее отношение, удовлетворяющее следующим условиям

1. Если e и f — два разных события одного и того же процесса, и при этом событие e происходит прежде события f, то $e \prec f$.

Причинно-следственный порядок событий.

Определение 9.

Пусть задано выполнение E. Отношением **причинно-следственного порядка** $\prec$ называется наименьшее отношение, удовлетворяющее следующим условиям

1. Если e и f — два разных события одного и того же процесса, и при этом событие e происходит прежде события f, то $e \prec f$.
2. Если s — это событие отправления сообщения, а r — вазаимосвязанное с ним событие приема сообщения, то $s \prec r$.

Причинно-следственный порядок событий.

Определение 9.

Пусть задано выполнение E. Отношением **причинно-следственного порядка** $\prec$ называется наименьшее отношение, удовлетворяющее следующим условиям

1. Если e и f — два разных события одного и того же процесса, и при этом событие e происходит прежде события f, то $e \prec f$.
2. Если s — это событие отправления сообщения, а r — вазаимосвязанное с ним событие приема сообщения, то $s \prec r$.
3. $\prec$ обладает свойством транзитивности.

Причинно-следственный порядок событий.

Определение 9.

Пусть задано выполнение E. Отношением **причинно-следственного порядка** $\prec$ называется наименьшее отношение, удовлетворяющее следующим условиям

1. Если e и f — два разных события одного и того же процесса, и при этом событие e происходит прежде события f, то $e \prec f$.
2. Если s — это событие отправления сообщения, а r — вазаимосвязанное с ним событие приема сообщения, то $s \prec r$.
3. $\prec$ обладает свойством транзитивности.

$a \preceq b$ — отношение **частичного порядка**, которое определятся формулой $(a \prec b \vee a = b)$.

Причинно-следственный порядок событий.

Определение 9.

Пусть задано выполнение E. Отношением **причинно-следственного порядка** $\prec$ называется наименьшее отношение, удовлетворяющее следующим условиям

1. Если e и f — два разных события одного и того же процесса, и при этом событие e происходит прежде события f, то $e \prec f$.
2. Если s — это событие отправления сообщения, а r — вазаимосвязанное с ним событие приема сообщения, то $s \prec r$.
3. $\prec$ обладает свойством транзитивности.

$a \preceq b$ — отношение **частичного порядка**, которое определятся формулой $(a \prec b \vee a = b)$.
События a и b, для которых не выполняются оба отношения $a \preceq b$ и $b \preceq a$, будем называть **параллельными** и обозначать $a \parallel b$.

Причинно-следственный порядок событий.

Задача.

Причинно-следственный порядок событий.

Задача.

1. Докажите, что введенное таким образом отношение $\prec$ является отношением частичного порядка?

Причинно-следственный порядок событий.

Задача.

1. Докажите, что введенное таким образом отношение $\prec$ является отношением частичного порядка?
2. При каких условиях это отношение будет являться отношением полного (линейного) порядка?

Причинно-следственный порядок событий.

Задача.

1. Докажите, что введенное таким образом отношение $\prec$ является отношением частичного порядка?
2. При каких условиях это отношение будет являться отношением полного (линейного) порядка?
3. Будет ли частично упорядоченное множество событий выполнения образовывать решетку?

Причинно-следственный порядок событий.

Рассмотрим некоторое выполнение $E = (\gamma_0, \gamma_1, \gamma_2, \ldots)$ и связанную с ним последовательность событий $\overline{\mathbf{E}} = (e_0, e_1, e_2, \ldots)$.

Причинно-следственный порядок событий.

Рассмотрим некоторое выполнение $E = (\gamma_0, \gamma_1, \gamma_2, \ldots)$ и связанную с ним последовательность событий $\overline{\mathbf{E}} = (e_0, e_1, e_2, \ldots)$.

Предположим, что задана перестановка f элементов последовательности $\overline{\mathbf{E}}$. Это означает, что существует такая перестановка σ на множестве натуральных чисел, что $f_i = e_{\sigma(i)}$.

Причинно-следственный порядок событий.

Рассмотрим некоторое выполнение $E = (\gamma_0, \gamma_1, \gamma_2, \ldots)$ и связанную с ним последовательность событий $\overline{\mathbf{E}} = (e_0, e_1, e_2, \ldots)$.

Предположим, что задана перестановка f элементов последовательности $\overline{\mathbf{E}}$. Это означает, что существует такая перестановка σ на множестве натуральных чисел, что $f_i = e_{\sigma(i)}$.

