Распределенные системы и технологии.

Построение моментальных снимков распределенных систем

Дмитрий Юрьевич Чалый
декан факультета ИВТ,
зав. кафедрой информационных и сетевых технологий

22 мая 2016 г.

Зачем нужно сохранять состояние?

Распределенное приложение состоит из многих взаимодействующих процессов.

- Checkpointing. Восстановление после сбоев;
- сборка мусора;
- определение тупиковых состояний;
- корректная остановка вычислений.

Глобальное состояние

Глобальное состояние распределенной системы=

Множество индивидуальных состояний процессов

+

Множество состояний каналов связи

Наивное решение

- Синхронизировать часы всех процессов распределенной системы;
- дать команду процессам сохранить свое состояние в момент времени t;

Проблемы:

- синхронизация часов — трудная задача;
- не происходит сохранения состояния каналов.

Изменение состояния системы

Какой-то процесс:

- получает сообщение;
- отправляет сообщение;
- выполняет действие.

Изменения состояния системы находятся в причинно-следственной связи.

Модель системы

- В системе выполняются процессы $p_1, \ldots, p_n$;
- между любыми двумя процессами существуют два однонаправленных канала связи:

$$p_i \to p_j \text{ и } p_j \to p_i;$$

- каналы связи работают по принципу FIFO;
- в системе нет сбоев;
- все сообщения приходят неповрежденными и не дублируются в каналах.

Дополнительные требования

- Создание образа системы не должно прерывать ее нормальную работу;
- каждый процесс p_i может сохранить и восстановить свое состояние;
- глобальное состояние формируется распределенным образом;
- любой процесс может инициировать создание образа.

Алгоритм Ченди-Лэмпорта

Действия процесса-инициатора p_i:

1. Инициатор p_i сохраняет свое состояние;
2. p_i рассылает специальные сообщения-маркеры другим процессам;
3. p_i начинает запись всех входящих сообщений по всем входящим каналам C_{ji};

Процесс p_j при получении сообщения-маркера от процесса p_k из канала C_{kj}:

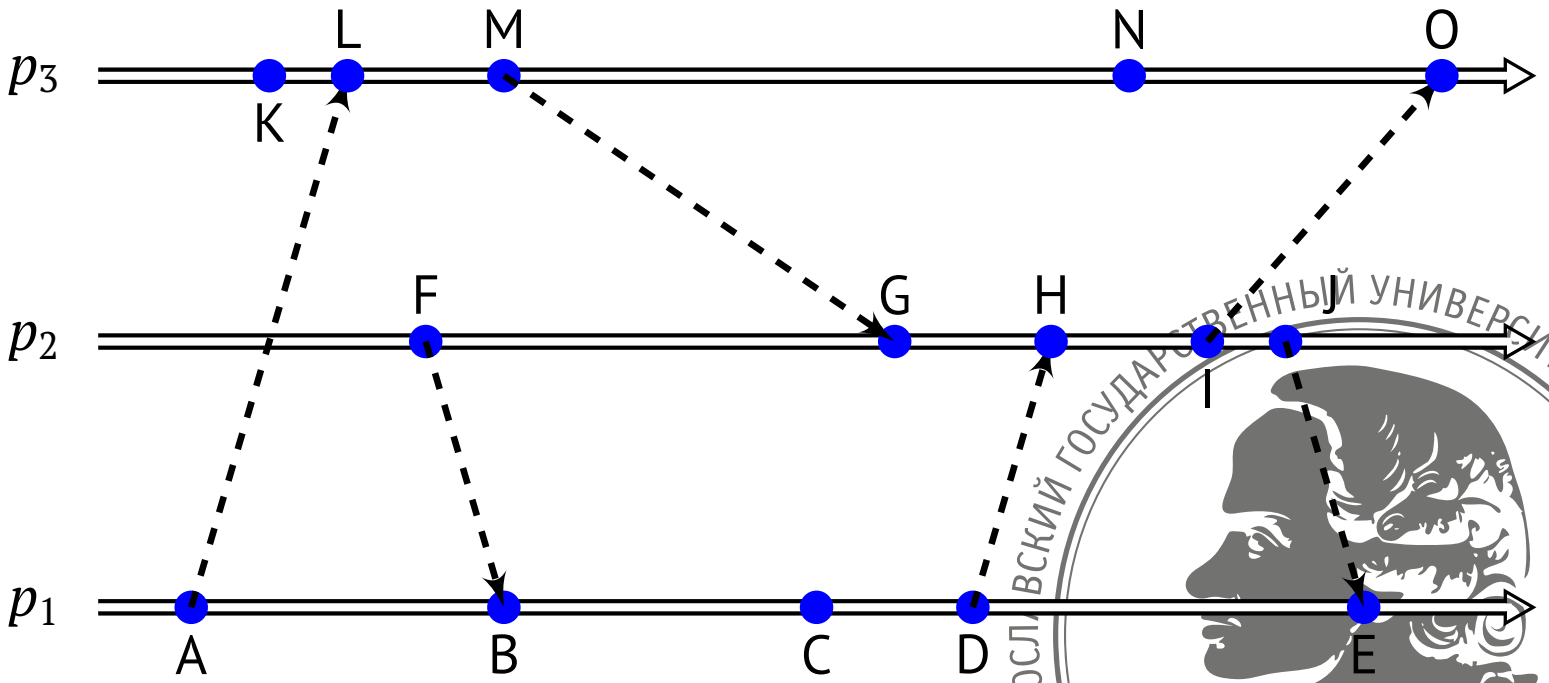
1. если это первый маркер, то p_j:
   - сохраняет свое состояние;
   - отмечает канал C_{kj} как пустой;
   - рассылает сообщения-маркеры по всем исходящим каналам C_{jm};
   - начинает запись всех входящих сообщений по входящим каналам C_{mj}.
2. если это второй маркер, то записывает все сообщения, полученные по каналу C_{kj}, и помечает его как пустой.

