

Petri Net

automata 是一个理论的理想的模型

petri net 相对的更适合工程实现

- state is distributed, transitions are localised
- local causality replaces global time
- subsystems interact by explicit communication

PN 中的状态迁移是异步的，因此可以用来建模并发分布式系统

Definition

PN 的图形化表示可以看作一个二分图 (bipartite graph)，顶点分为两类

- places：用于建模系统中的资源或状态
- transitions：用于建模系统中的状态迁移和同步

图中的边 (arc) 总是连接两种不同的顶点

Token 是位于 places 中的资源

更为正式的定义为

PN 是一个 4-tuple $C = (P, T, I, O)$ 其中

- P 是 places 的集合
- T 是 transitions 的集合
- $I : T \rightarrow 2^P$ 是 transition 的输入，记为 $\cdot t$
- $O : T \rightarrow 2^P$ 是 transition 的输出，记为 $t \cdot$

由于 token 的存在，故可以定义 marking $\mu = (\mu_1, \mu_2, \dots, \mu_n)$ 为各个 place 中 token 的数量

Basics of PN

Fire

对于一个 transition t ， t 在当前 marking 为 **enabled** \iff 对于 t 的所有 input，其中都存在一个 token

一个 enabled transition 即可 **fire**，即从其所有 input 中减去一个 token，再在其所有 output 中增加一个 token

Fire 是一个原子操作

PN 中的边可以带有权重，对于边带权的 PN，一个 transition 被 enabled 要求其所有 input 中的 token 数都大于等于对应边的权重

PN 可以为多种现实中的行为建模

- 顺序执行：当前 transition 被 enabled 只能在前一个 transition fire 之后
- 同步：所有 input 中都有 token (resource) 后当前 transition 才能 fire
- merge&fork：多个 resource 汇总到一个 place 或是 resource 从一个 place 分发到多个 place
- 并发：多个 enabled transition 可以各自独立地 fire
- conflict：多个 enabled transition 在其中一个 fire 之后其余便不处于 enable 状态

Run

PN 的 run 是一个有限/无限的 marking 和 transition 的序列

$$\mu_0 \xrightarrow{t_0} \mu_1 \xrightarrow{t_1} \cdots \xrightarrow{t_{n-1}} \mu_n \xrightarrow{t_n} \cdots$$

其中 μ_0 是 PN 的初始 marking，而 $t_i \in \text{enabled}(\mu_i)$ ，且

$$\mu_i = (\mu_{i-1} - \cdot t_{i-1}) + t_{i-1} \cdot, \forall i \geq 1$$

上式的含义是从当前 marking 减去所有 t 的 input 再加上所有 t 的 output 即是 fire 之后的 marking

Other definitions

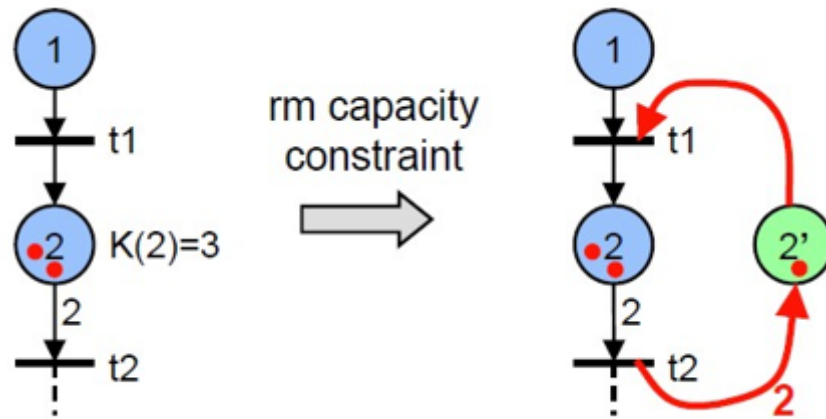
- source transition: no input
- sink transition: no output
- self-loop: a pair (p, t) s. t. $p \in \cdot t \wedge p \in t \cdot$
- pure PN: no self-loops
- weighted PN: arcs with weight
- ordinary PN: all arcs weights are 1
- infinite capacity net: places can accommodate an unlimited number of tokens
- finite capacity net: each place p has a maximum capacity $K(p)$
- strict transition rule: after firing, each output place can't have more than $K(p)$ tokens