Перестановка $(f_0, f_1, f_2, \ldots)$ событий выполнения E **сохраняет причинно-следственный порядок**, если $f_i \preceq f_j$ влечет $i \leq j$, т. е. ни одно событие не появляется в этой последовательности прежде того события, от которого оно зависит.

Причинно-следственный порядок событий.

Теорема 2.

Пусть $f = (f_0, f_1, f_2, \ldots)$ — перестановка событий выполнения E, сохраняющая причинно-следственный порядок событий.

Причинно-следственный порядок событий.

Теорема 2.

Пусть $f = (f_0, f_1, f_2, \ldots)$ — перестановка событий выполнения E, сохраняющая причинно-следственный порядок событий. Тогда f определяет единственное выполнение F, начинающееся с той же конфигурации, что и E. При этом в F происходит столько же событий, сколько и в E, и в том случае, если E — конечное выполнение, то последняя конфигурация выполнения F будет точно такой же, как последняя конфигурация выполнения E.

Причинно-следственный порядок событий.

Теорема 2.

Пусть $f = (f_0, f_1, f_2, \ldots)$ — перестановка событий выполнения E, сохраняющая причинно-следственный порядок событий. Тогда f определяет единственное выполнение F, начинающееся с той же конфигурации, что и E. При этом в F происходит столько же событий, сколько и в E, и в том случае, если E — конечное выполнение, то последняя конфигурация выполнения F будет точно такой же, как последняя конфигурация выполнения E.

Доказательство (эскиз).

1. Всякая перестановка (в т. ч. сохраняющая порядок) — композиция транспозиций (перестановок) соседних элементов.

Причинно-следственный порядок событий.

Теорема 2.

Пусть $f = (f_0, f_1, f_2, \ldots)$ — перестановка событий выполнения E, сохраняющая причинно-следственный порядок событий. Тогда f определяет единственное выполнение F, начинающееся с той же конфигурации, что и E. При этом в F происходит столько же событий, сколько и в E, и в том случае, если E — конечное выполнение, то последняя конфигурация выполнения F будет точно такой же, как последняя конфигурация выполнения E.

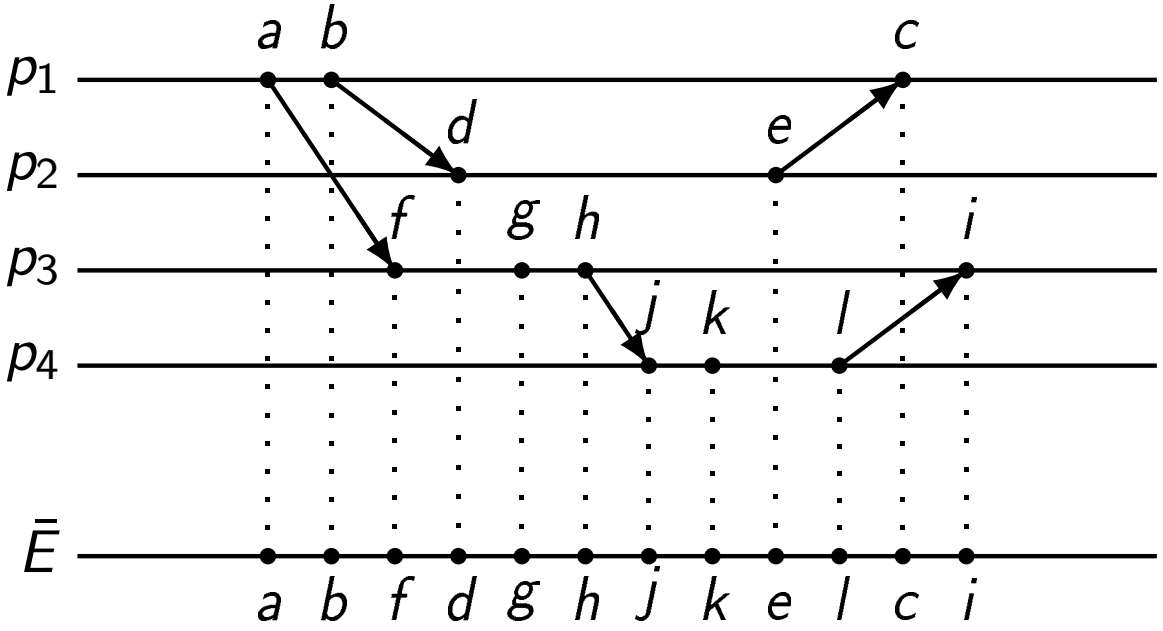
Доказательство (эскиз).