Завершение работы алгоритма

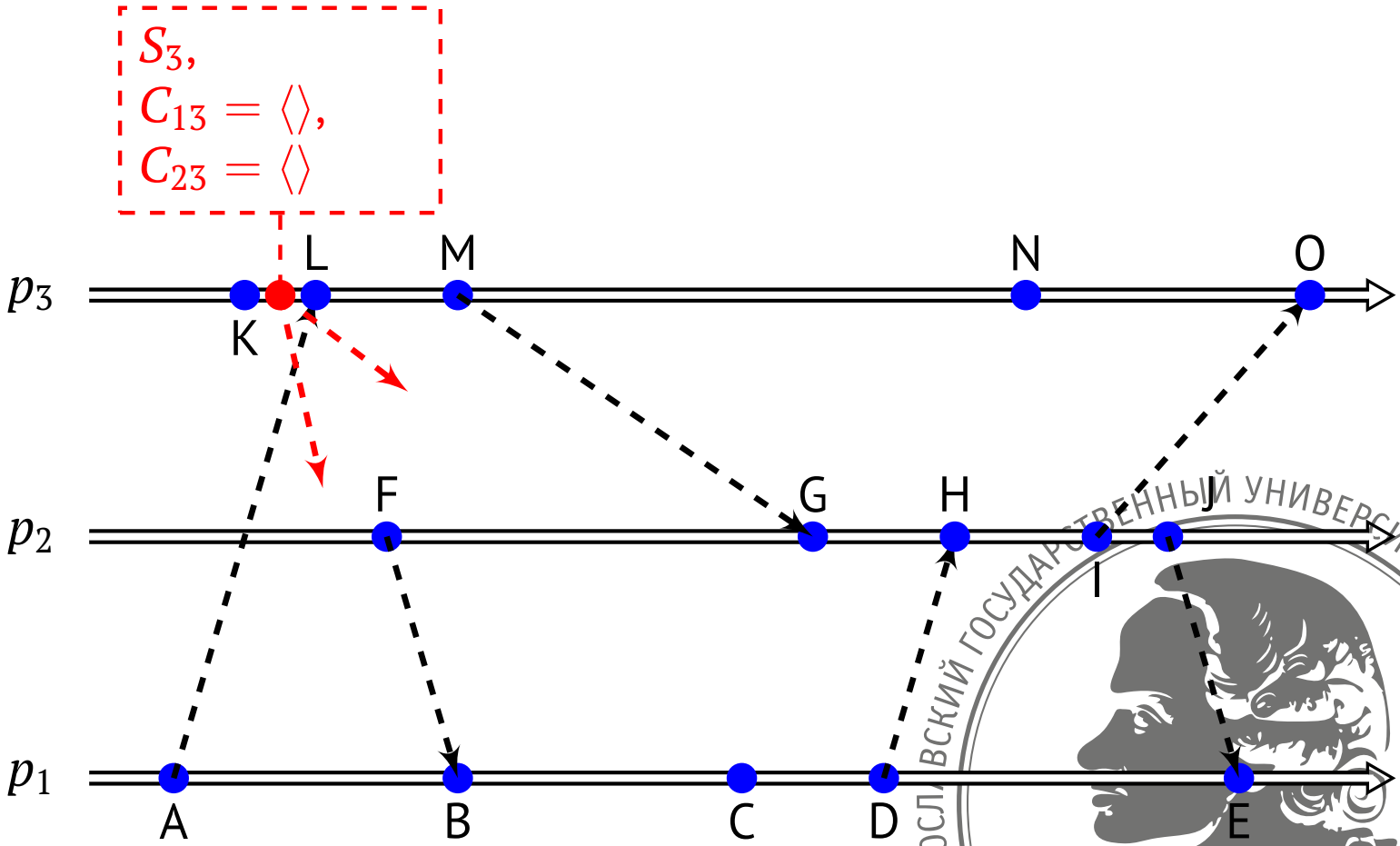
Алгоритм завершает свою работу когда:

- все процессы получили сообщение-маркер и записали свое состояние;
- все процессы получили маркер со всех входящих каналов и записали все входящие сообщения;
- далее централизованно можно собрать эти локальные части и собрать общий образ.

