LEANCLOUD 内部分享 - 江宏

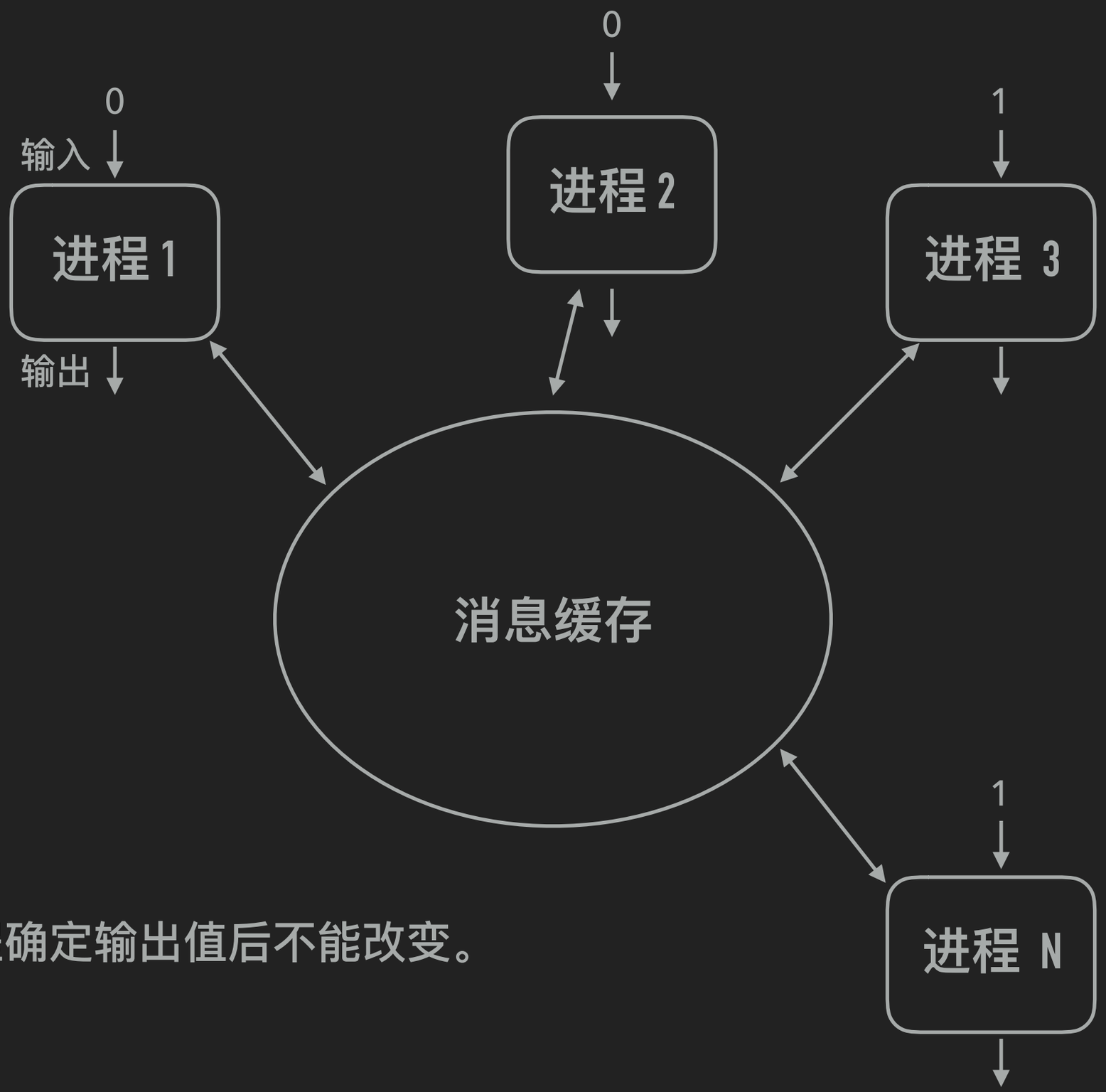
异步系统的一致性问题

一致性问题

- 网络中多个进程就特定决策达成共识
  - 分布式数据库的事物提交
  - 领导选举
- 在可靠的系统中容易做到
- 在不可靠的系统中是过去几十年热门的研究话题

