

# *Behavioral Design*

---

## *Topics in Behavioral Design*

Based on Material in [Rosenblum94][Budgen94] [Ghezzi91] [Harel88]

# *Behavioral Design Topics*

---

- State Transition Diagrams
- Petri Nets
- Higraphs and Statecharts

# *State Transition Diagrams*

---

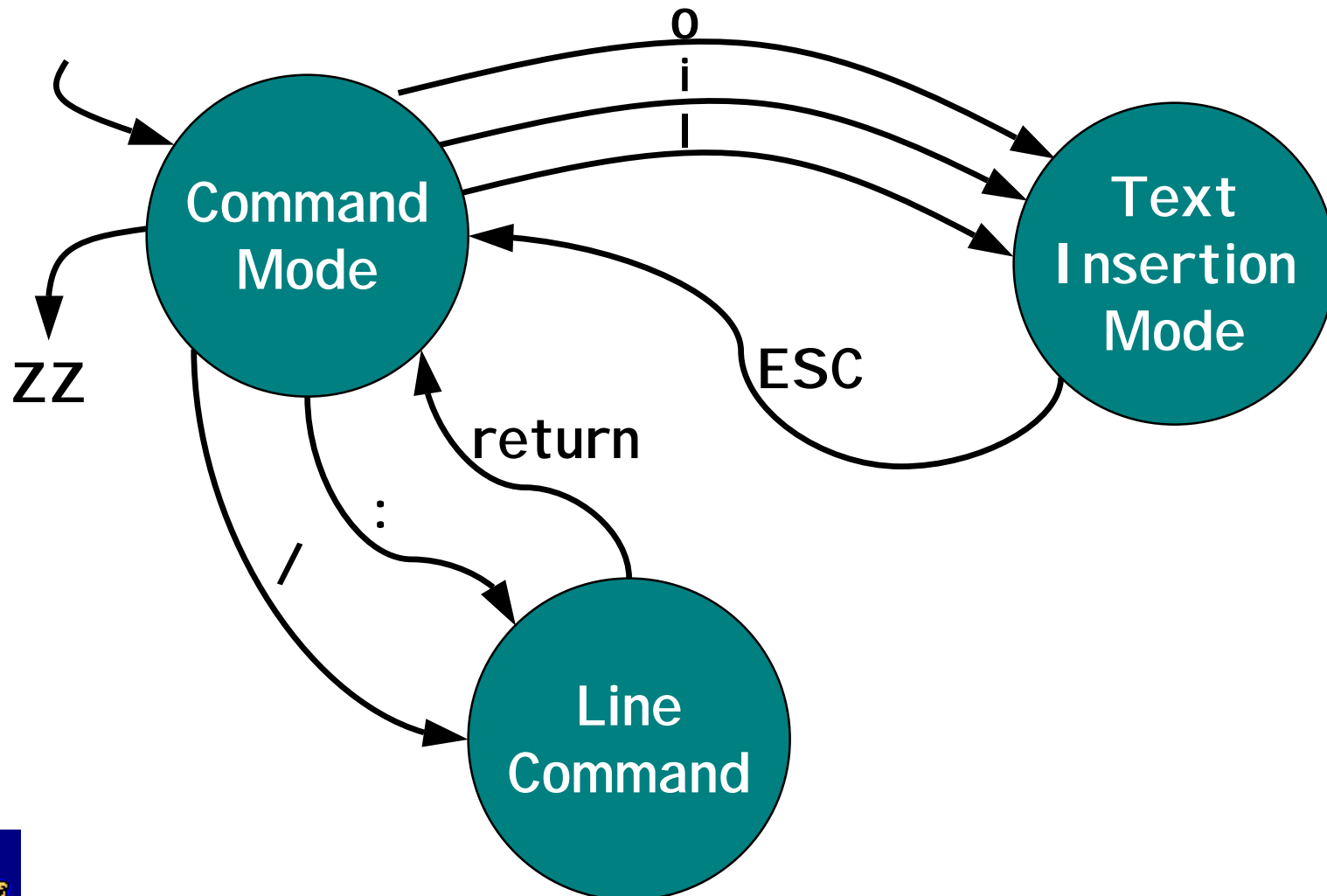
# *State Transition Diagrams (STD)*

---

- Systems exist in a finite set of possible states. External events are triggers that lead to transitions between the states.
- Since most systems have many states, a partial model of the system may be a good compromise.
- STDs are the cornerstone of more powerful diagrams for specifying system behavior, such as Petri Nets and State Charts.