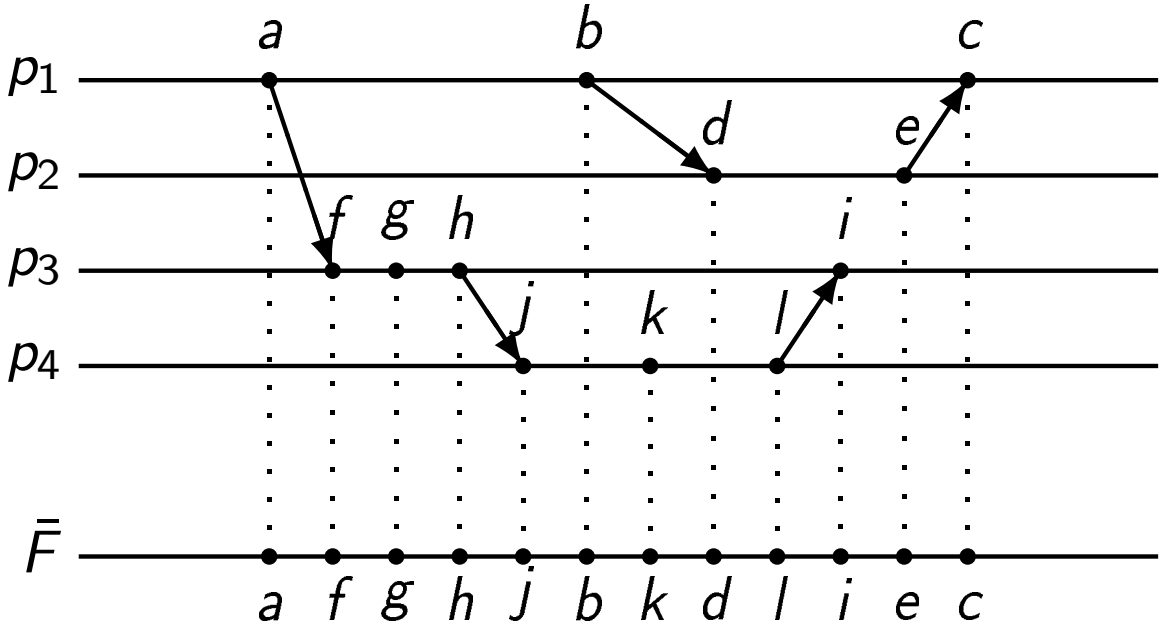
1. Всякая перестановка (в т. ч. сохраняющая порядок) — композиция транспозиций (перестановок) соседних элементов.
2. Для транспозиции соседних событий применяем теорему 1.

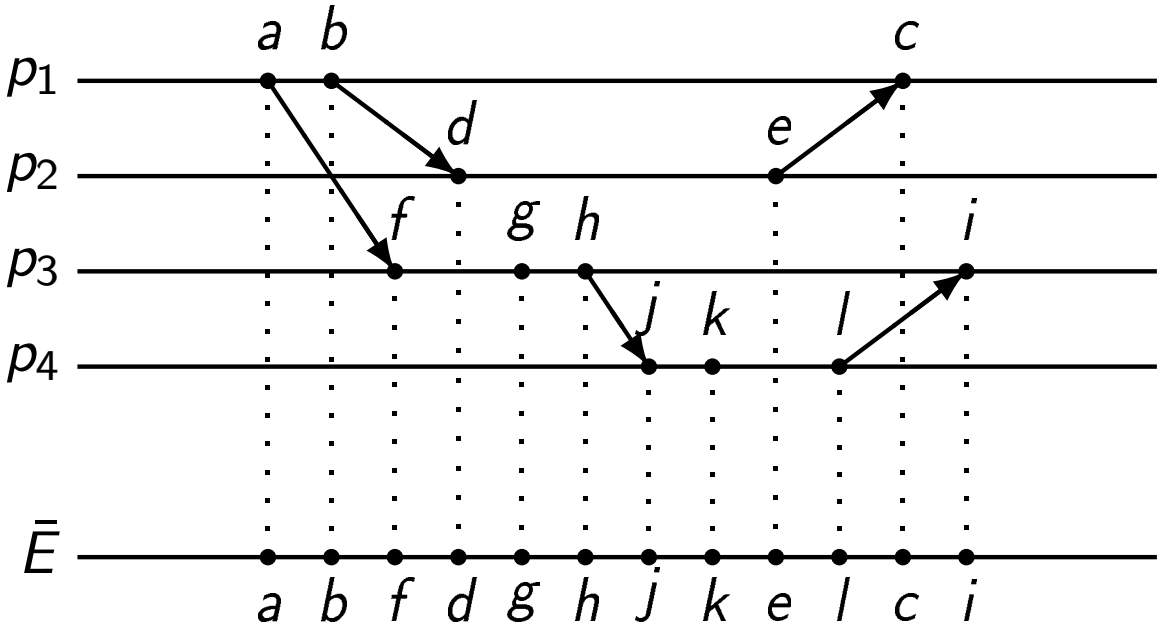
Причинно-следственный порядок событий.

