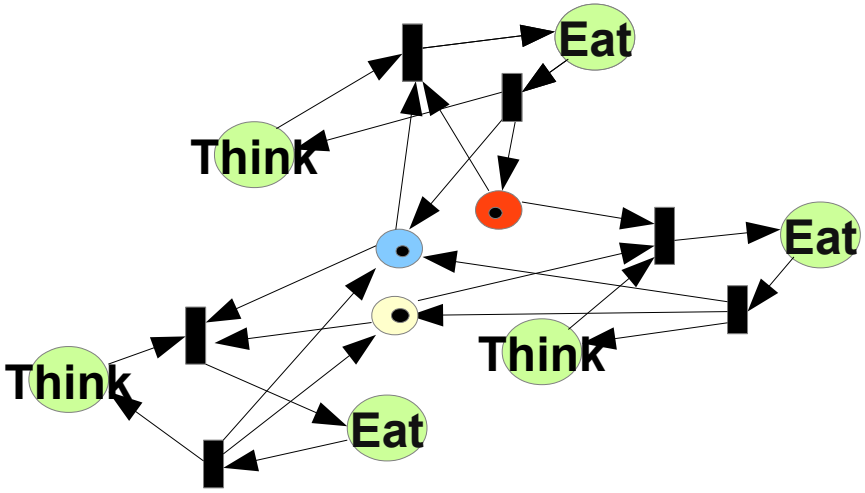
Verification of Petri Nets

Giorgio Delzanno
AILA 2017



Introduzione

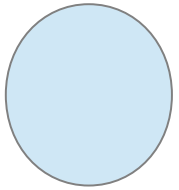
- Le reti di Petri (Petri Net) (*) sono un modello computazionale per sistemi concorrenti basato su grafi

- Usato per modellare sistemi concorrenti asincroni, distribuiti, paralleli, nondeterministici, stocastici
  - Simulazione e analisi di performance
  - Analisi qualitativa (correttezza)

(*) C.A. Petri's dissertation 1962

Nodi

Place:
modellano parti dello stato o risorse



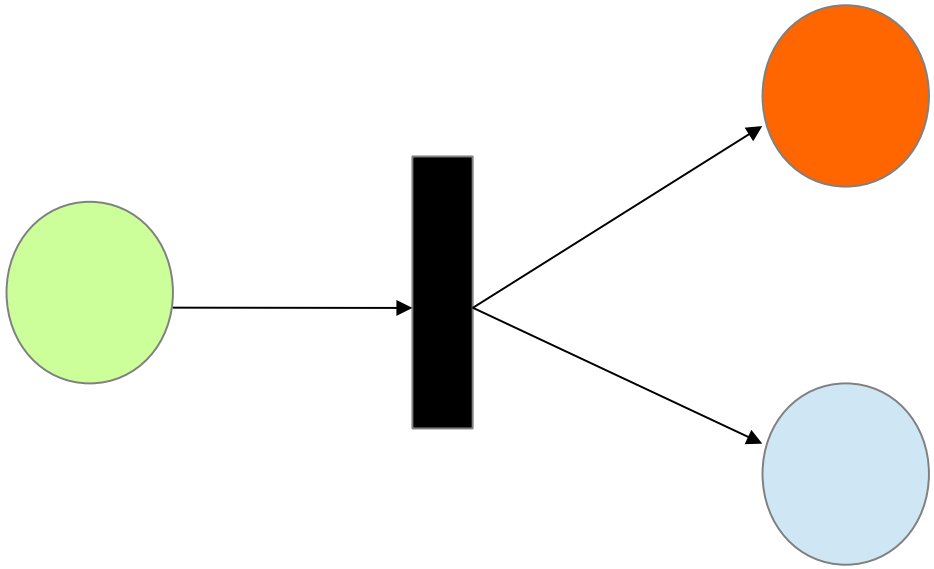
Transition:
modellano transizioni tra stati e sincronizzazione

Archi

Transition

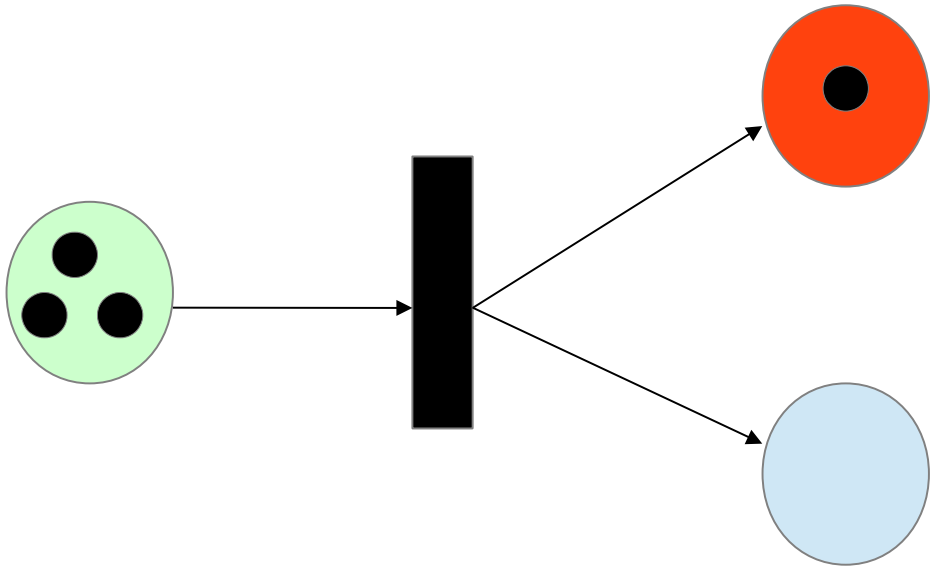
Diretti e collegano nodi di tipo diverso