# *Partial Unix vi STD*

---



# *Formal Definition of an STD*

---

$STD = (Q, \Sigma, q_0, \delta, F)$ , where :

$Q$  is a set of states

$\Sigma$  is an input alphabet

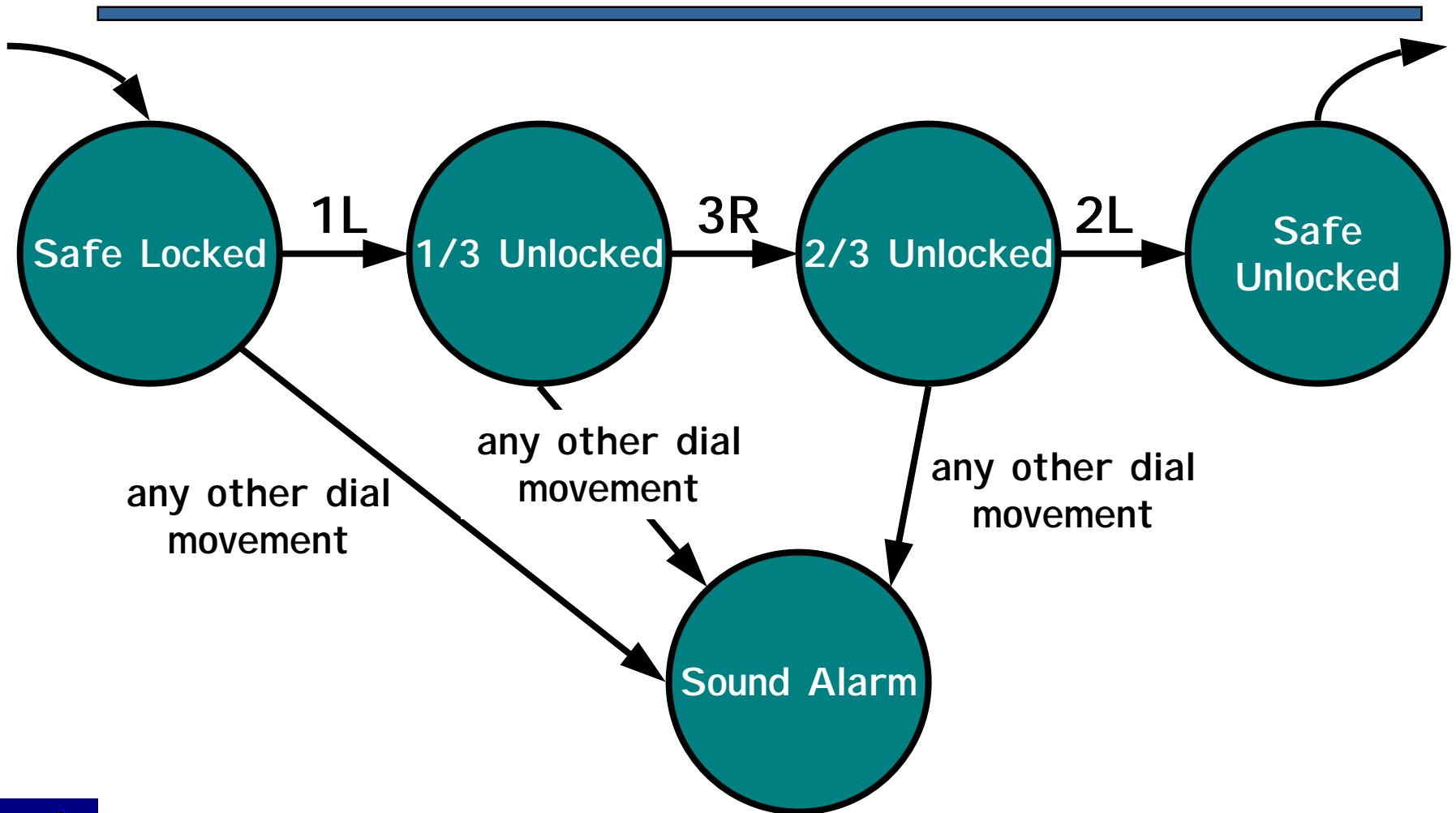
$q_0 \in Q$  is the start state

$\delta$  is a transition function

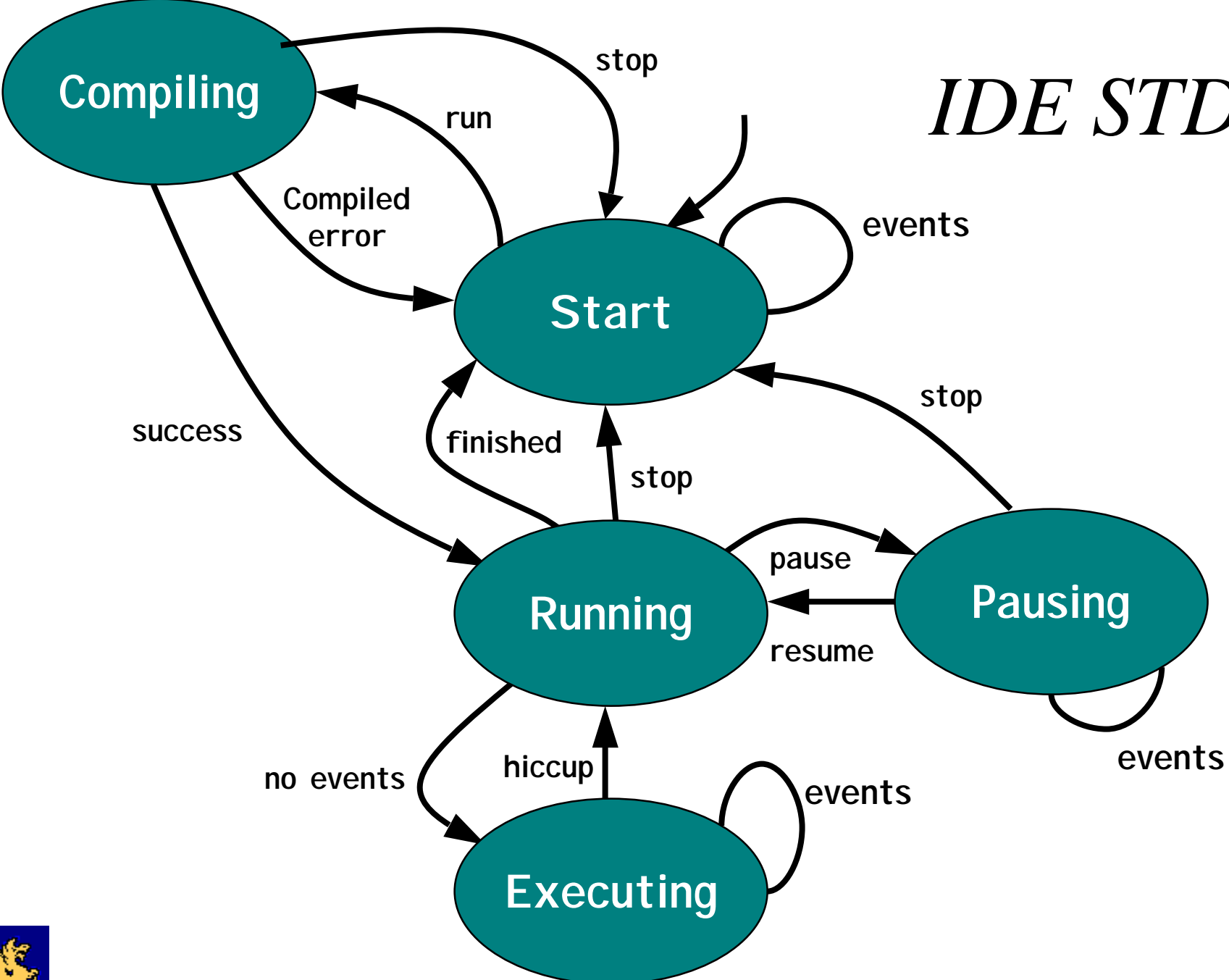
$$\delta : Q \times \Sigma \rightarrow Q$$