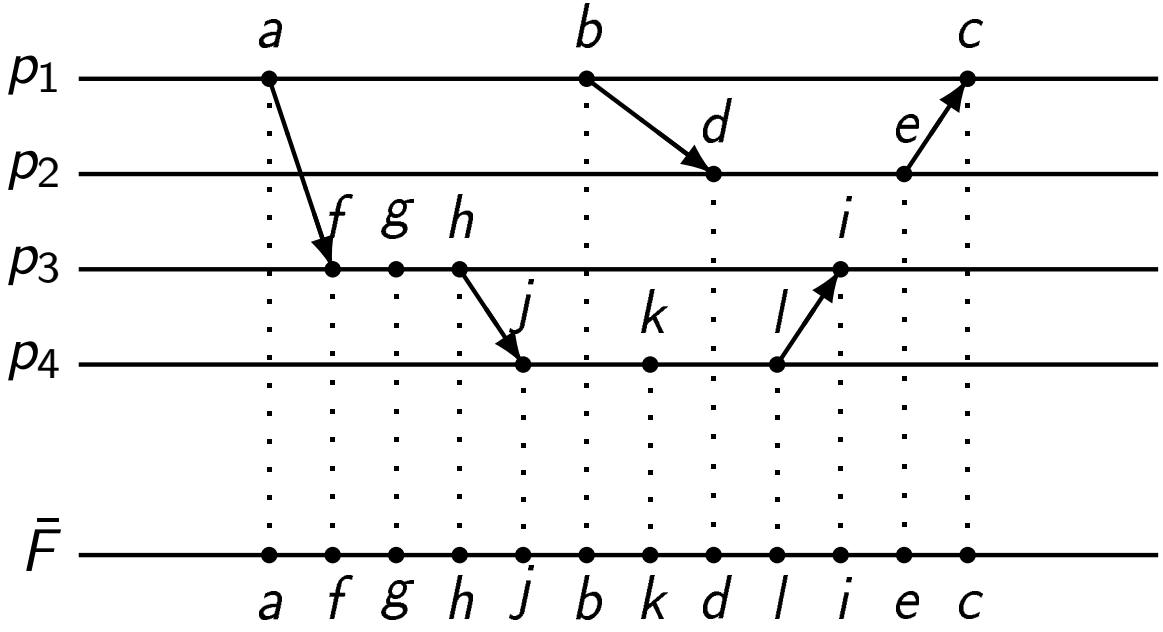
Определение 10.

Два выполнения E и F, имеющие одно и то же множество событий, в котором задан один и тот же причинно-следственный порядок событий, называются **эквивалентными** ($E \sim F$).


a
b
c
p_1
d
e
p_2
f
g
h
i
p_3
j
k
l
p_4
$\bar{E}$
a b f d g h j k e l c i


a
b
c
p_1
d
e
p_2
f
g
h
i
p_3
j
k
l
p_4
$\bar{F}$
a f g h j b k d l i e c


a b c
p1
d
e
p2
f g h i
p3
j k l
p4
Ē
a b f d g h j k e l c i


a b c
p1
d
e
p2
f g h i
p3
j k l
p4
F̄
a f g h j b k d l i e c

Причинно-следственный порядок событий.

Определение 11.

Вычислением распределенного алгоритма называется класс эквивалентности выполнений алгоритма по отношению эквивалентности $\sim$.

Причинно-следственный порядок событий.

Определение 11.

Вычислением распределенного алгоритма называется класс эквивалентности выполнений алгоритма по отношению эквивалентности $\sim$.