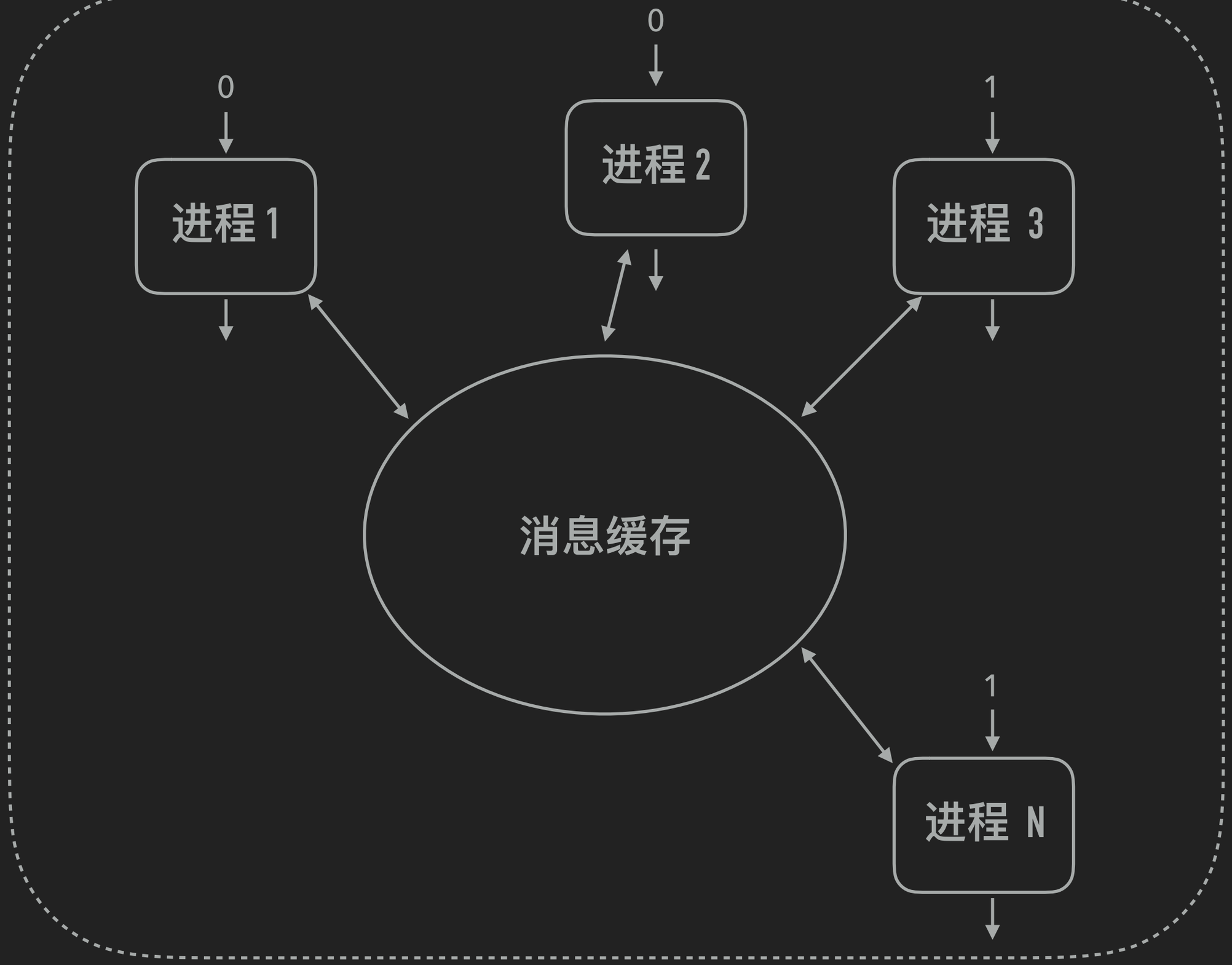
分布式系统中的错误

- 链接的错误
- 进程的错误
  - 崩溃错误
- 拜占庭错误



每个进程确定输出值后不能改变。

整个系统的一个全局状态被称为一个配置



每个进程 p 按以下的方式运行

1. 接收消息：receive(p)，从消息缓存中得到一个消息 (p, m) 并把它从消息缓存删除，或者得到空消息；
2. 在本地进行计算；
3. 给若干个节点发消息：send(p′, m′), send(p″, m″), …；
4. 决定是否输出结果；
5. 回到第 1 步。