Definiscono il flusso della rete

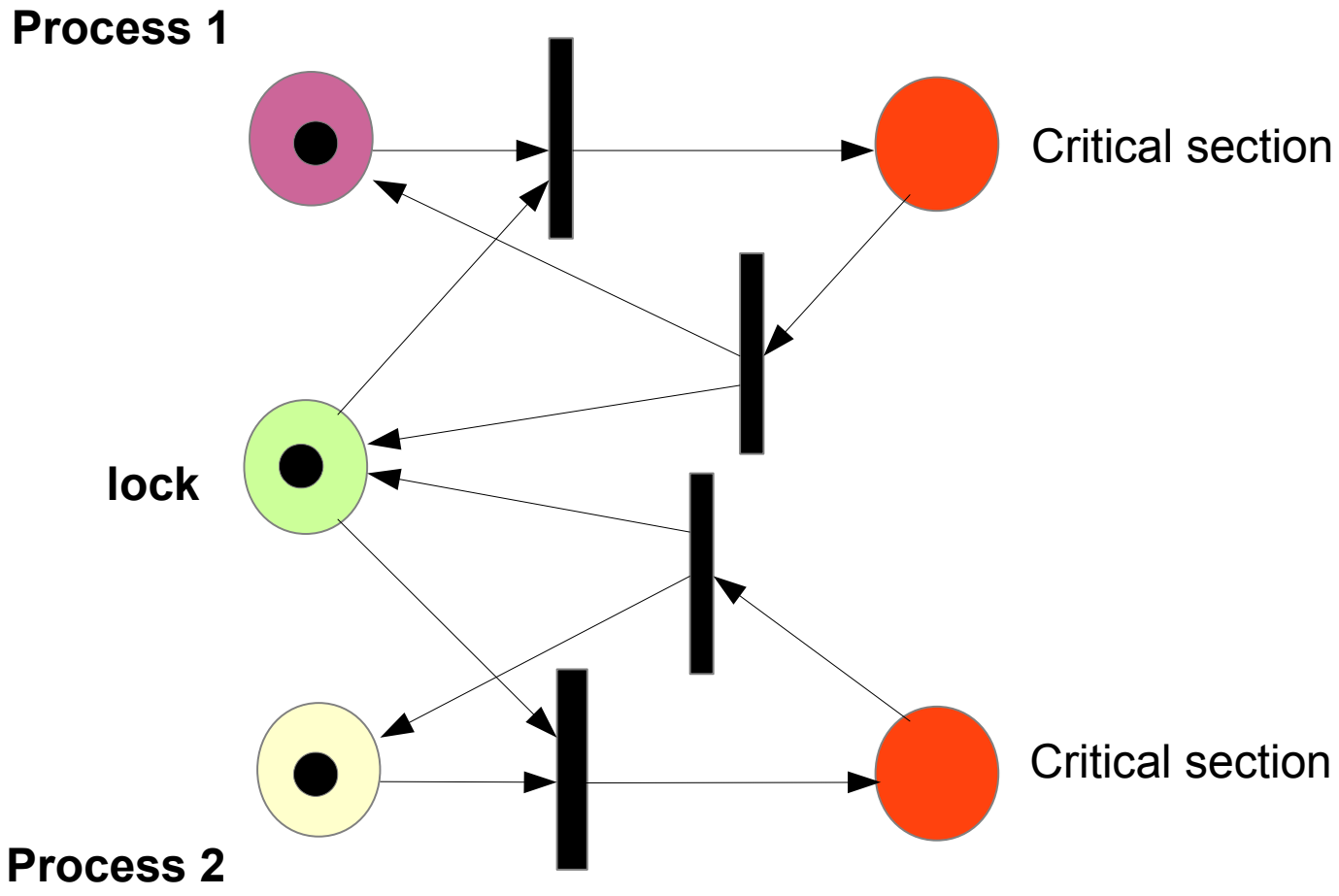


Marking (Stato)

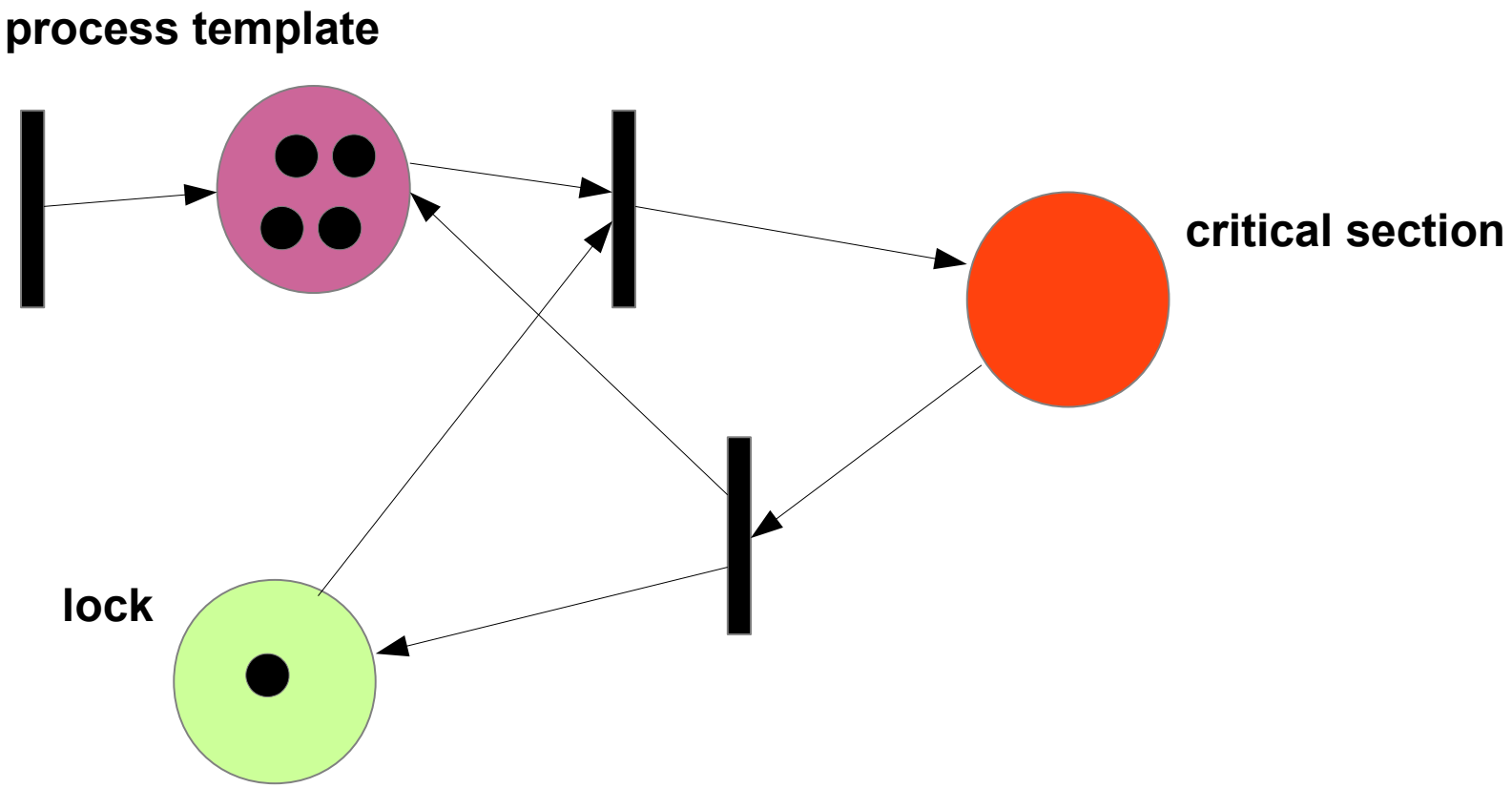
Stato della rete è ottenuto marcando i place con dei token