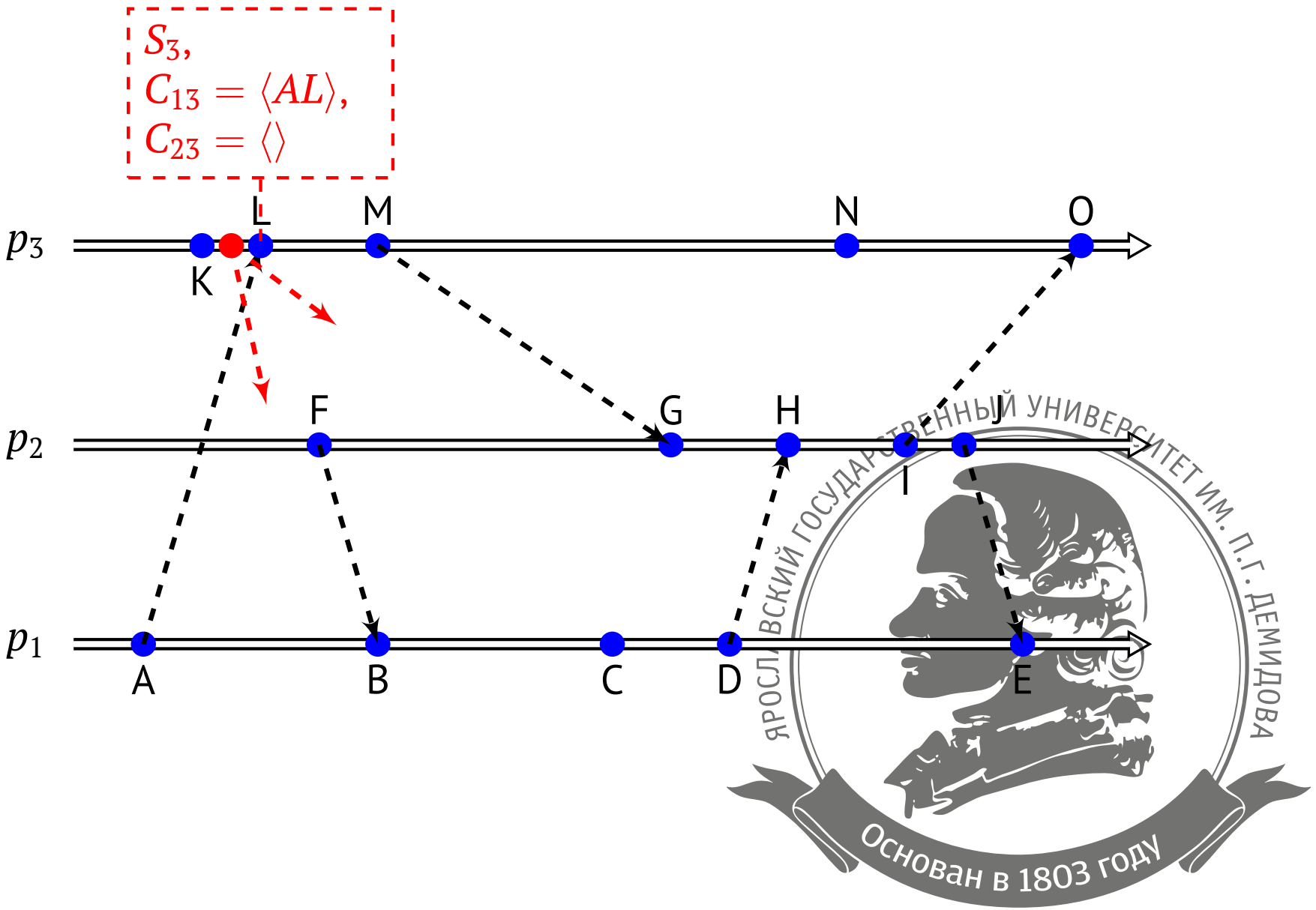
Пример



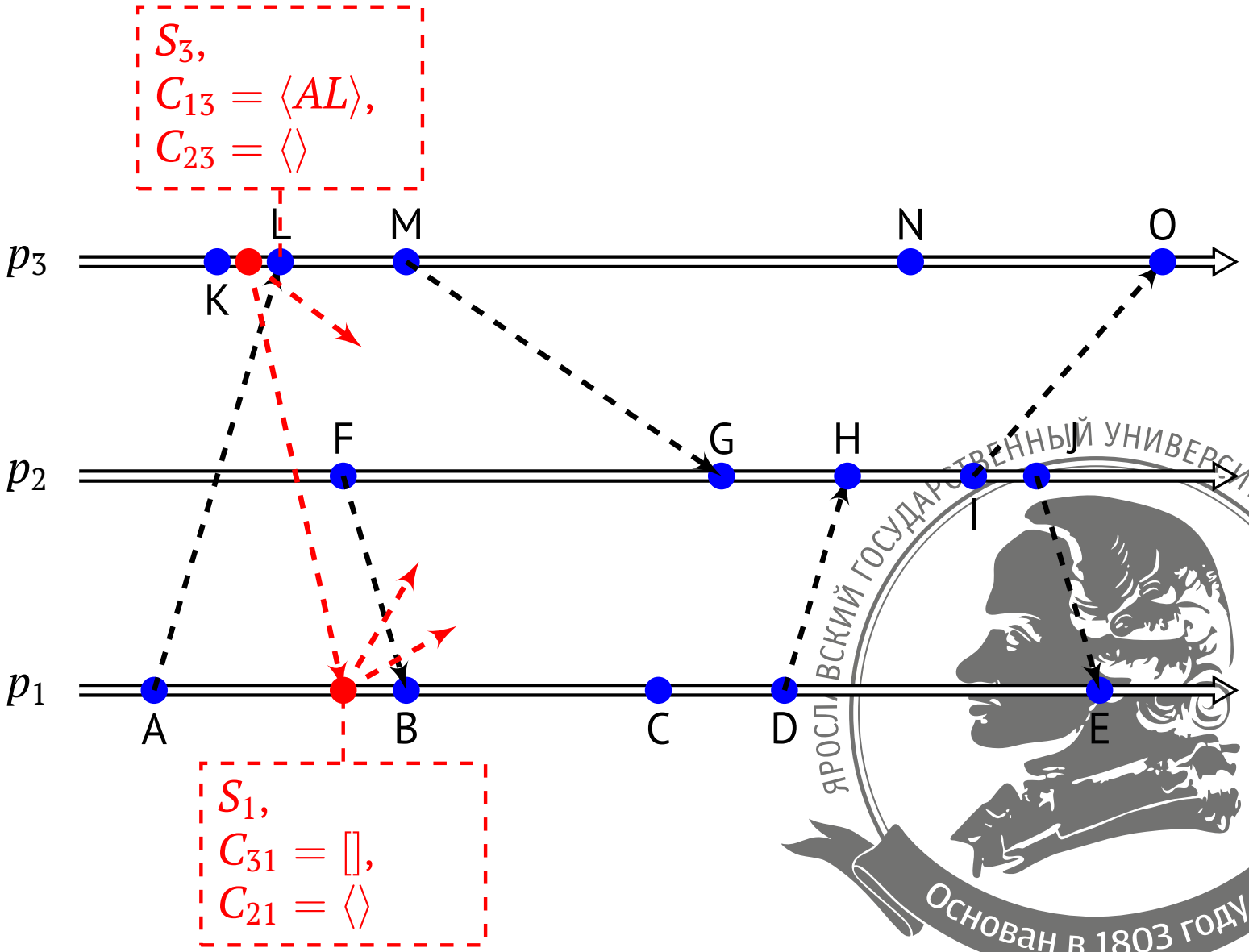
Пример


S_3,
$C_{13} = \langle\rangle$,
$C_{23} = \langle\rangle$
p_3
p_2
p_1
K
L
M
N
O
F
G
H
I
J
A
B
C
D
E