$F \subseteq Q$  is the set of final states

# Combination Safe STD



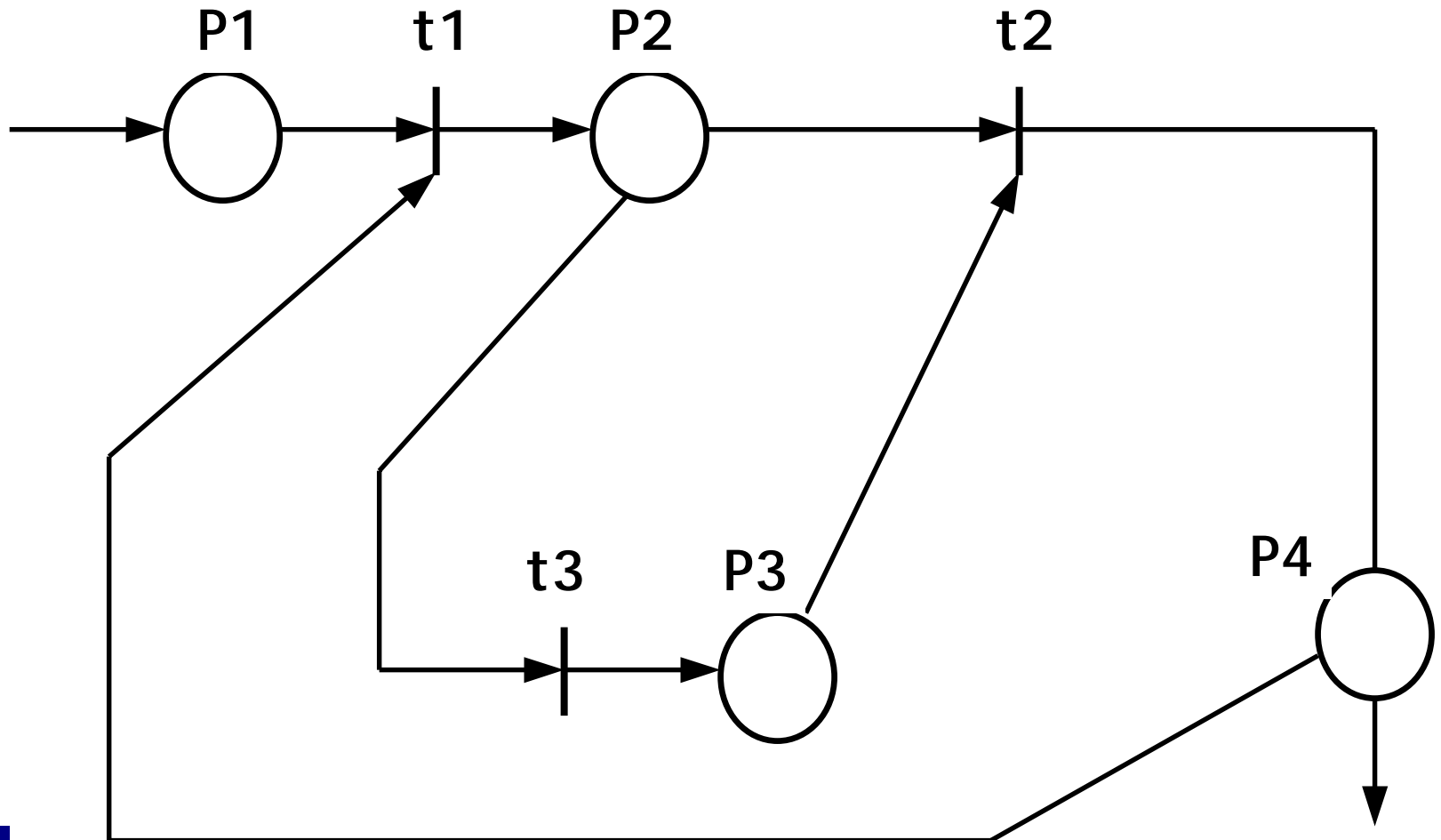
*IDE STD*



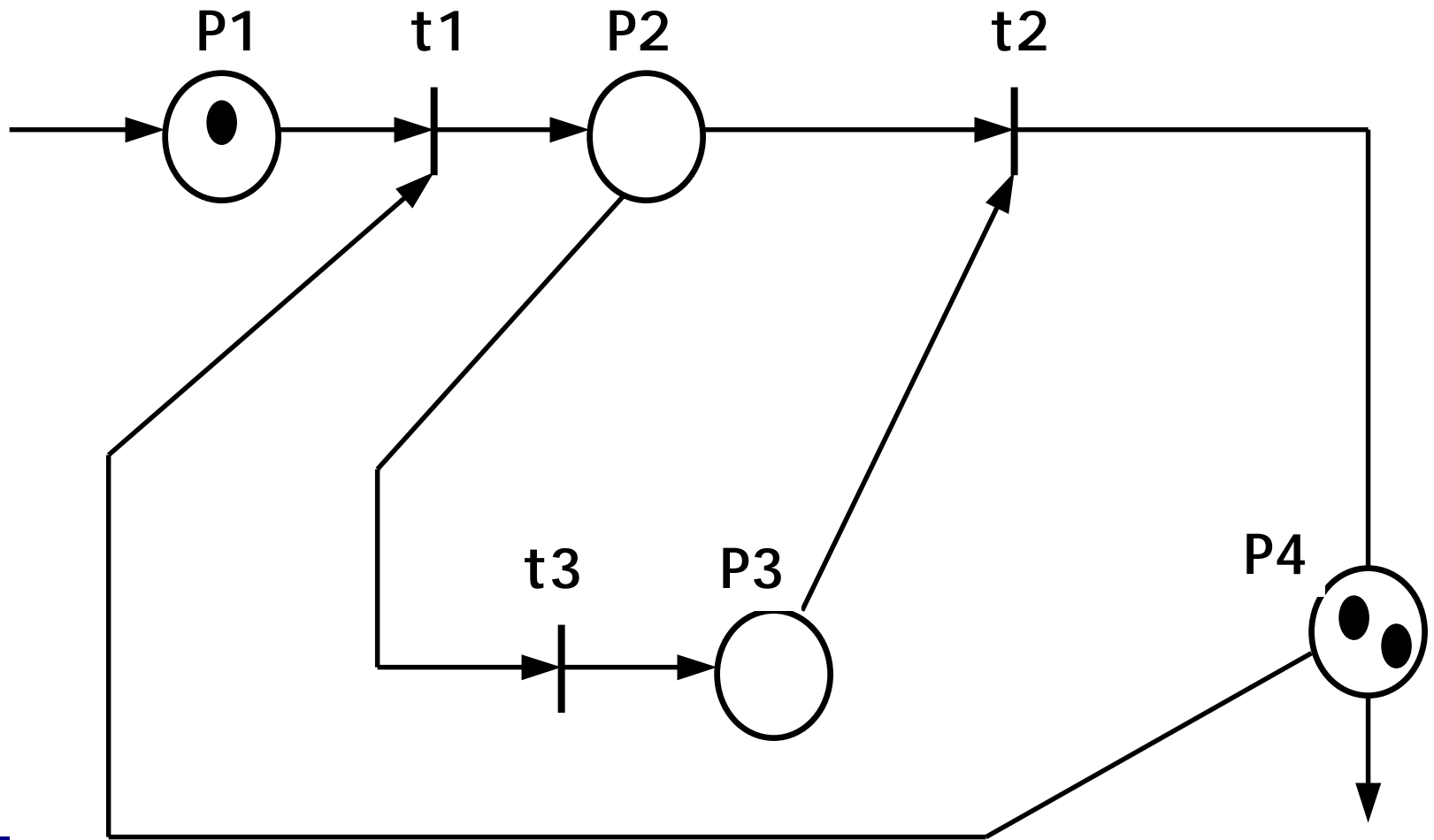
# *Petri Nets*

---

# *A Simple Petri Net*



# *A Marked Petri Net*





# *Definition of a Petri Net*

---

- $PNet = (P, T, A, M0)$ 
  - **P** is a finite set of *places* (labeled circles), where a place holds tokens.
  - **T** is a finite set of *transitions* (bars), where a transition represents an activity.
  - **A** is a finite set of directed *arcs*, where an arc connects a place and a transition.
  - **M0** is the initial *marking* of  $PNet$ , where a marking is an arrangement of tokens in places representing state.