Esempio: due processi concorrenti



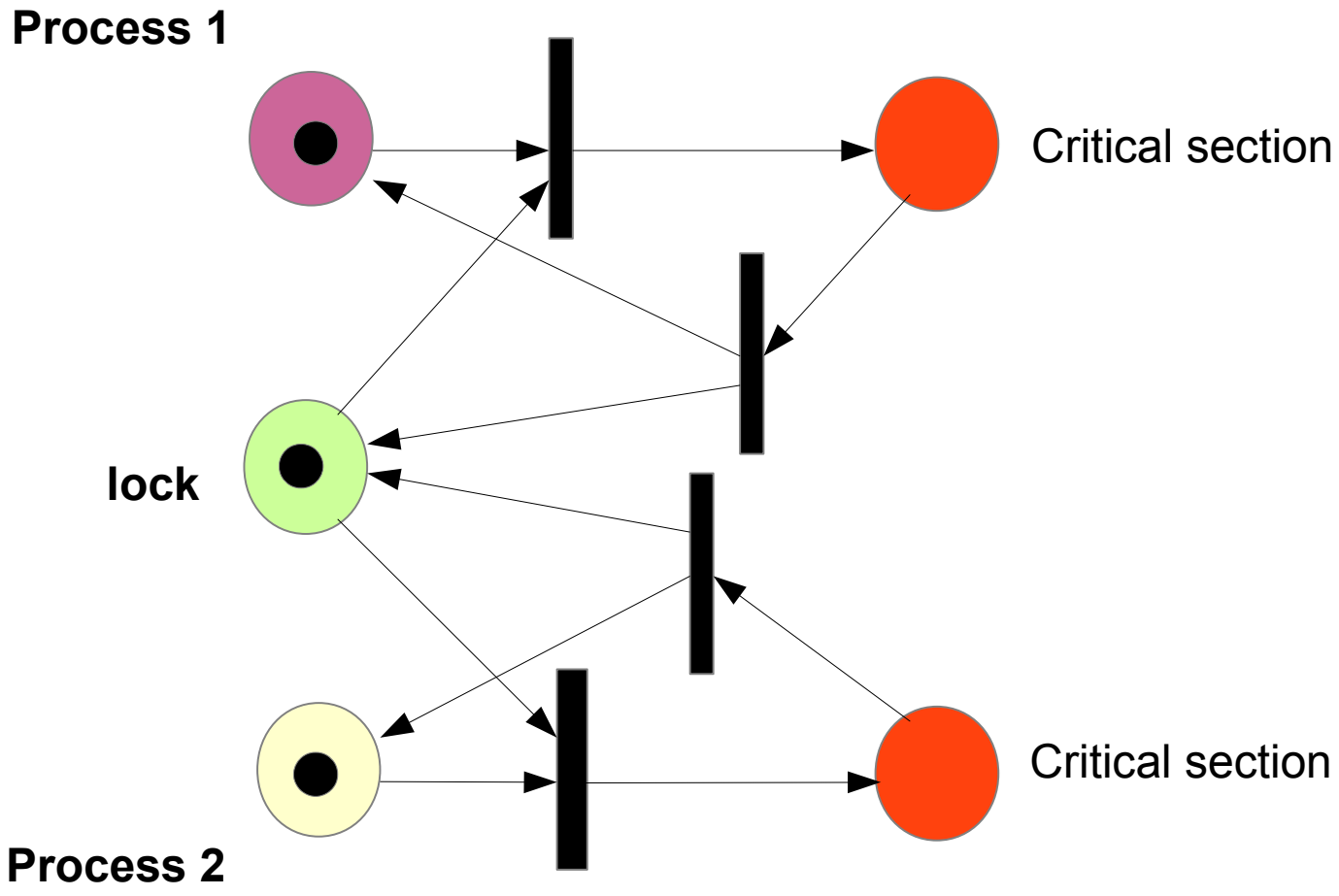
Esempio: N processi concorrenti



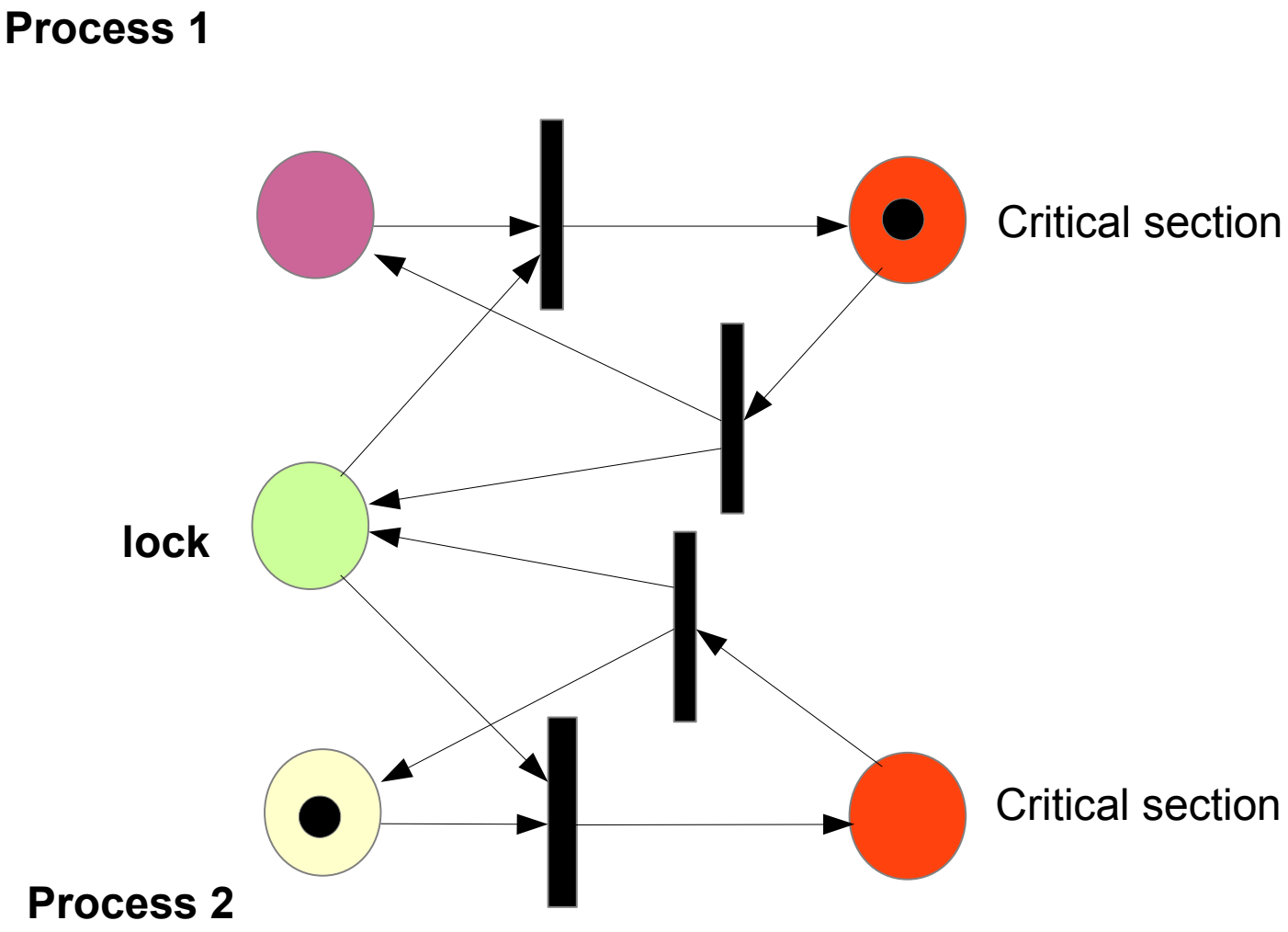
Fire

- Una transizione t è abilitata (*enabled)* in un marking M se:
  - Per ogni arco da un place p alla transizione t, esiste un token distinto dagli altri in M
- Una transizione può essere eseguita (fired) e produrre un nuovo marking
- Il firing di una transizione è un'operazione atomica