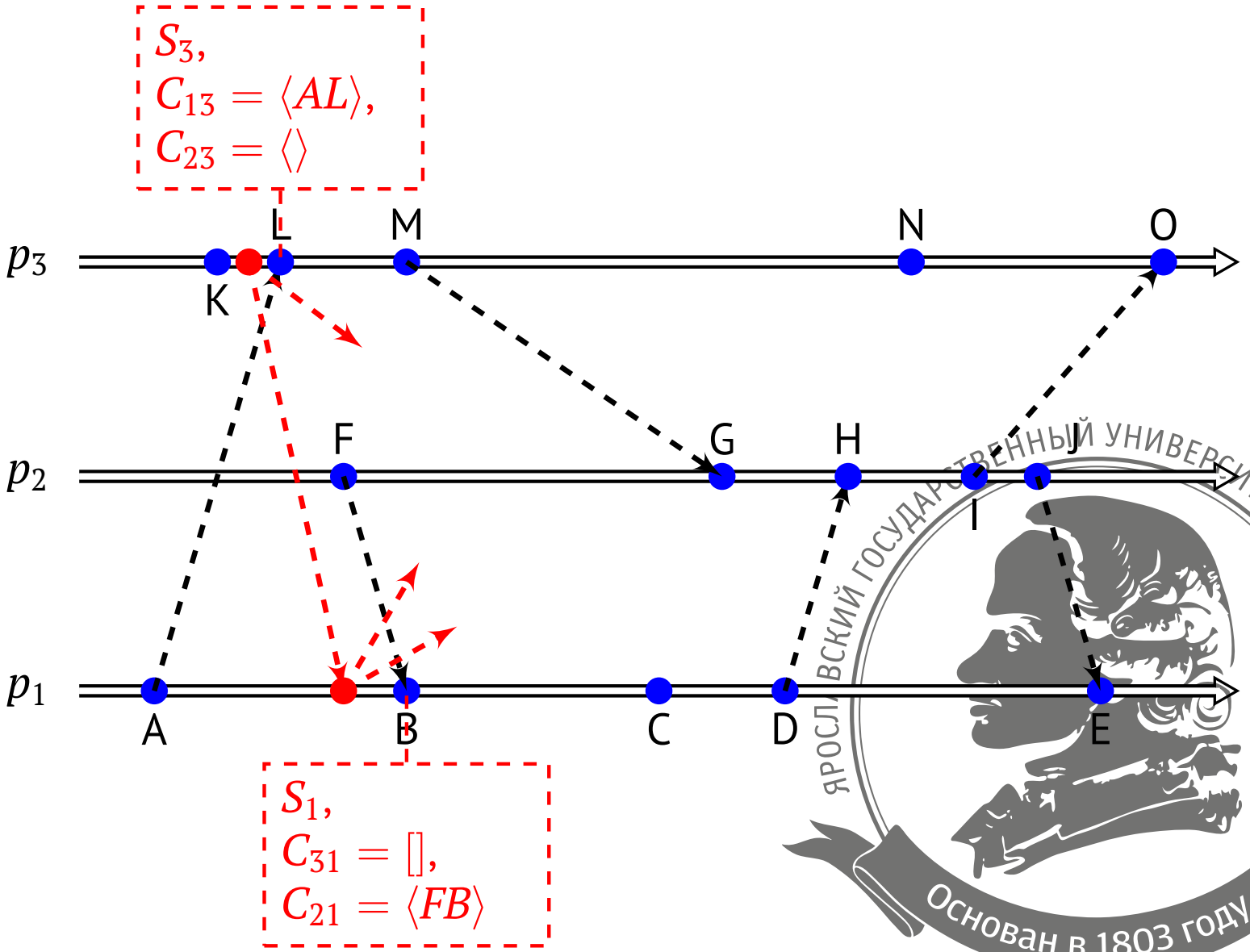
Пример


S_3,
$C_{13} = \langle AL \rangle$,
$C_{23} = \langle\rangle$
p_3
p_2
p_1
K
L
M
N
O
F
G
H
I
J
A
B
C
D
E