# *Petri Net Execution Model*

---

- **Input Place:** Place  $P$  is an *input place* for transition  $T$  if there is an arc from  $P$  to  $T$ .
- **Output Place:** Place  $P$  is an *output place* for transition  $T$  if there is an arc from  $T$  to  $P$ .
- **Enabled Transition:** A transition is *enabled* if there is at least one token at each of its input places.

# Petri Net

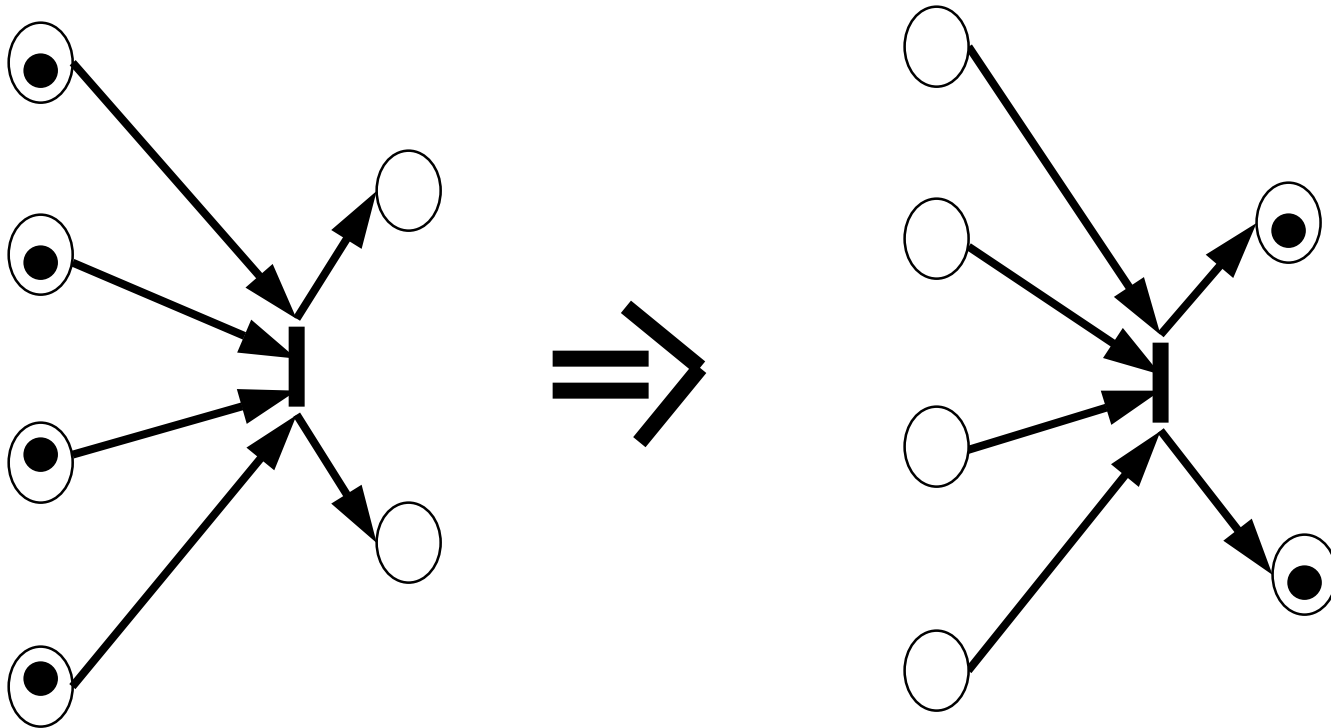
## Execution Model (Cont'd)