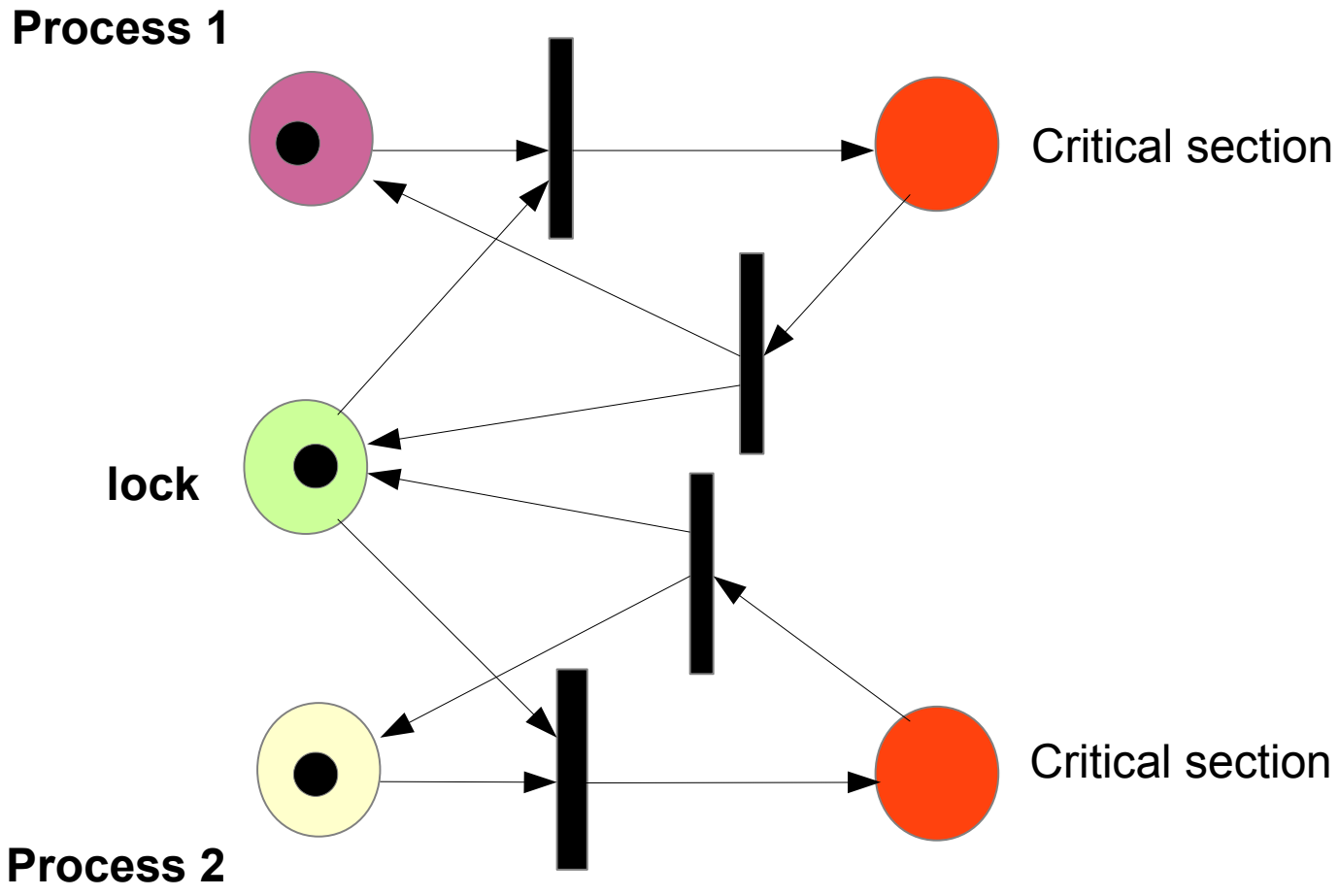
Esempio: due processi concorrenti



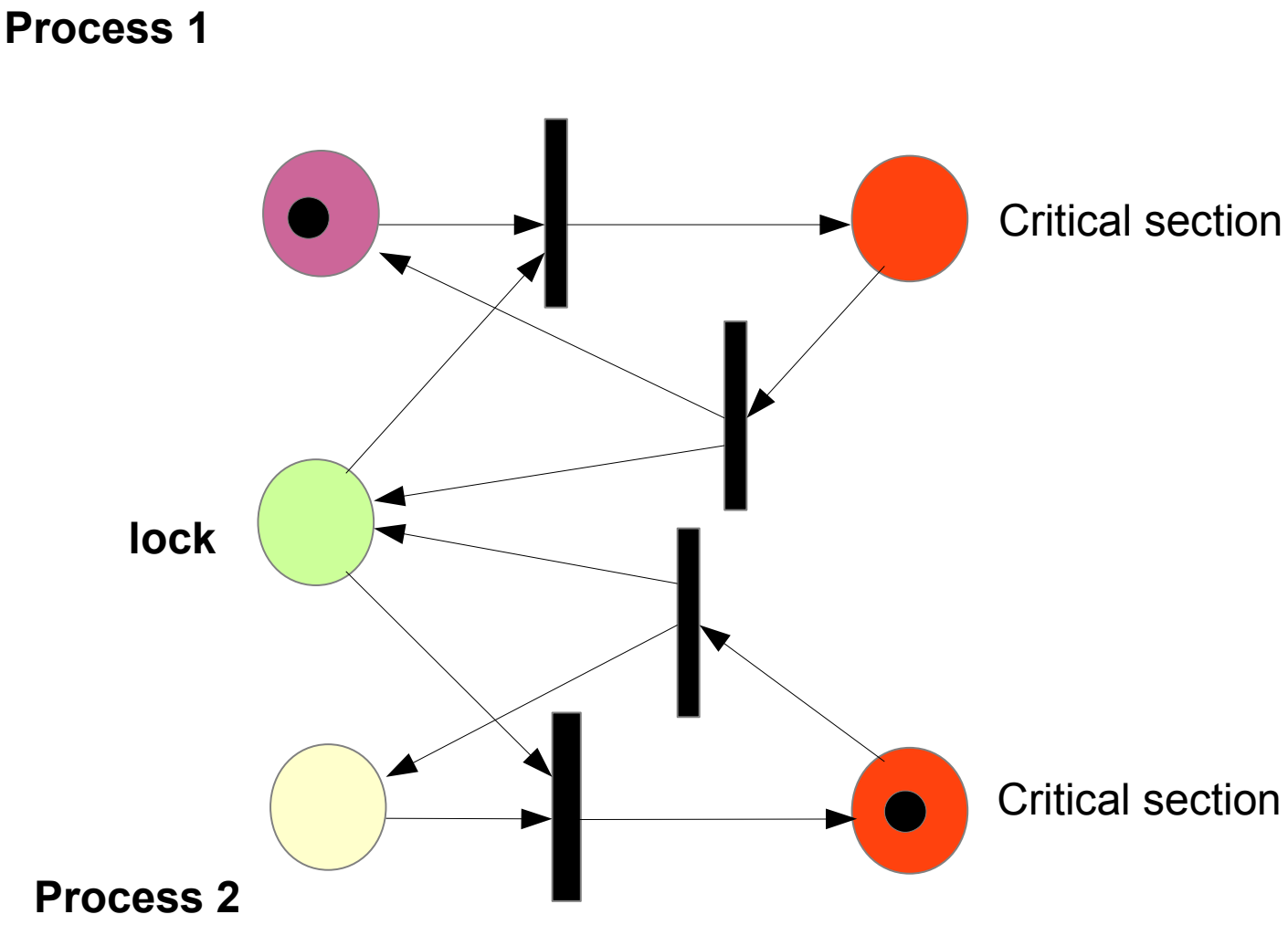
Esempio: due processi concorrenti



Esempio: due processi concorrenti



Esempio: due processi concorrenti



Nondeterminismo

- L'esecuzione dei una rete di Petri è non deterministica
  1. Diverse transizioni possono essere abilitate simultaneamente
  2. Non è necessario che vengano eseguite subito
  3. Una viene eseguita le altre verranno eseguite successivamente

Esempio: Dining Philosophers (semplificato)

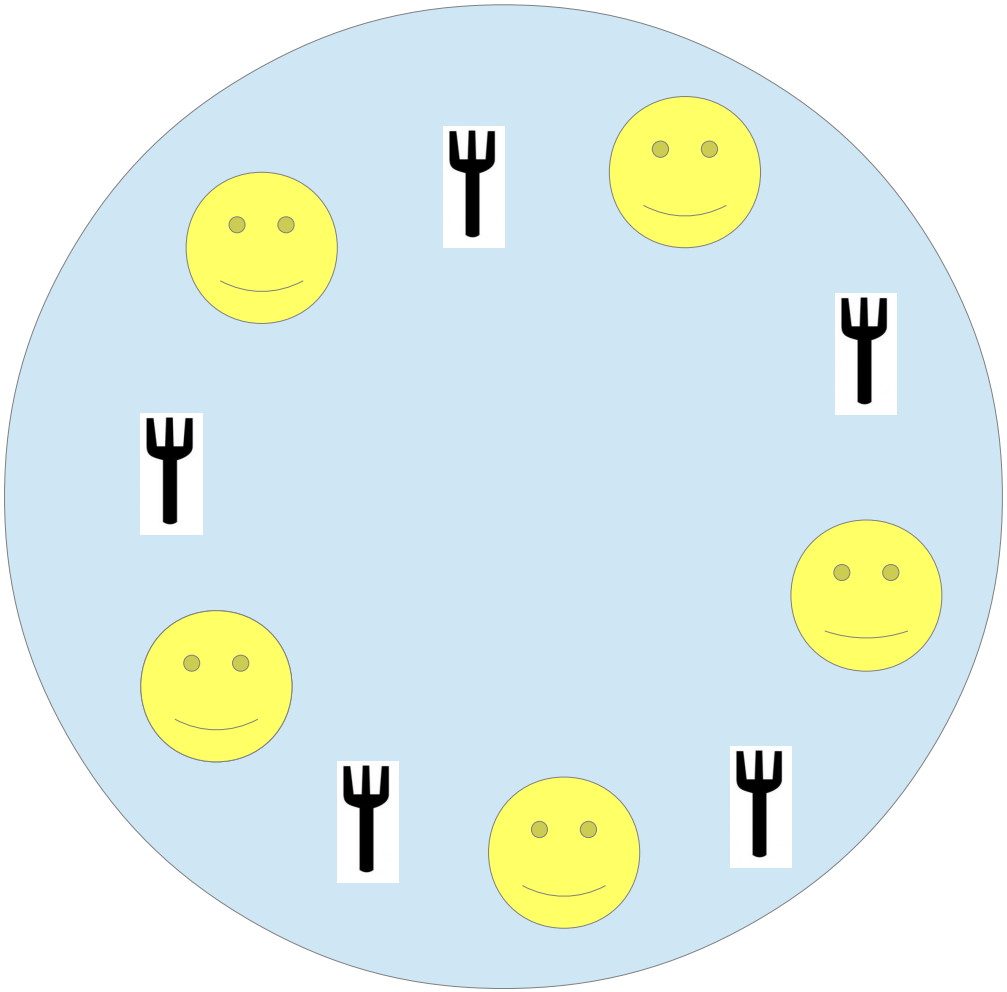
Cinque filosofi siedono ad una tavola rotonda con un piatto di spaghetti davanti, una forchetta a destra e una forchetta a sinistra (bastoncini cinesi)

Ciascun filosofo ha bisogno di due forchette per mangiare. Le forchette vengono prese una per volta.