则有定理：任何 pure finite-capacity PN 可以被转换为等价的 infinite-capacity

具体做法为：

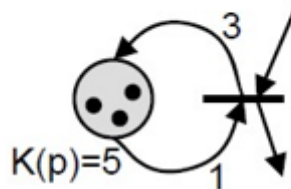
- 对任意 $p, K(p) > 1$, 添加一个 place p' , 其中初始 marking $M_0(p') = K(p) - M_0(p)$
- 对任意 output arc $e = (p, t)$, 添加 $e' = (t, p')$ 且权重为 $W(e)$
- 对任意 input arc $e = (t, p)$, 添加 $e' = (p', t)$ 且权重为 $W(e)$

如下图



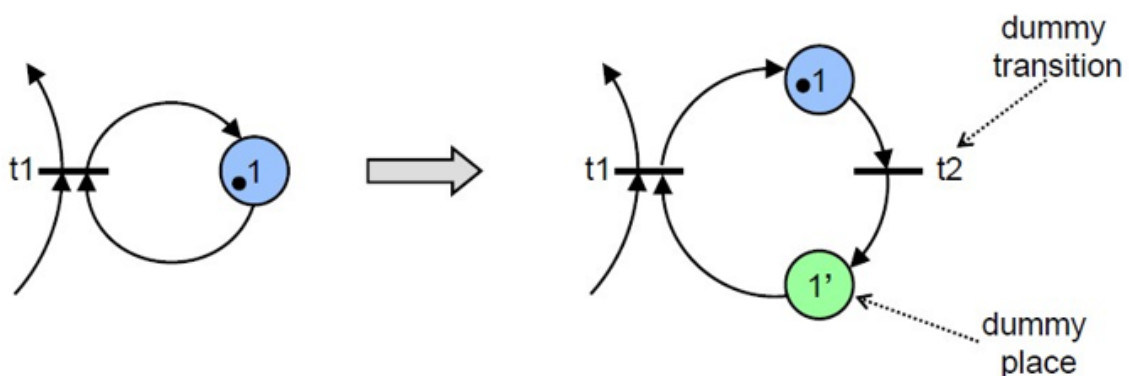
新添加的 p' 满足了在 p, p' 中的 token 数之和恒定且等于 $K(p)$, 但是对于有 self-loop 的便不能使用这种方法, 如下图

Note: Only works for **pure** Petri nets, i.e. without self loops.



在添加后变为了不能 fire 的状态

但是对于 non-pure 的 PN 可以通过消除 self-loop 的方式将其变为等价的 pure PN



Behavioral properties

Behavioral property 是取决于初始 marking 的属性

- Reachability: 称一个 marking M_n 可达, 如果存在一个从 M_0 到 M_n 的迁移序列
- Boundness: 一个 bounded PN 任意可达 marking 中 place 中的 token 不会超过某个上限。称 1-bounded 的 PN 为 safe PN
- Liveness: 即任何可达的 marking 都有 transition 可以 fire, 等价于系统无死锁
- Reversibility: 任意可达的 marking 都可达初始 marking。对于 reversibility 有一个较为宽松的版本, 即存在一个 home state M' 满足任意可达的 M 都可达 M'
- Persistence: 对任意两个 enabled 的 transition, fire 其中一个不会导致另一个 disabled, 即一个 transition 如果 enabled 则会一直持续到其 fire 为止
- Fairness: 每个 transition 连续 fire 的次数是有上限的