---

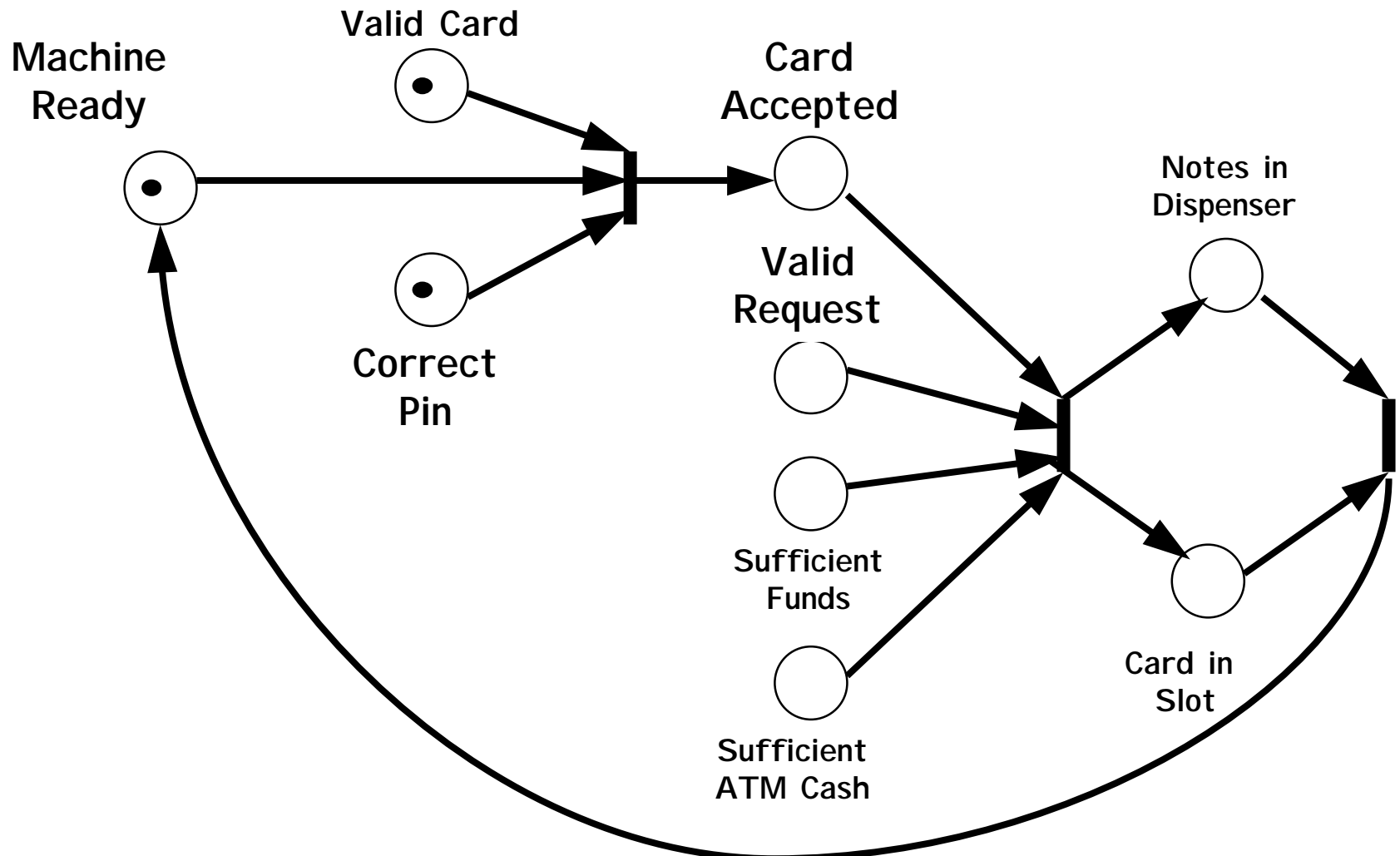
- **Firing a Transition:** An enabled transition is non deterministically selected and *fired* by removing one token from each of its input places and depositing one token at each of its output places.
- **Firing Sequence:** A firing sequence  $\langle t_0, t_1, \dots, t_n \rangle$  such that  $t_0$  is enabled and fired in  $M_0$ ,  $t_1$  is enabled and fired in  $M_1$ , ...