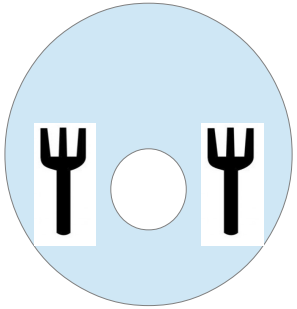
Dopo essere riuscito a prendere due forchette il filosofo mangia per un po', poi lascia le forchette e ricomincia a pensare.

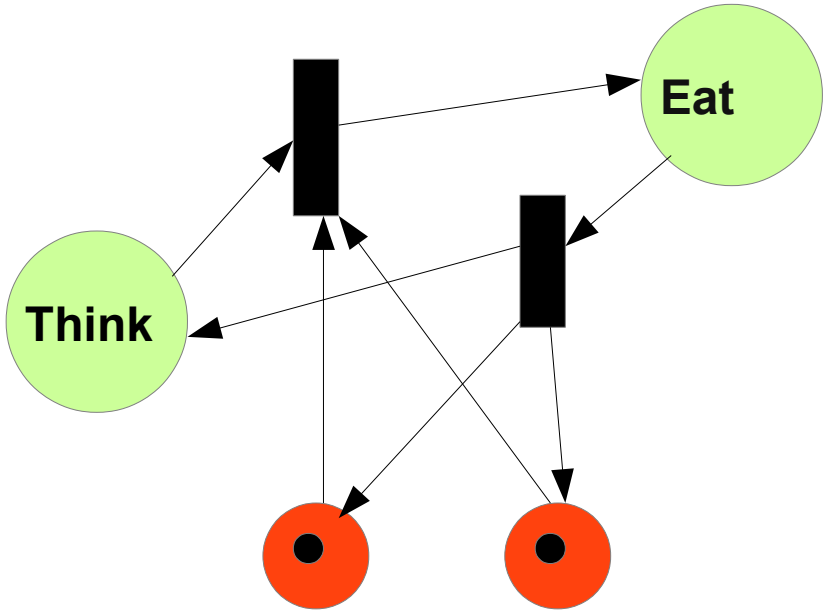
Il problema consiste nello sviluppo di un algoritmo che impedisca lo stallo (deadlock).

Esempio: Dining Philosophers (semplificato)