Нет смысла говорить о конфигурациях вычисления, потому что различные выполнения одного и того же вычисления могут иметь разные конфигурации. Но при этом имеет смысл говорить о совокупности событий вычисления, так как во всех выполнениях одного и того вычисления происходят одни и те же события. Причинно-следственный порядок на множестве событий также однозначно определяется вычислением.

Причинно-следственный порядок событий.

Задача.

Причинно-следственный порядок событий.

Задача.

1. Будет ли для синхронных распределенных систем справедлив аналог Теоремы 1?

Причинно-следственный порядок событий.

Задача.

1. Будет ли для синхронных распределенных систем справедлив аналог Теоремы 1?
2. Можно ли для синхронных распределенных систем определить отношение причинно-следственной зависимости, аналогичное отношению $\prec$?

Причинно-следственный порядок событий.

Задача.

1. Будет ли для синхронных распределенных систем справедлив аналог Теоремы 1?
2. Можно ли для синхронных распределенных систем определить отношение причинно-следственной зависимости, аналогичное отношению $\prec$?
3. Будет ли для синхронных распределенных систем справедлив аналог Теоремы 2?

Логические часы.

Мы можем снабдить распределенные системы часами, при помощи которых можно «отсчитывать» причинно-следственную зависимость событий, подобно тому, как физические часы отсчитывают реальное время.

Логические часы.

Мы можем снабдить распределенные системы часами, при помощи которых можно «отсчитывать» причинно-следственную зависимость событий, подобно тому, как физические часы отсчитывают реальное время.

Определение 12.

Часами назовем всякую функцию Θ, отображающую множество событий в некоторое упорядоченное множество так, что при этом выполняется соотношение

$$a \prec b \implies \Theta(a) < \Theta(b).$$

Примеры часов.

Глобальные часы.

Для выполнения E, представленного последовательностью событий $(e_0, e_1, e_2, \ldots)$, положим $\Theta_g(e_i) = i$. Таким образом, каждое событие помечается тем порядковым номером, который оно имеет в этой последовательности.

Примеры часов.

Глобальные часы.

Для выполнения E, представленного последовательностью событий $(e_0, e_1, e_2, \ldots)$, положим $\Theta_g(e_i) = i$. Таким образом, каждое событие помечается тем порядковым номером, который оно имеет в этой последовательности.

Такой функцией может может воспользоваться сторонний наблюдатель системы, который способен обозревать последовательность происходящих событий. Однако эту последовательность нельзя наблюдать, пребывая *внутри* системы; иначе говоря, Θ_g определена на множестве выполнений, а не вычислений. Распределенный алгоритм не способен вычислять функцию Θ_g. Почему?

Примеры часов.

Задача.

Докажите, что существуют такие распределенные системы, которые не способны вычислять функцию глобальных часов Θ_g.

Примеры часов.

Задача.

Докажите, что существуют такие распределенные системы, которые не способны вычислять функцию глобальных часов Θ_g.

Намек:

Покажите, что возможны такие вычисления, для которых в разных выполнениях функция Θ_g будет иметь разные значения для событий каждого процесса.

Попробуйте показать, что это противоречит предположению о вычислимости функции Θ_g распределенным алгоритмом.

Примеры часов.

Часы реального времени.

Можно *расширить* нашу модель, снабдив каждый процесс встроенным часовым механизмом. Тогда для каждого события можно зафиксировать время его осуществления. Вычисляемые таким образом значения удовлетворяют определению часов.

Примеры часов.

Часы реального времени.

Можно *расширить* нашу модель, снабдив каждый процесс встроенным часовым механизмом. Тогда для каждого события можно зафиксировать время его осуществления. Вычисляемые таким образом значения удовлетворяют определению часов. Распределенные системы с часами реального времени не подпадают под определение 6, потому что физические свойства часов позволяют синхронизировать изменения состояний в разных процессах. Течение времени происходит во всех процессах, и это приводит к тому, что возникают переходы, которые изменяют состояния (а именно, показания часов) во всех процессах системы. Оказывается, что эти «глобальные переходы» решительным образом изменяют всю модель: если предполагать наличие часов реального времени, то теорема 1 перестает быть справедливой.

Примеры часов.

Логические часы Лэмпорта.

Часовая функция Лэмпорта приписывает событию a число $\Theta_L(a) = k$, равное длине самой протяженной последовательности событий $(e_1, \ldots, e_k)$, которая удовлетворяет условию

$$e_1 \prec e_2 \prec \ldots \prec e_k = a.$$

Примеры часов.

Логические часы Лэмпорта.