Cover tree

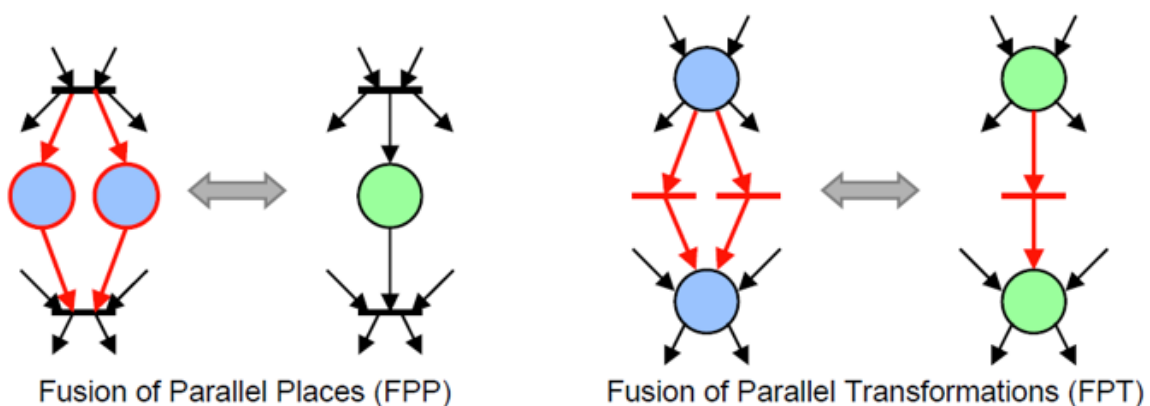
可以通过树的形式表现出所有可能的 marking

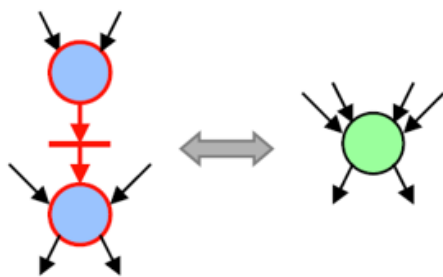
根节点为初始 marking, 节点则是可达的 marking, 通过 transition 的 fire 相连

对于 unbounded PN, 引入 ω 表示任意个数的 token, 则有

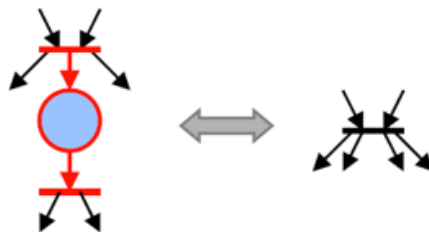
- PN 为 bounded \iff 任何节点中不出现 ω
- PN safe \iff 节点中只有 0, 1
- transition t is dead $\iff t$ 不出现在任何边中

在分析 PN 时代价往往是随着 PN 规模的增大指数级增长的, 故需要化简 PN, 常用的方法有

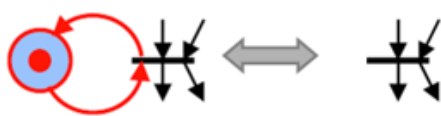




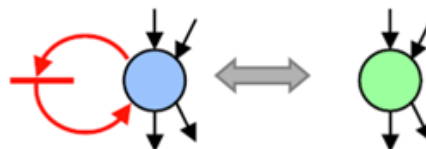
Fusion of Series Places (FSP)



Fusion of Series Transformations (FST)



Elimination of Self Loop Places (ESP)



Elimination of Self Loop Transitions (EST)

PN with time

Time PN 是在经典 PN 的基础上，为每个 transition t 分配一个时间区间 $[a; b]$ ，其值与 t 最近一次 enabled 有关

如果 t 是在时间 c enabled 的，则其能 fire 的时间一定位于 $[c + a; c + b]$ ，且 t 或是在 $c + b$ 之前完成 fire 或是在 fire 之前因为另一个 transition 的 fire 而不再 enabled

fire 不需要花费时间

Time PN 的思路很符合常识：一个 Transition 在 enabled 之后不一定会立刻 fire，但一定会在某段时间过后 fire

time PN 的定义为

time PN 是一个 6-tuple $N = (P, T, F, Eft, Lft, \mu_0)$ ，其中

- P 是有限的 place 集合
- T 是有限的 transition 集合
- $F \subset (P \times T) \cup (T \times P)$ 是 flow relation
- $Eft, Lft : T \rightarrow \mathbb{N}$ 是 earliest fire time 和 latest fire time，满足 $\forall t \in T, Eft(t) \leq Lft(t) \leq \infty$
- μ_0 是初始 marking

令 T 为非负实数（代表时间），则 N 中的状态是一个 pair (μ, c) ，其中 μ 是 marking，且 $c : enabled(\mu) \rightarrow T$ 被称为 clock function

对于初始状态 (μ_0, c_0) 满足

$$\forall t \in enabled(\mu_0), c_0(t) = 0$$

则一个 transition t 如果能在状态 (μ, c) 以 δ 的延迟 fire 等价于满足下列条件

- $t \in enabled(\mu)$
- $(\mu - \cdot t) \cap t \cdot = \emptyset$
- $Eft(t) \leq c(t) + \delta$
- $\forall t' \in enabled(\mu), c(t') + \delta \leq Lft(t')$

则 time PN 的 run 是一个 state, transition, delay 的序列

$$\rho = s_0 \xrightarrow{(t_0, \delta_0)} s_1 \xrightarrow{(t_1, \delta_1)} \dots \xrightarrow{(t_{n-1}, \delta_{n-1})} s_n \xrightarrow{(t_n, \delta_n)} \dots$$

满足 s_0 是初始状态, 且对任意 s_i , 通过 transition t_{i-1} 在 delay δ_{i-1} 后 fire 得到

time PN 可以为同步的 fire 添加逻辑上的先后关系