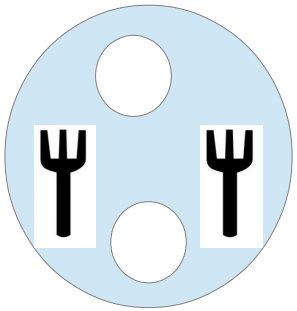


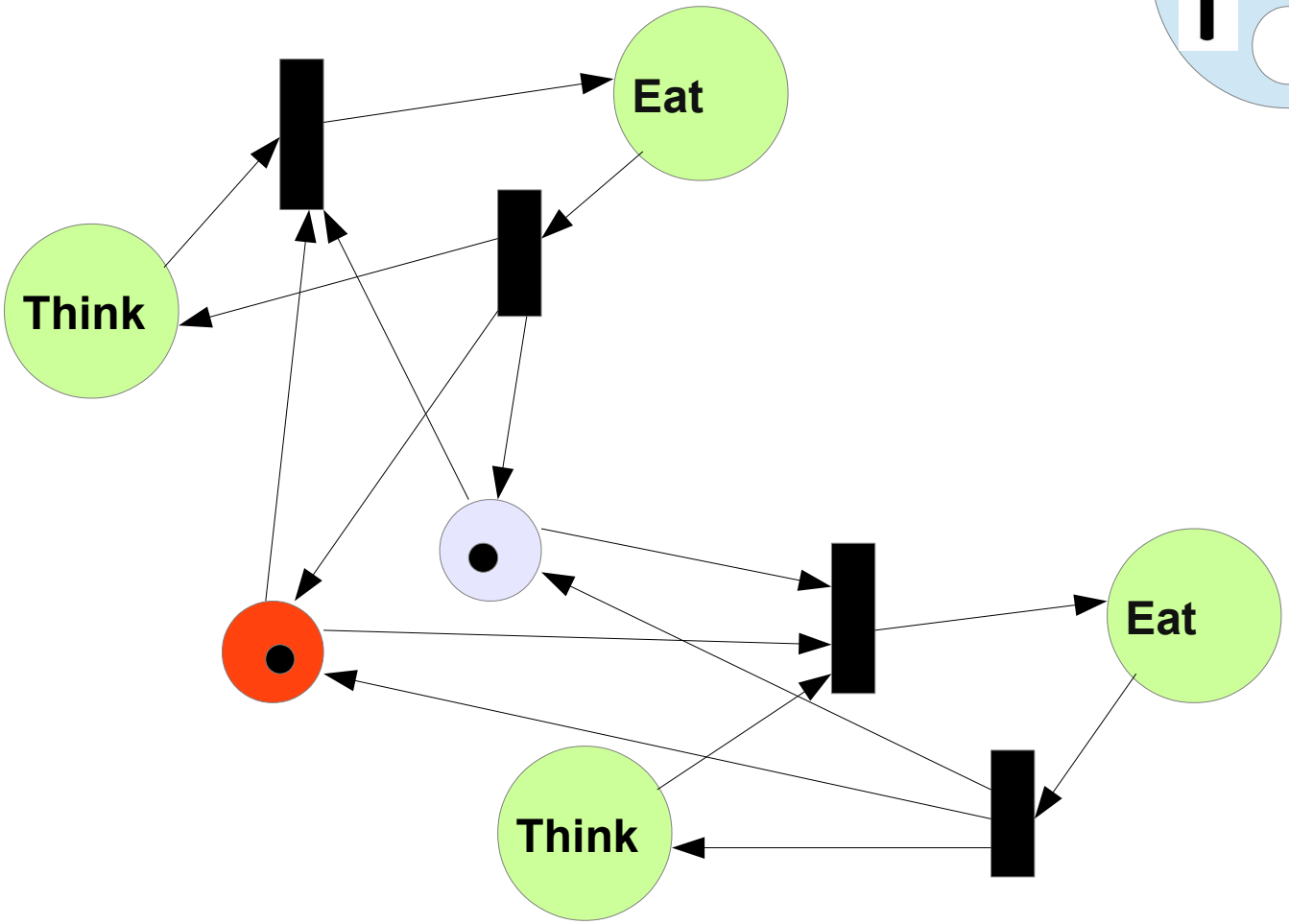
Un filosofo



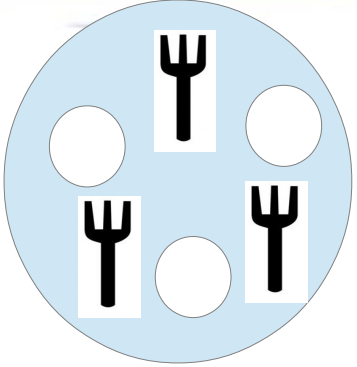


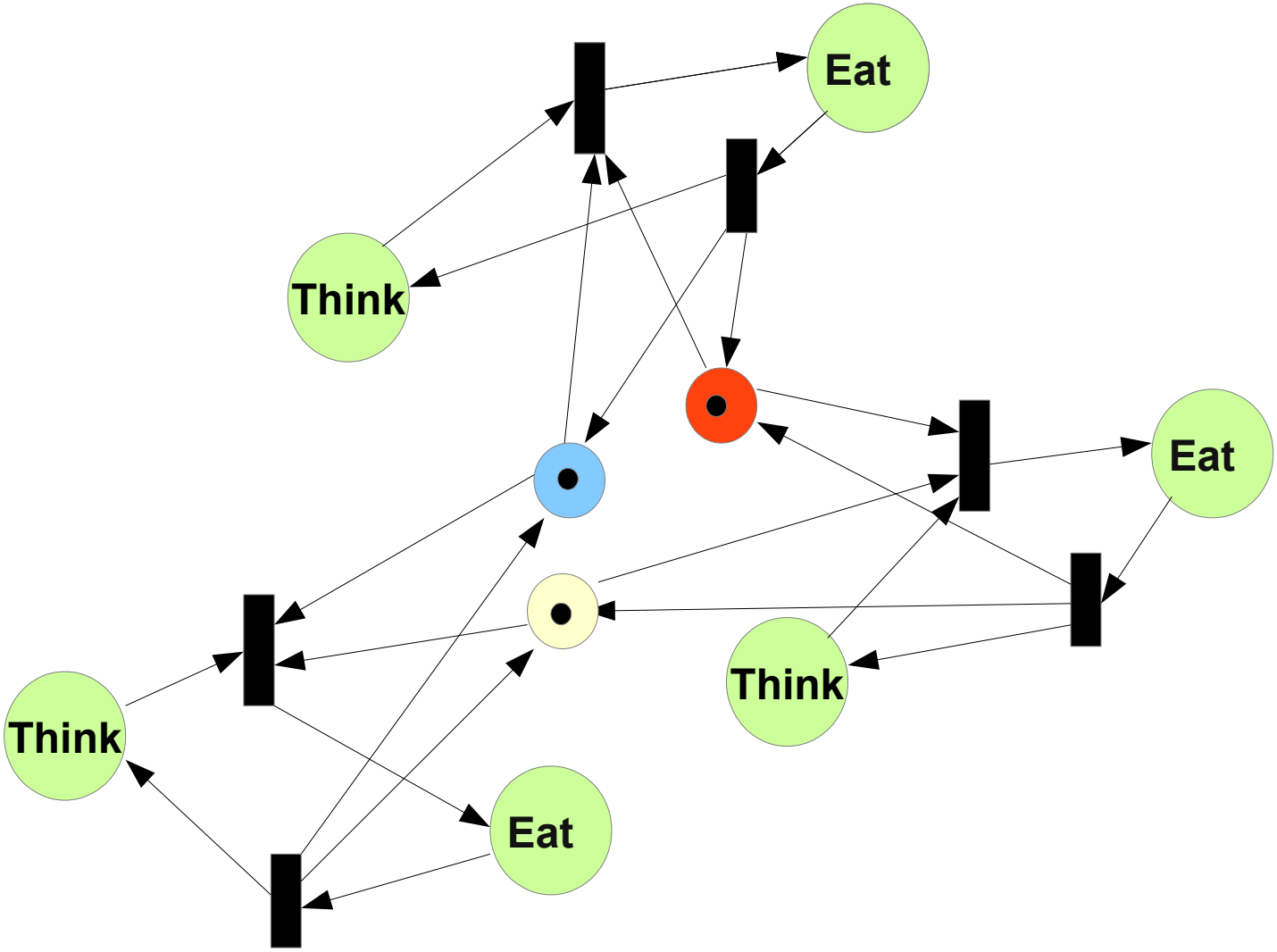
Due filosofi





Tre filosofi





Definizione Formale

Una rete di Petri è una quadrupla (P,T,I,O) tale che:

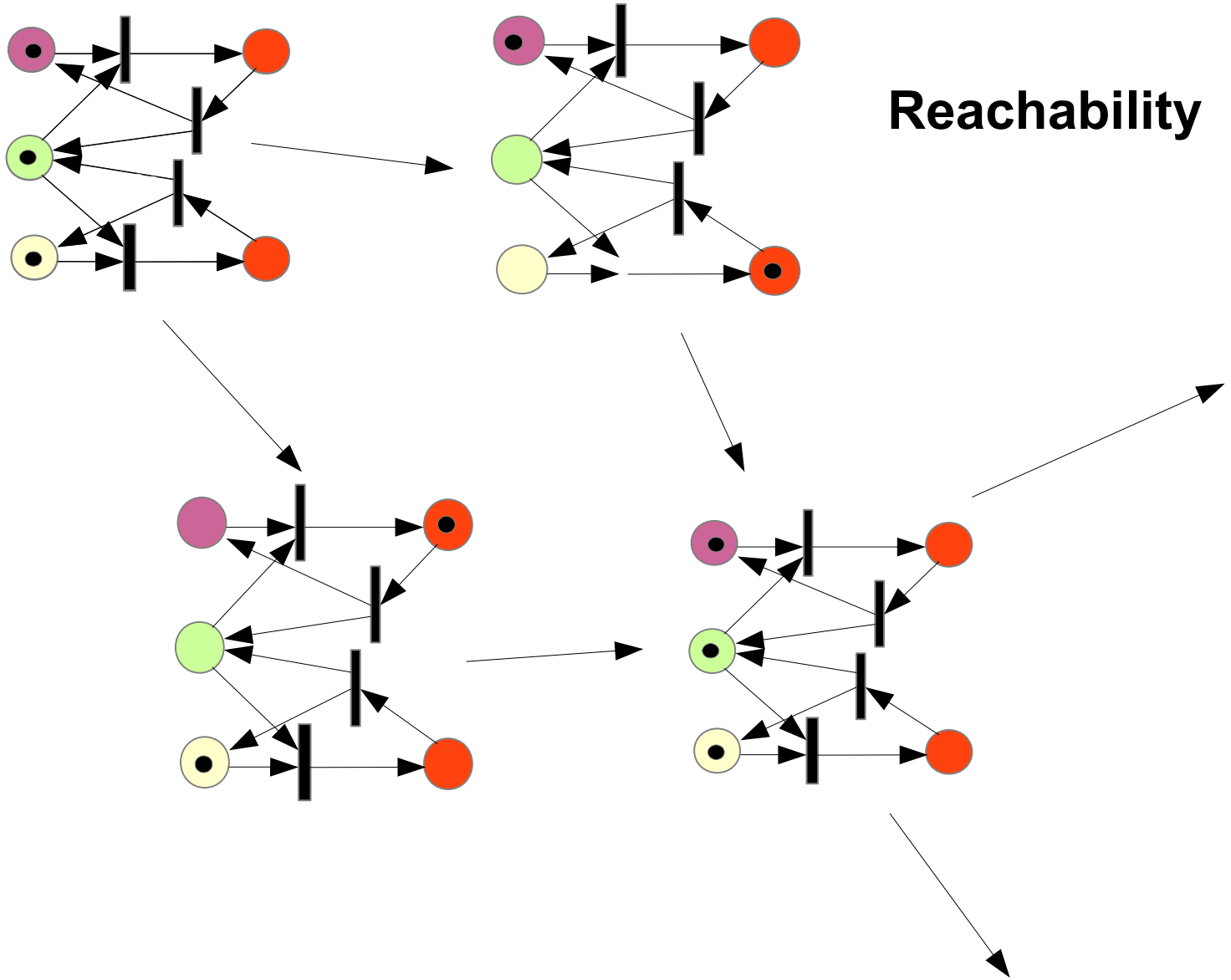
1. P è un'insieme finito di place
   T è un'insieme finito di transizioni
2. $I: P \times T \rightarrow N$ è una funzione che determina la molteplicità di ogni arco entrante in una transizione
3. $O: T \times P \rightarrow N$ è una funzione che determina la molteplicità di ogni arco uscente da una transizione

Analisi di una rete di Petri

Una rete di Petri determina un grafo di stati (**reachability graph**) ottenuto a partire dalla marcatura iniziale

- Se il numero di marcature possibili è finito (la rete è BOUNDED) allora il grafo di raggiungibilità è finito
- Se la rete pùo produrre nuovi token, il grafo potrebbe avere cammini infiniti