Часовая функция Лэмпорта приписывает событию a число $\Theta_L(a) = k$, равное длине самой протяженной последовательности событий $(e_1, \ldots, e_k)$, которая удовлетворяет условию

$$e_1 \prec e_2 \prec \ldots \prec e_k = a.$$

Задача

Покажите, что введенная таким образом функция Θ_L действительно является часами.

Примеры часов.

Логические часы Лампорта.

Значение Θ_L для каждого события может быть вычислено распределенным алгоритмом по следующим правилам.

Примеры часов.

Логические часы Лампорта.

Значение Θ_L для каждого события может быть вычислено распределенным алгоритмом по следующим правилам.

1. Значение $\Theta_L(a)$ полагается равным 0, если a — первое событие в процессе.

Примеры часов.

Логические часы Лампорта.

Значение Θ_L для каждого события может быть вычислено распределенным алгоритмом по следующим правилам.

1. Значение $\Theta_L(a)$ полагается равным 0, если a — первое событие в процессе.
2. Если a — это внутреннее событие процесса или событие отправления сообщения, и a' — это предшествующее событие в том же самом процессе, то $\Theta_L(a) = \Theta_L(a') + 1$.

Примеры часов.

Логические часы Лампорта.

Значение Θ_L для каждого события может быть вычислено распределенным алгоритмом по следующим правилам.

1. Значение $\Theta_L(a)$ полагается равным 0, если a — первое событие в процессе.
2. Если a — это внутреннее событие процесса или событие отправления сообщения, и a' — это предшествующее событие в том же самом процессе, то $\Theta_L(a) = \Theta_L(a') + 1$.
3. Если a — событие приема сообщения, a' — предшествующее событие в том же самом процессе, и b — событие отправления сообщения, соответствующее a, то $\Theta_L(a) = \max(\Theta_L(a'), \Theta_L(b)) + 1$.

Примеры часов.

---

var θ_p : **integer** **init** 0 ;

(* Внутреннее событие *)
$\theta_p := \theta_p + 1$;
Изменение состояния

(* Событие отправления сообщения *)
$\theta_p := \theta_p + 1$;
send (**messg**, θ_p) ; Изменение состояния

(* Событие приема сообщения *)
receive (**messg**, θ) ; $\theta_p := \max(\theta_p, \theta) + 1$;
Изменение состояния

АЛГОРИТМ: Логические часы Лэмпорта.

Примеры часов.

Задача.

Можно ли построить такую функцию часов Θ, которая

Примеры часов.

Задача.

Можно ли построить такую функцию часов Θ, которая

1. могла быть вычислена распределенным алгоритмом;

Примеры часов.

Задача.

Можно ли построить такую функцию часов Θ, которая

1. могла быть вычислена распределенным алгоритмом;
2. для любого вычисления и для любых двух событий a и b в этом вычислении обладала свойством

$$a \prec b \iff \Theta(a) < \Theta(b).$$

Примеры часов.

Задача.

Можно ли построить такую функцию часов Θ, которая

1. могла быть вычислена распределенным алгоритмом;
2. для любого вычисления и для любых двух событий a и b в этом вычислении обладала свойством

$$a \prec b \iff \Theta(a) < \Theta(b).$$

Намек:

А может быть значениями этой функции следует сделать векторы (наборы чисел)?

Классификация распределенных систем.

Распределенные системы классифицируются по нескольким различным параметрам, наиболее важными из которых являются:

Классификация распределенных систем.

Распределенные системы классифицируются по нескольким различным параметрам, наиболее важными из которых являются:

- тип коммуникации между отдельными процессами (синхронная, асинхронная, смешанная);

Классификация распределенных систем.

Распределенные системы классифицируются по нескольким различным параметрам, наиболее важными из которых являются:

- тип коммуникации между отдельными процессами (синхронная, асинхронная, смешанная);
- топология коммуникационной подсистемы;

Классификация распределенных систем.

Распределенные системы классифицируются по нескольким различным параметрам, наиболее важными из которых являются:

- тип коммуникации между отдельными процессами (синхронная, асинхронная, смешанная);
- топология коммуникационной подсистемы;
- свойства каналов связи;

Классификация распределенных систем.

Распределенные системы классифицируются по нескольким различным параметрам, наиболее важными из которых являются:

- тип коммуникации между отдельными процессами (синхронная, асинхронная, смешанная);
- топология коммуникационной подсистемы;
- свойства каналов связи;
- осведомленность процессов (начальные знания о всей распределенной системе, которыми обладают процессы).