异步系统

- 各进程以不同速度运行，没有共同的时钟；
- 消息传递可能存在任意延迟，但消息不会丢失。
- 如果有一条发送给进程 p 的消息 (p, m) 被发出，只要 p 不停接收消息，(p, m) 一定会被收到。

配置和事件

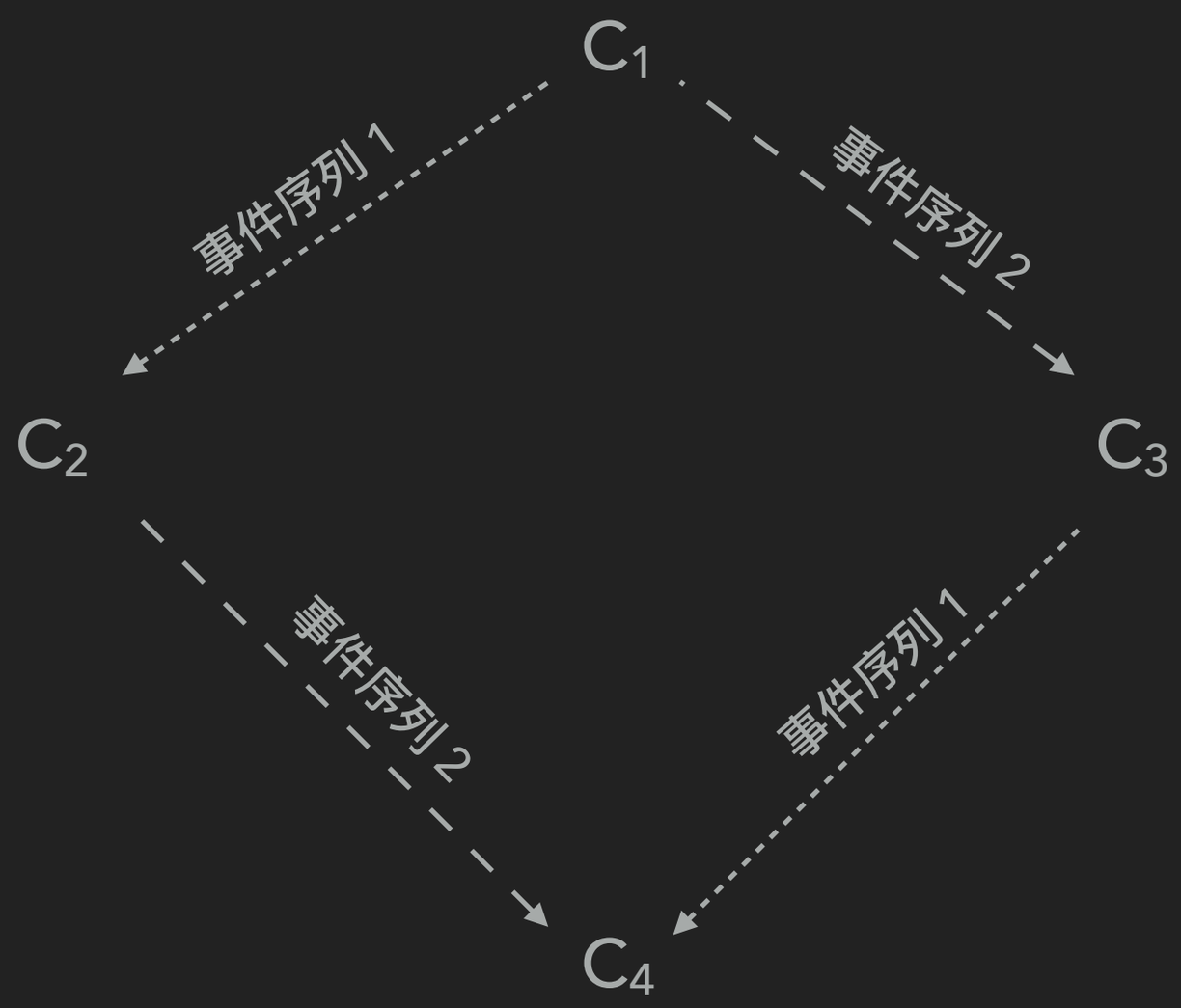
- 各进程的输入、状态，以及消息缓存的内容被称为这个系统的配置；
- 一个进程收到一条消息以及引起的相关状态改变被称为一个事件；
- 因为一个事件的发生，系统从一个配置进入到另一个配置。
- 一个算法的一次运行对应于一个事件的序列。