Reachability Graph

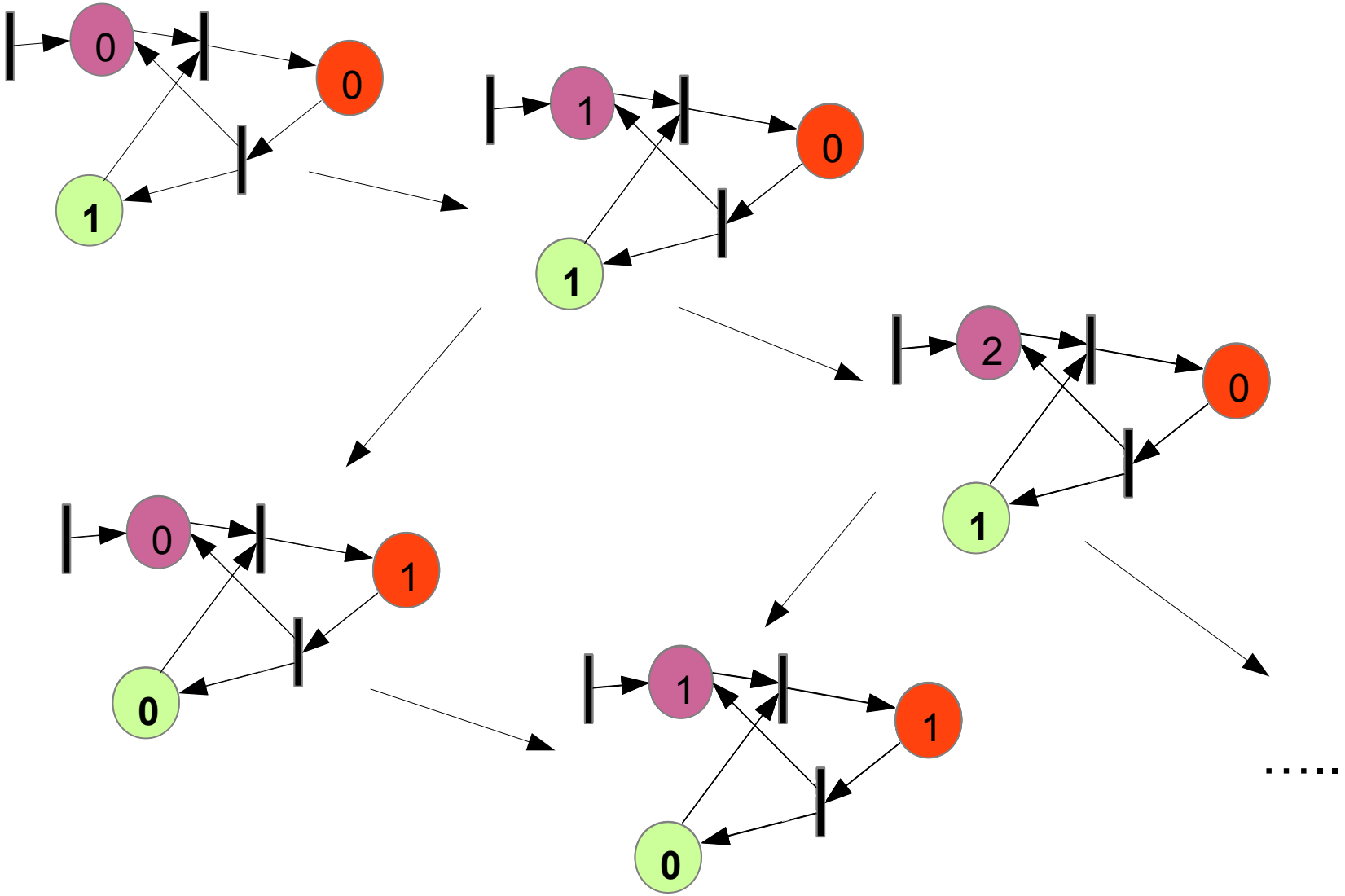


Approssimazioni

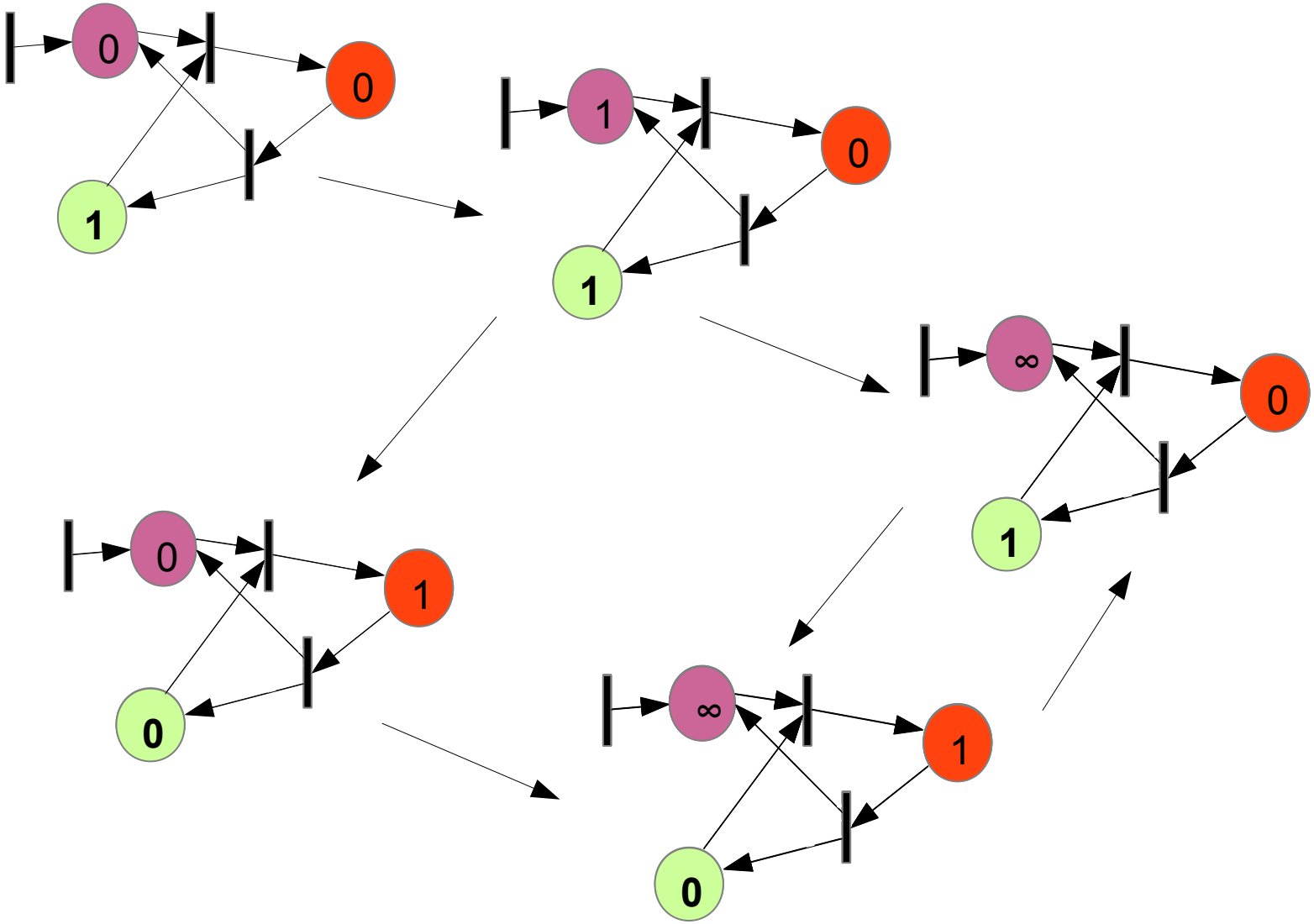
- Se la rete non è BOUNDED possono usare tecniche di approssimazione per gestire cammini infiniti
- Se lungo un cammino una componente cresce rispetto alle marcature precedenti si accelera l'analisi mettendo il valore limite

  ∞ (valore > di qualsiasi naturale)