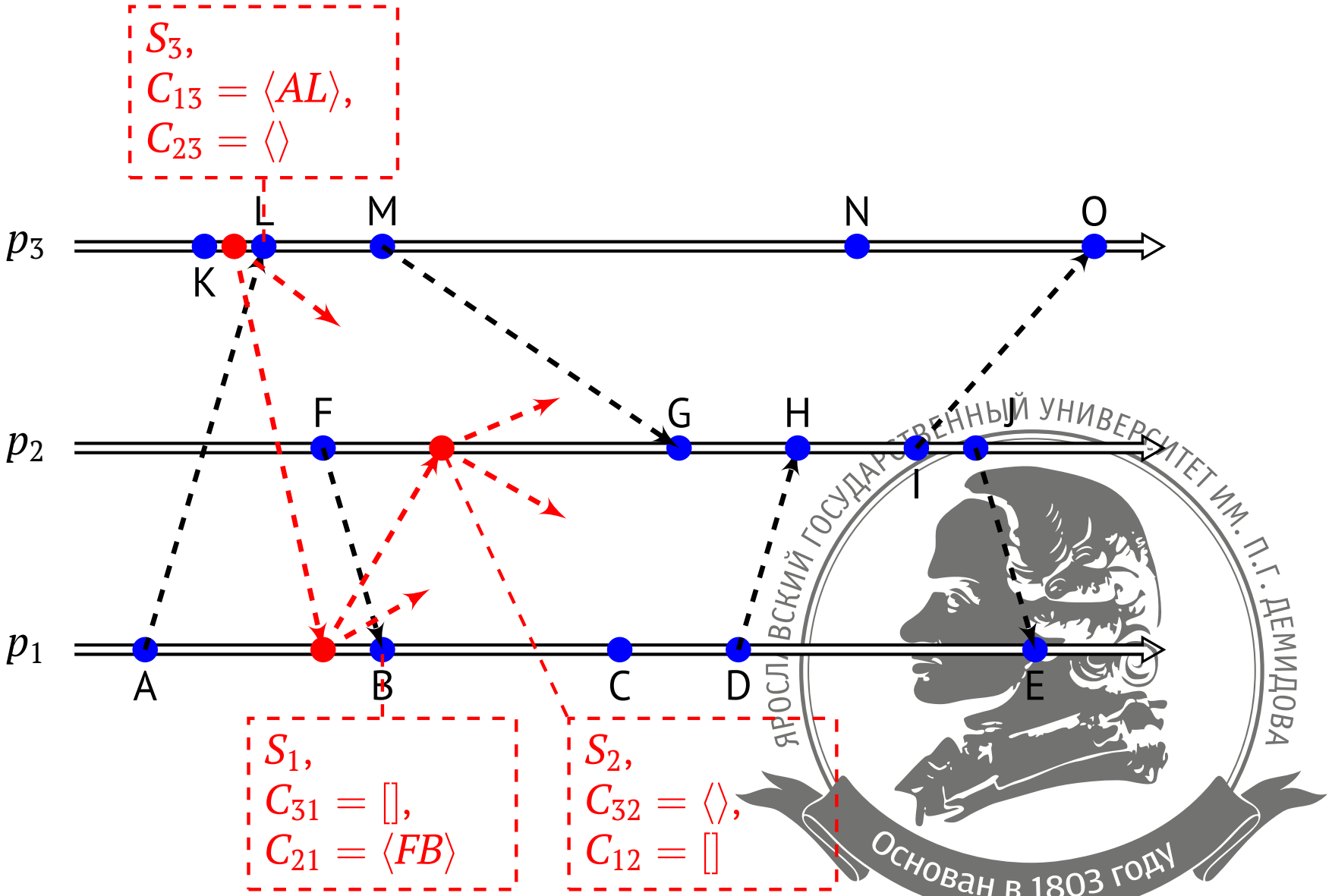
Пример



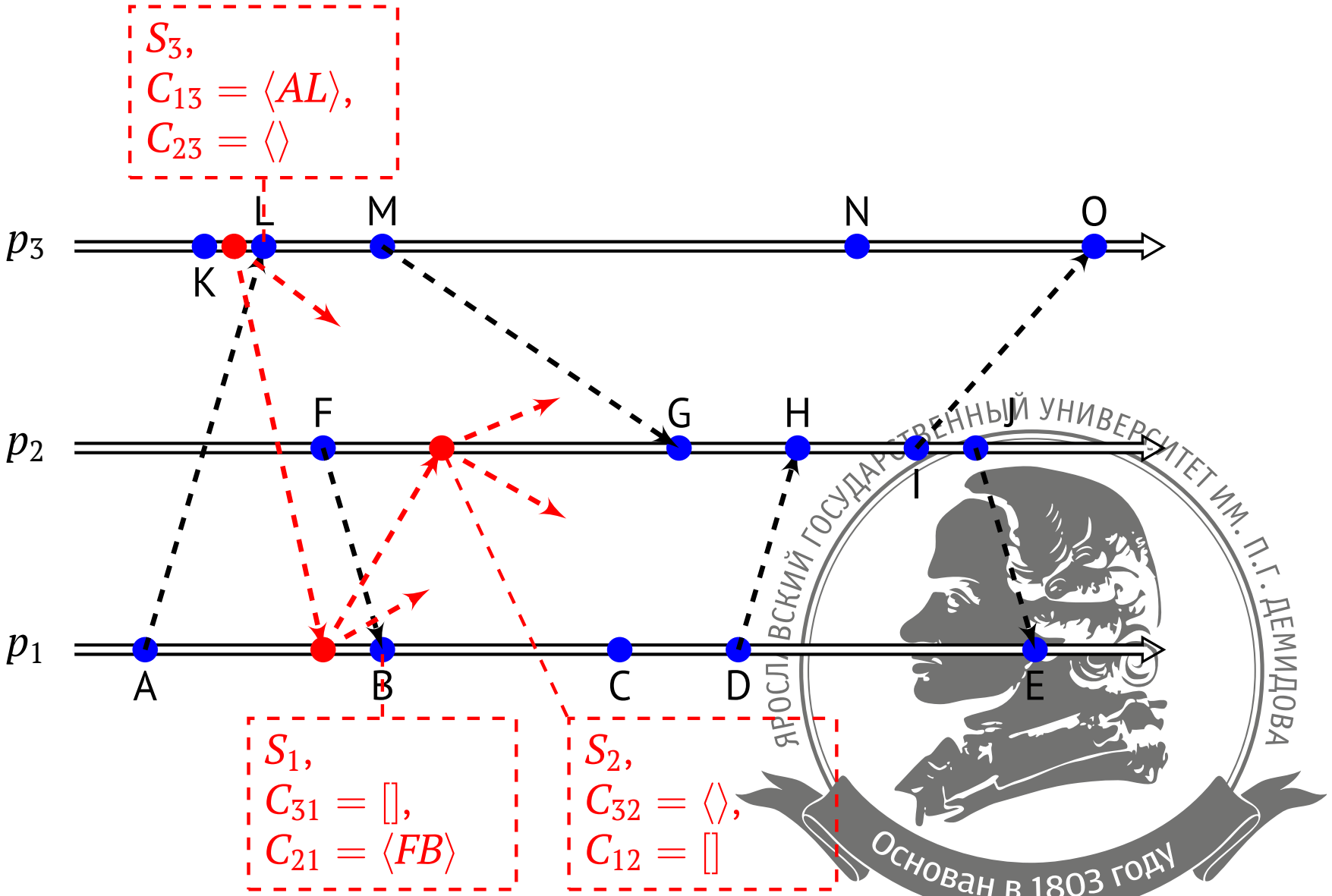
Пример


S_3,
$C_{13} = \langle AL \rangle$,
$C_{23} = \langle\rangle$
p_3
p_2
p_1
K
L
M
N
O
F
G
H
I
J
A
B
C
D
E
S_1,
$C_{31} = []$,
$C_{21} = \langle FB \rangle$