一致性问题

- 每个进程有一个为 0 或 1 的输入；
- 要求在有限个事件之后，有至少一个进程输出一个结果；
- 对于所有输出结果的进程，他们的结果必须一致；
- 结果必须是某个进程的输入。

- 在有一个进程可能崩溃的情况下，是否有算法可以满足要求?

定理 1：如果两个事件序列所涉及的进程无交集，他们的顺序可以互换，最终到达同一个配置。



假设一致性算法存在：如果从一个配置开始运行，最终的输出可能为 0 也可能为 1，这个配置被称为二价的，否则这个配置被称为一价的。

定理 2：一定存在一个二价的初始配置。

0 0 0 0 0 0 0 0 0 0 0 0 0 ⟶ 0

……

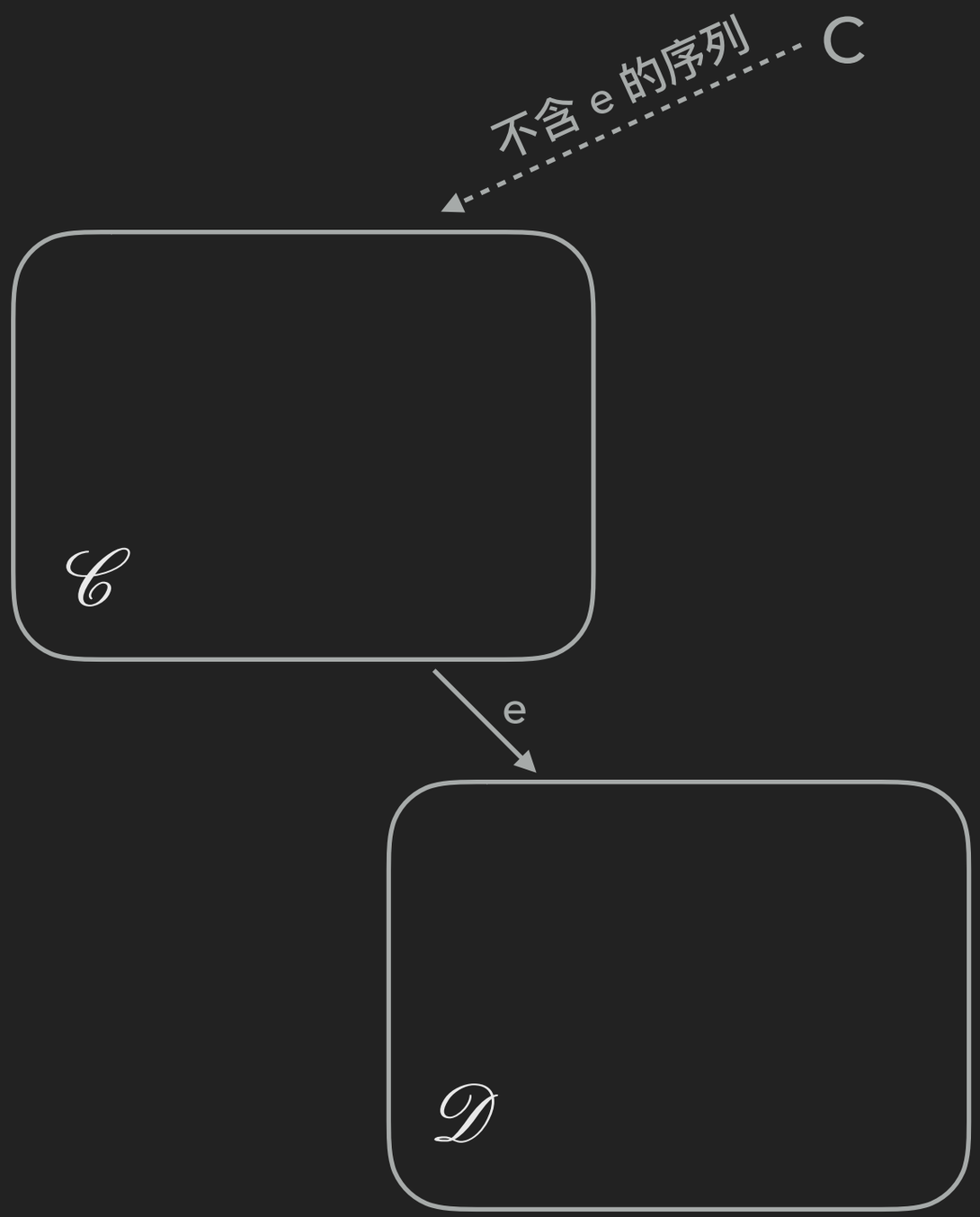
0 0 0 0 0 0 0 0 1 1 1 1 1 ⟶ 0?