Синхронные и асинхронные системы.

1. Синхронные и асинхронные системы способны взаимно моделировать друг друга. Как?

Синхронные и асинхронные системы.

1. Синхронные и асинхронные системы способны взаимно моделировать друг друга.
2. Асинхронные системы — менее «безопасные», но зато более «живучие», чем синхронные. Почему?

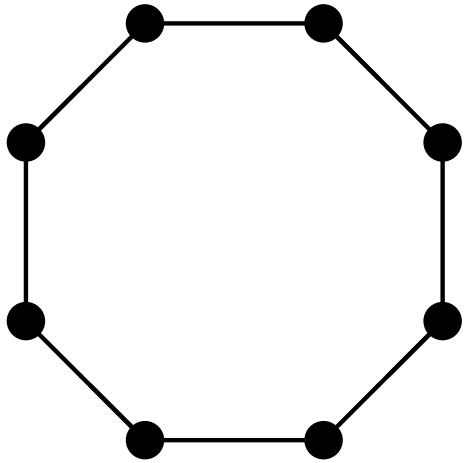
Синхронные и асинхронные системы.

1. Синхронные и асинхронные системы способны взаимно моделировать друг друга.
2. Асинхронные системы — менее «безопасные», но зато более «живучие», чем синхронные.
3. Асинхронные системы более эффективны. Почему?

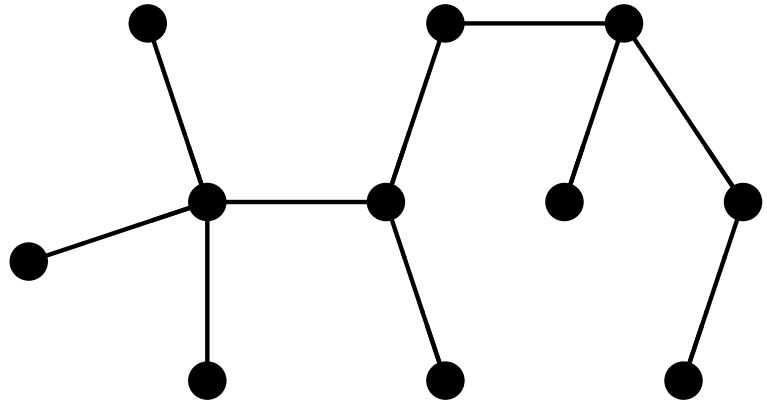
Синхронные и асинхронные системы.

1. Синхронные и асинхронные системы способны взаимно моделировать друг друга.
2. Асинхронные системы — менее «безопасные», но зато более «живучие», чем синхронные.
3. Асинхронные системы более эффективны.
4. Синхронные системы более устойчивы к неисправностям. Почему?

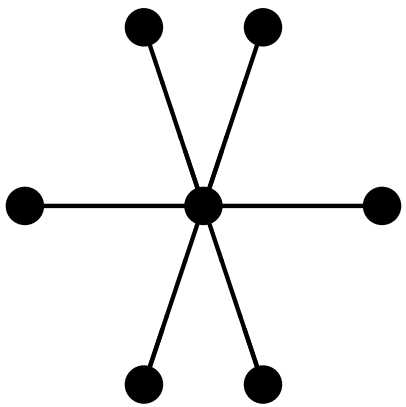
Топология коммуникационной сети.



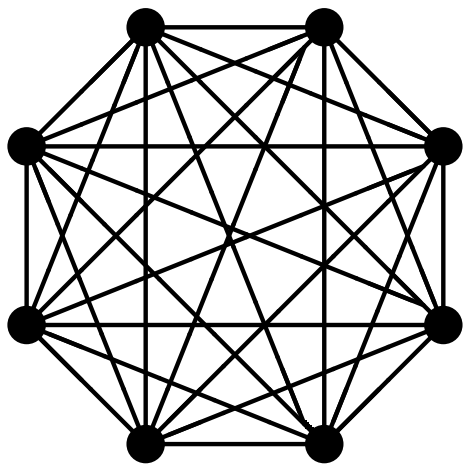
Кольцо



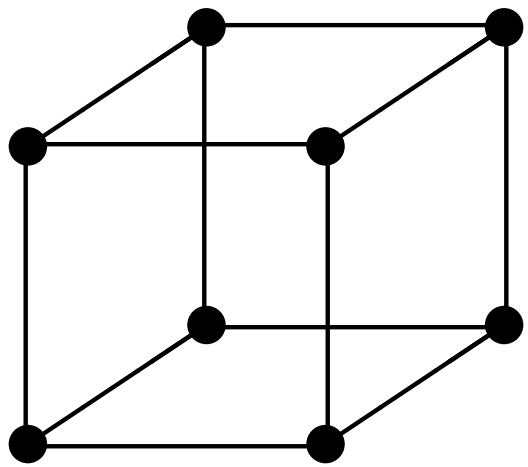
Дерево



Звезда



Клика



Гиперкуб

Свойства каналов связи.