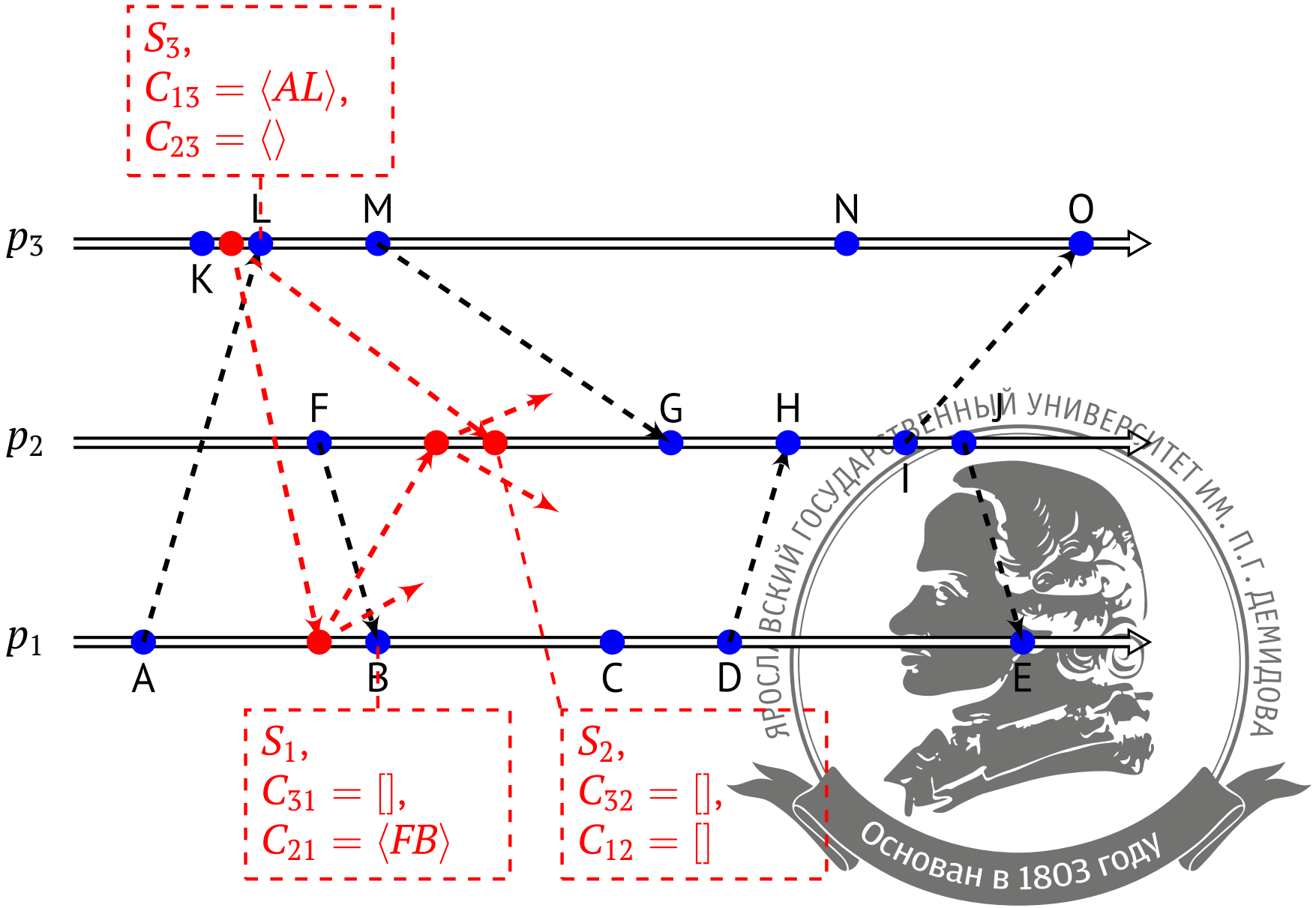
Пример


S_3,
$C_{13} = \langle AL \rangle$,
$C_{23} = \langle\rangle$
p_3
p_2
p_1
K
L
M
N
O
F
G
H
I
J
A
B
C
D
E
S_1,
$C_{31} = []$,
$C_{21} = \langle FB \rangle$
S_2,
$C_{32} = \langle\rangle$,
$C_{12} = []$

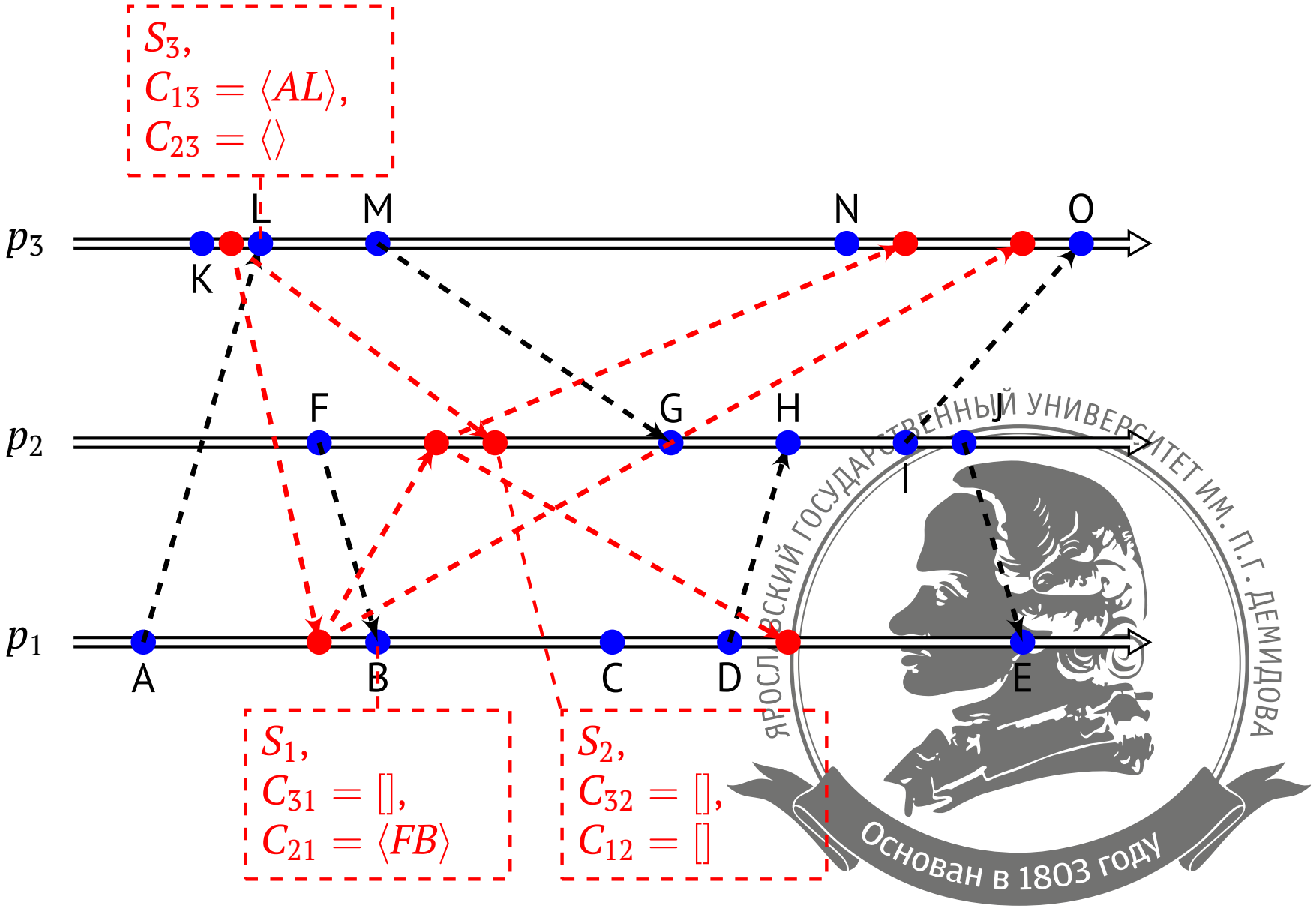
Пример


S_3,
$C_{13} = \langle AL \rangle$,
$C_{23} = \langle\rangle$
p_3
p_2
p_1
K
L
M
N
O
F
G
H
I
J
A
B
C
D
E
S_1,
$C_{31} = []$,
$C_{21} = \langle FB \rangle$
S_2,
$C_{32} = \langle\rangle$,
$C_{12} = []$