0 0 0 0 0 0 0 1 1 1 1 1 1 ⟶ 1?

……

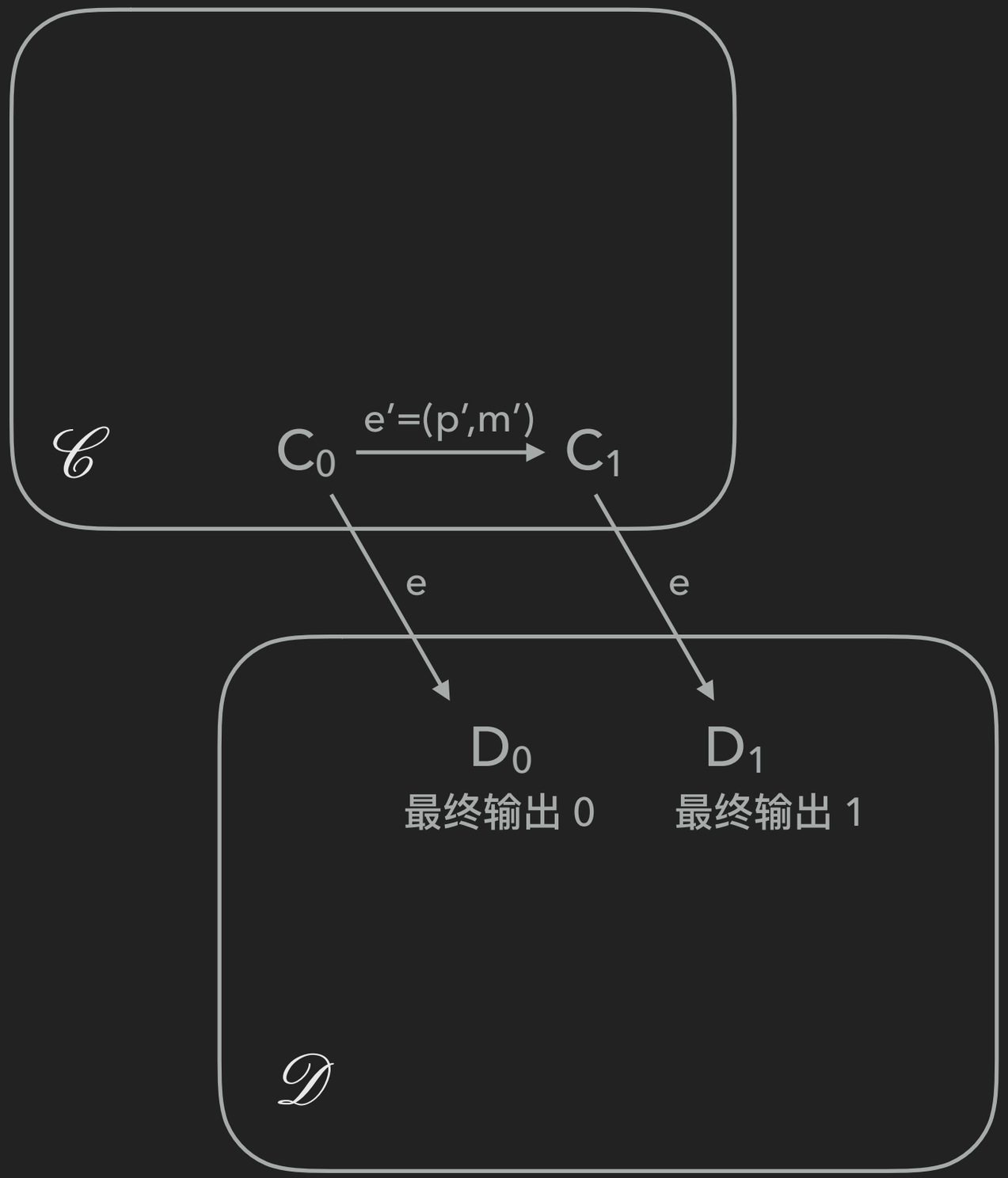
1 1 1 1 1 1 1 1 1 1 1 1 1 ⟶ 1

定理 3：对于一个二价的配置 C 和一个事件 e=(p, m)，存在一个以 e 结尾的事件序列使得从 C 开始执行这个序列会到达另一个二价的配置 D。

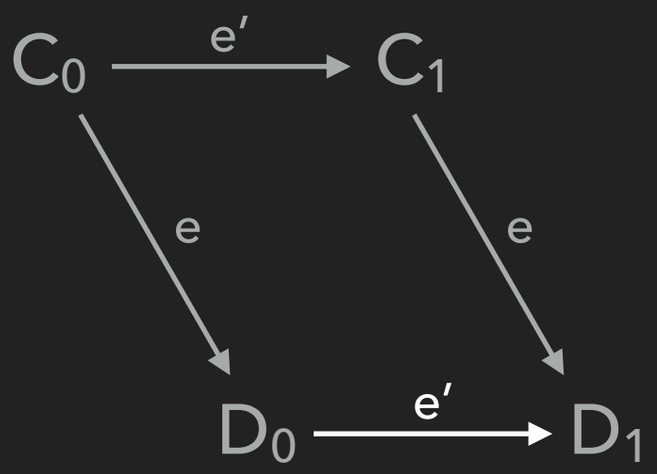


先证明：$\mathscr{D}$ 既含有会输出 0 的配置，