1. **Надежность.** Канал считается **надежным**, если каждое сообщение отправленное по этому каналу непременно доставляется адресату и при этом в единственном экземпляре. Канал, не являющийся надежным, может быть подвержен **коммуникационным неисправностям** нескольких типов: потеря, искажение, дублирование, спонтанное порождение.

Свойства каналов связи.

1. **Надежность.** Канал считается **надежным** , если каждое сообщение отправленное по этому каналу непременно доставляется адресату и при этом в единственном экземпляре. Канал, не являющийся надежным, может быть подвержен **коммуникационным неисправностям** нескольких типов: потеря, искажение, дублирование, спонтанное порождение.
2. **Свойство очередности.** Канал, сохраняющий порядок передаваемых по нему сообщений, называется **очередью** .

Свойства каналов связи.

1. **Надежность.** Канал считается **надежным**, если каждое сообщение отправленное по этому каналу непременно доставляется адресату и при этом в единственном экземпляре. Канал, не являющийся надежным, может быть подвержен **коммуникационным неисправностям** нескольких типов: потеря, искажение, дублирование, спонтанное порождение.
2. **Свойство очередности.** Канал, сохраняющий порядок передаваемых по нему сообщений, называется **очередью** .
3. **Пропускная способность каналов** — количество сообщений, которые могут одновременно пребывать в канале на этапе пересылки. Канал считается **переполненным** в тех конфигурациях, в которых число содержащихся в нем сообщений в точности равно его пропускной способности. Событие отправления допустимо только в тех случаях, когда канал не является переполненным.

Осведомленность процессов.

1. **Топологическая информация.** Информация о топологии включает сведения о числе процессов, о диаметре графа сети, о топологии сети.

Осведомленность процессов.

1. **Топологическая информация.** Информация о топологии включает сведения о числе процессов, о диаметре графа сети, о топологии сети.
2. **Отличительные признаки процессов.** Иногда предполагается, что в процессах имеется переменная, начальным значением которой является имя процесса (разные процессы имеют разные имена).

Осведомленность процессов.

1. **Топологическая информация.** Информация о топологии включает сведения о числе процессов, о диаметре графа сети, о топологии сети.
2. **Отличительные признаки процессов.** Иногда предполагается, что в процессах имеется переменная, начальным значением которой является имя процесса (разные процессы имеют разные имена).
3. **Отличительные признаки соседей.** Если процессы различаются по их уникальным именам, то можно предполагать, что каждому процессу заранее известны имена его соседей. Более сильное допущение предполагает, что каждому процессу известны имена всех процессов системы. При более слабом допущении предполагается, что процессу известно о существовании соседей, но их имена ему неизвестны.

Сложность распределенных алгоритмов.

1. **Сложность по числу обменов сообщениями.** Это общее число сообщений, которые были отправлены по ходу выполнения алгоритма.

Сложность распределенных алгоритмов.

1. **Сложность по числу обменов сообщениями.** Это общее число сообщений, которые были отправлены по ходу выполнения алгоритма.
2. **Битовая сложность.** Это суммарное количество битов, содержащихся в сообщениях.

Сложность распределенных алгоритмов.

1. **Сложность по числу обменов сообщениями.** Это общее число сообщений, которые были отправлены по ходу выполнения алгоритма.
2. **Битовая сложность.** Это суммарное количество битов, содержащихся в сообщениях.
3. **Сложность по времени.** Хронометрирование событий вычисления проводится согласно следующим допущениям.

   - Обработка событий не занимает ни одной единицы времени.
   - Для передачи сообщения требуется не более одной единицы времени, т.е. протяженность отрезка времени между событием отправления и событием приема сообщения не превосходит одной единицы.

   Сложностью алгоритма по времени будет считаться время, которое требуется для проведения вычисления в рамках указанных выше допущений.

КОНЕЦ ЛЕКЦИИ 2.