Пример



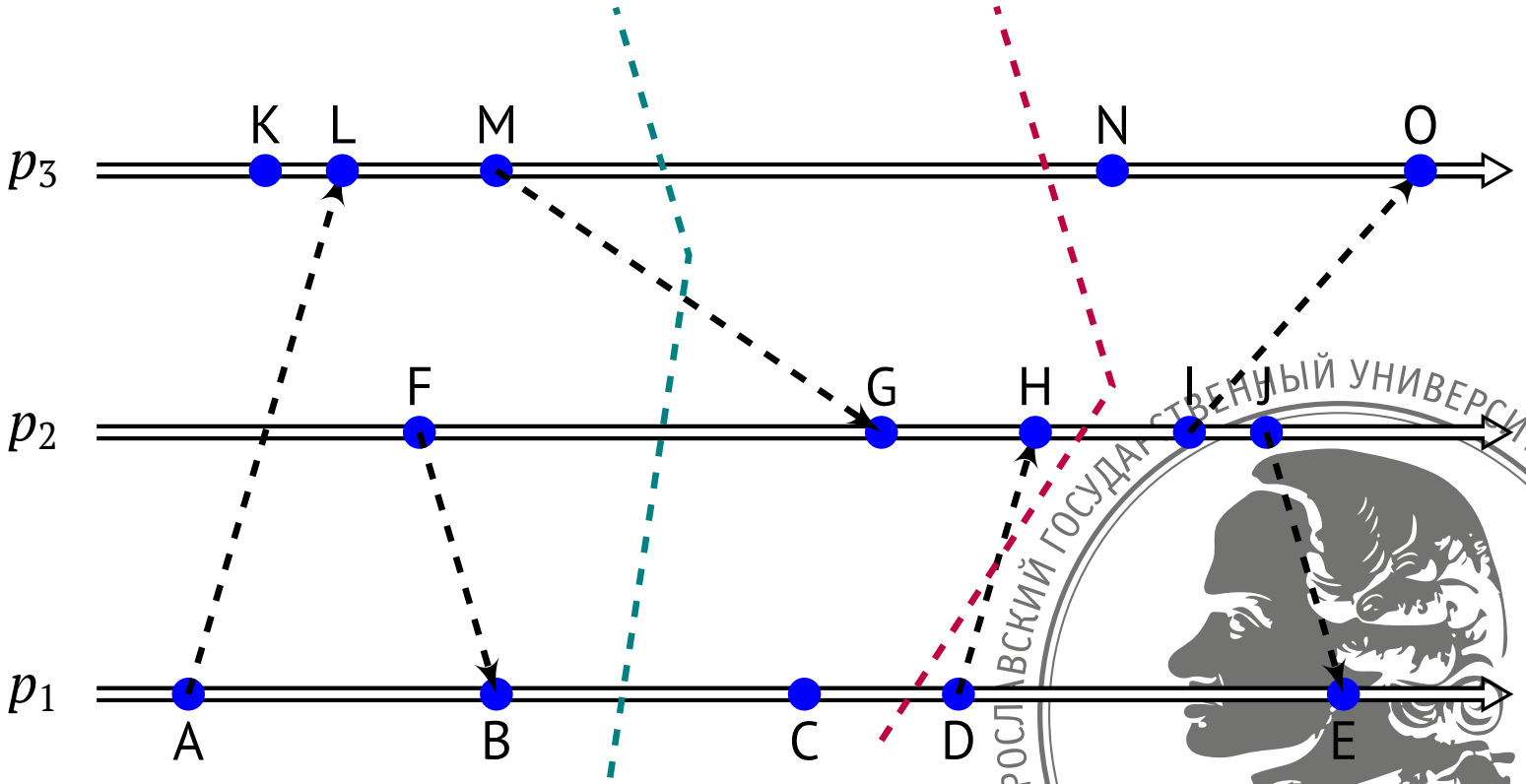
Пример



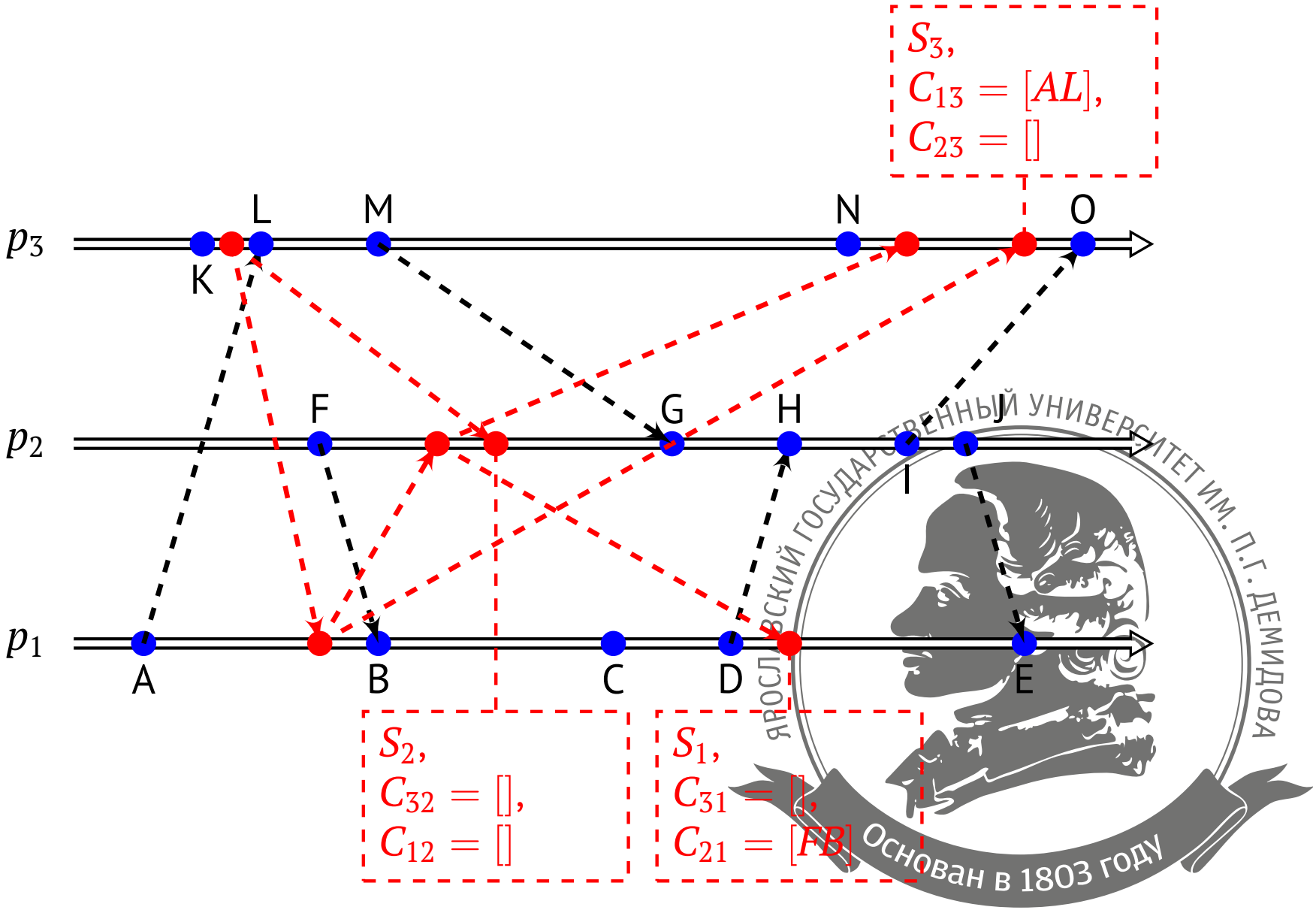
Целостный разрез

- Разбиваем множество событий распределенной системы на два подмножества;
- если событие e входит в разрез и $f \rightarrow e$, то f также входит в разрез;