Esempio: Reachability Tree



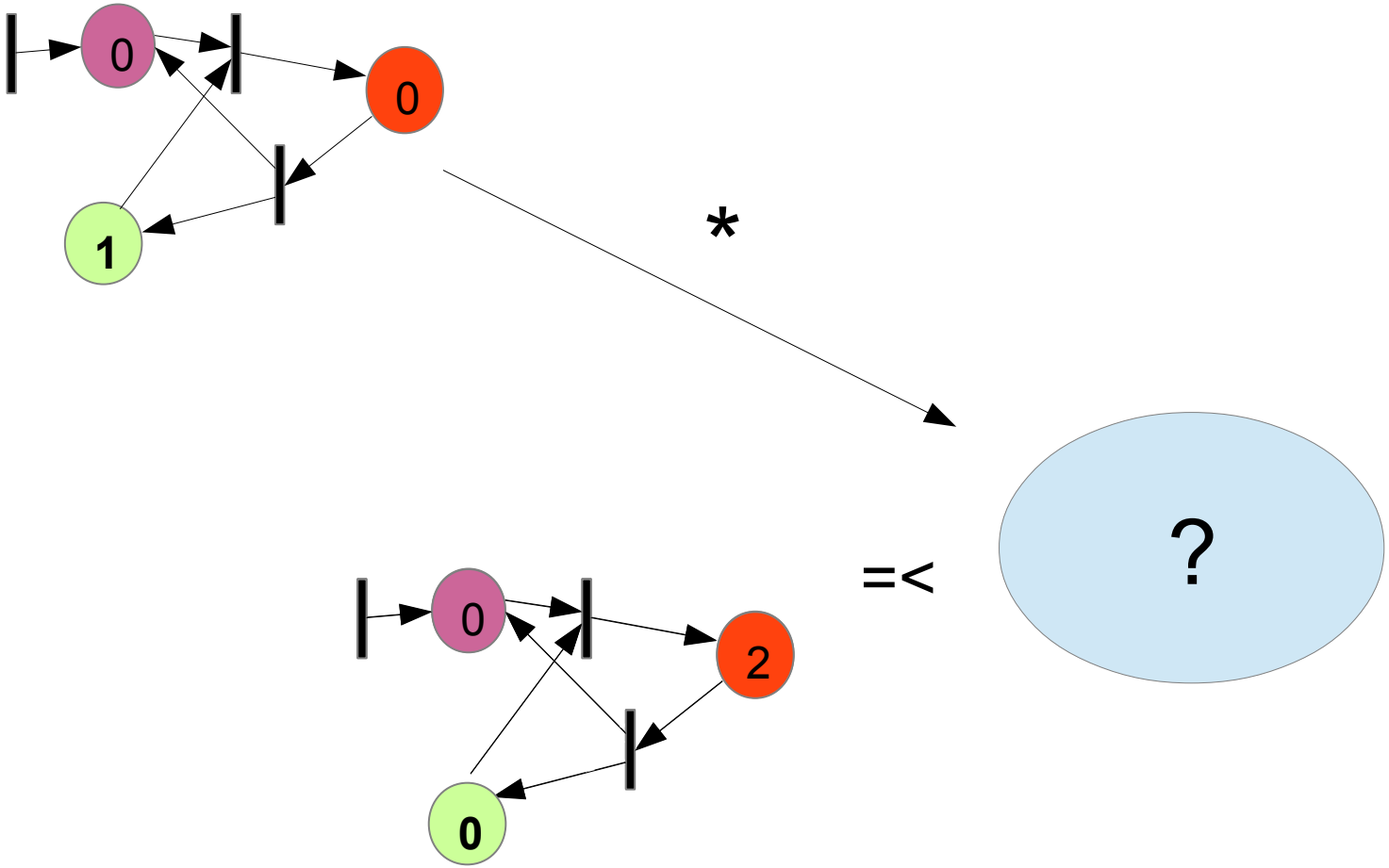
Esempio: Karp-Miller Tree



Coverability Problem

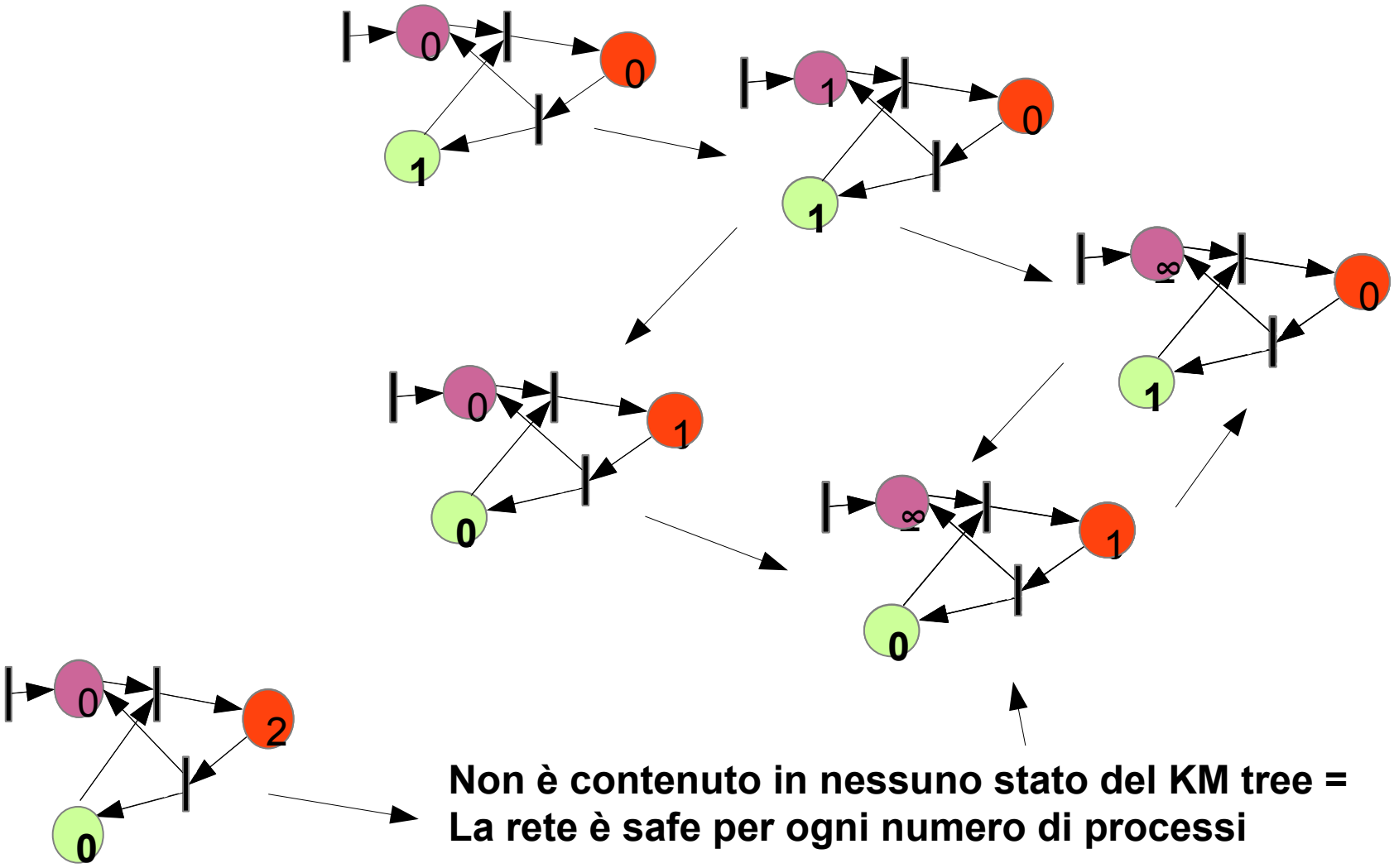
- Date la marcature M (iniziale) ed N (target) vogliamo determinare se esiste una computazione da M ad una marcatura P che in ogni place contiene almeno tanti token quanti N (N=< P rispetto all'ordinamento punto a punto)
- Si può risolvere costruendo il Karp-Miller Tree a partire da M e poi controllando se esistono marcature (eventualmente con ∞) più grandi di N
- L'algoritmo è EXPSPACE!!!

Esempio: Karp-Miller Tree



Pattern che rappresenta violazione mutua esclusione

Esempio: Karp-Miller Tree



Non è contenuto in nessuno stato del KM tree =
La rete è safe per ogni numero di processi

Reachability Problem

Date la marcatura M (iniziale) ed N (target) vogliamo determinare se esiste una computazione da M ad N

Questo problema è in generale meno utile di coverability per analisi di tipo funzionale ma molto più difficile da risolvere

E' comunque ancora decidibile

Nota: reachability è indecidibile per le macchine di Turing (equivalente all'halting problem)

Altre Tecniche di Analisi

Si possono usare tecniche algebriche per analizzare staticamente il reachability graph

Si utilizza la State Equation ottenuta interpretando

- marcature come vettori di naturali dove i-esima componente = numero di token nel place i-esimo
- transizioni come trasformazioni affini tra vettori