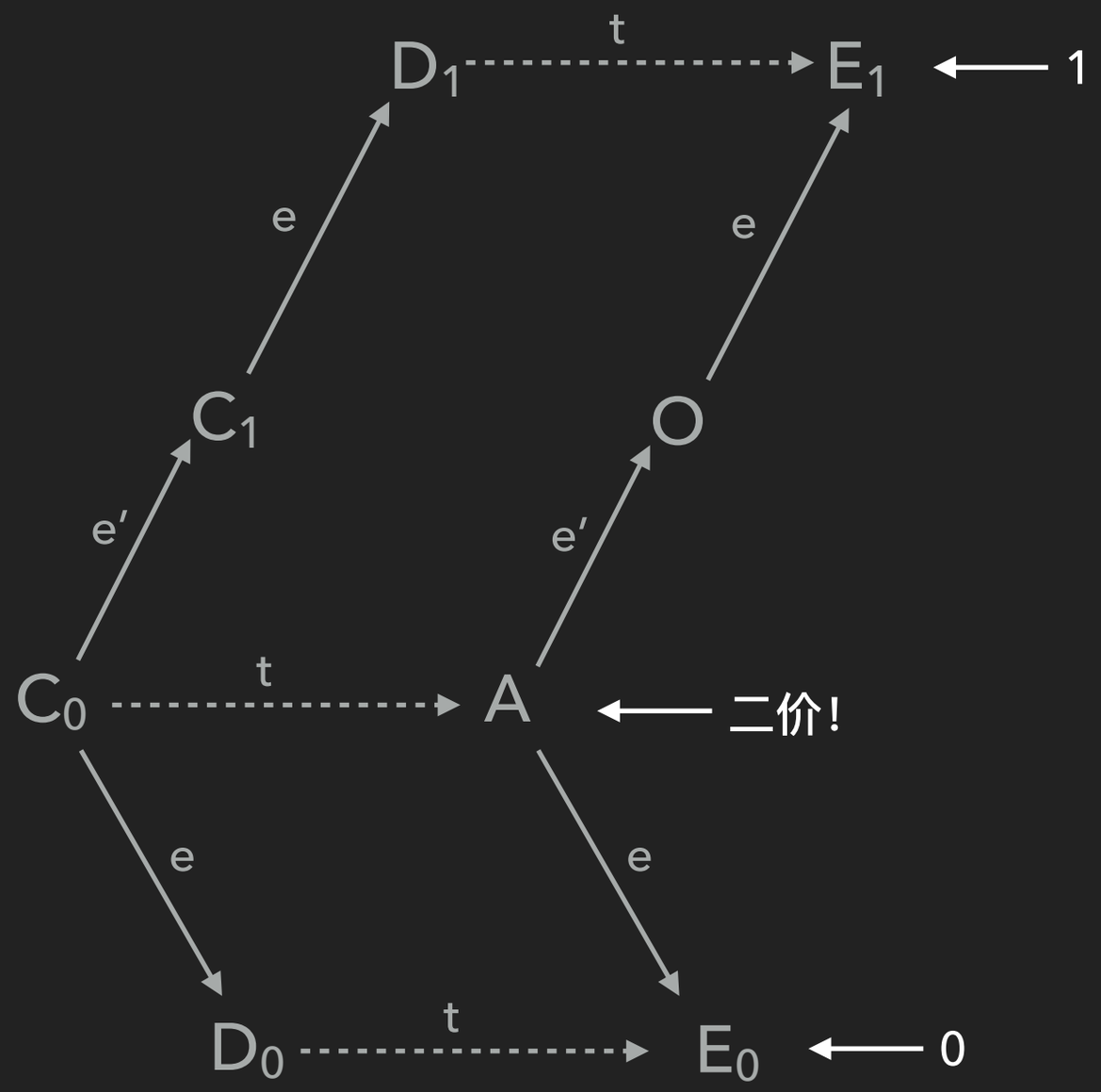
也含有会输出 1 的配置。


$\mathscr{C}$
C_0
e′=(p′,m′)
C_1
e
e
D_0
最终输出 0
D_1
最终输出 1
$\mathscr{D}$

如果 p != p′



如果 p = p′，即使 p 在 C_0 后崩溃不再参与，系统也要能到达一个一价的配置，假设 t 是这样的一个没有 p 参与的序列。



主要结论

- 任何从一个二价的初始配置运行到有一致输出的过程中都会有一个关键事件，它让系统从一个二价配置进入一个一价配置。

- 根据定理 3，我们可以通过在这个关键事件前面插入有限个其它的事件而构造出一个合法并让系统保持在二价配置的序列。

- 所以对于任何算法，都存在至少一个合法的运行过程，使得系统一直在二价配置而不会达到一致的输出。

那么什么样的容错是可能的?

- 随机算法：以 1 的概率达到一致。
- 增加同步机制：时钟、超时等。
- 对错误的进一步限制：比如崩溃只能出现在运行开始。
- 只要求最终一致：允许进程修改输出。