- целостный разрез не нарушает каузальность событий.

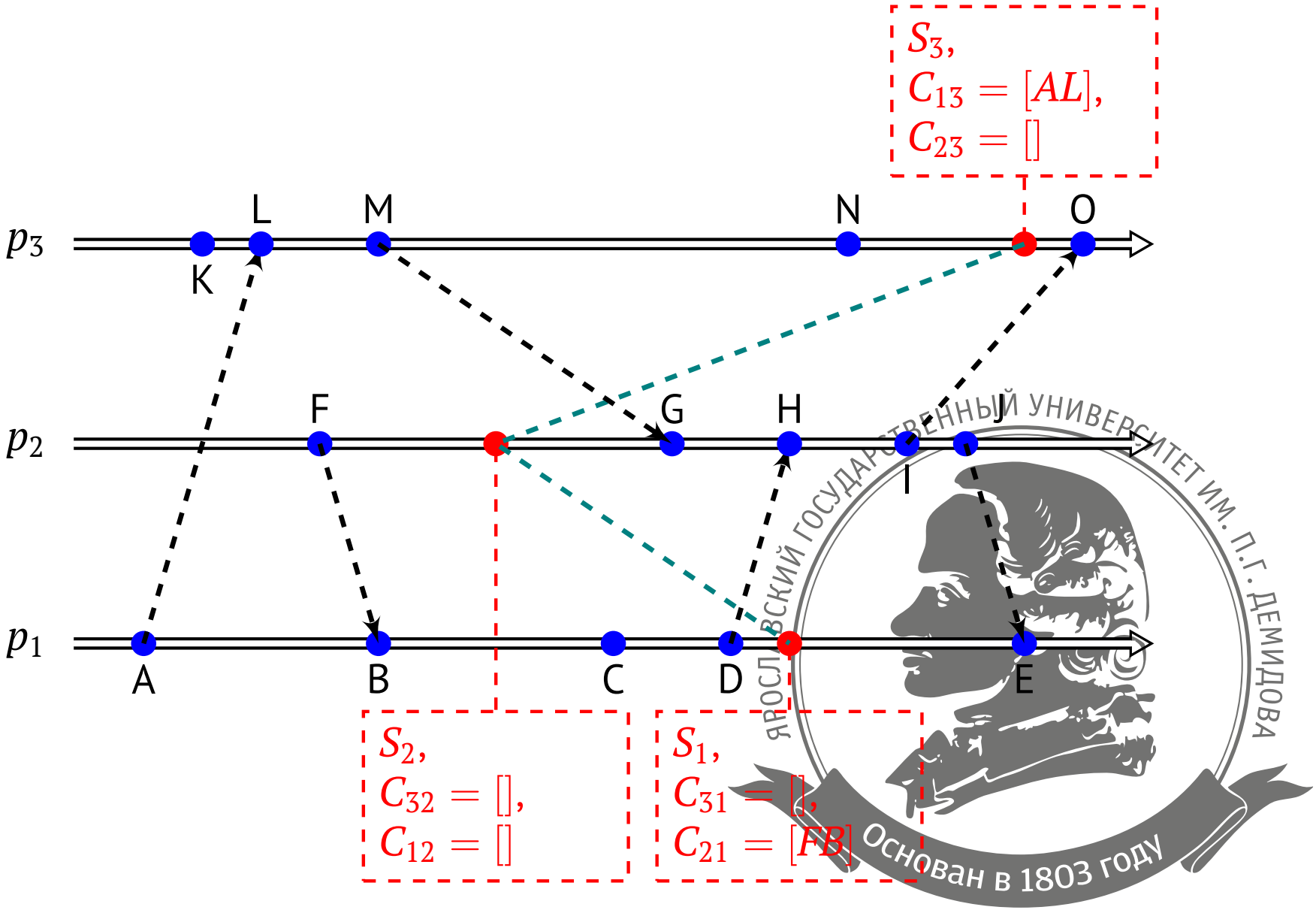
Пример



Наш пример создания образа



Наш пример создания образа



Корректность алгоритма Ченди-Лэмпорта

Алгоритм создает согласованный разрез