# *Petri Net Firing*

---

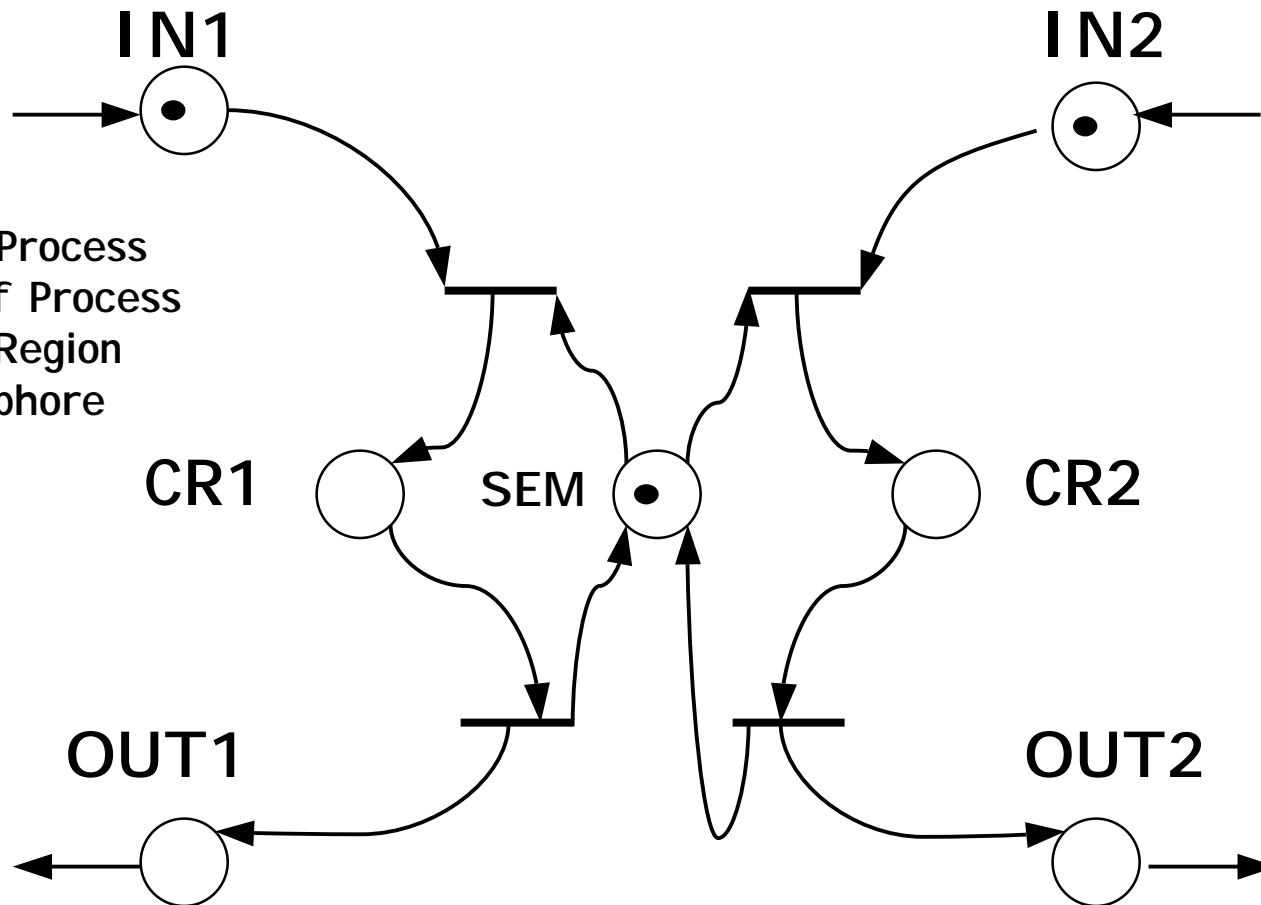


# A Petri Net Describing an ATM



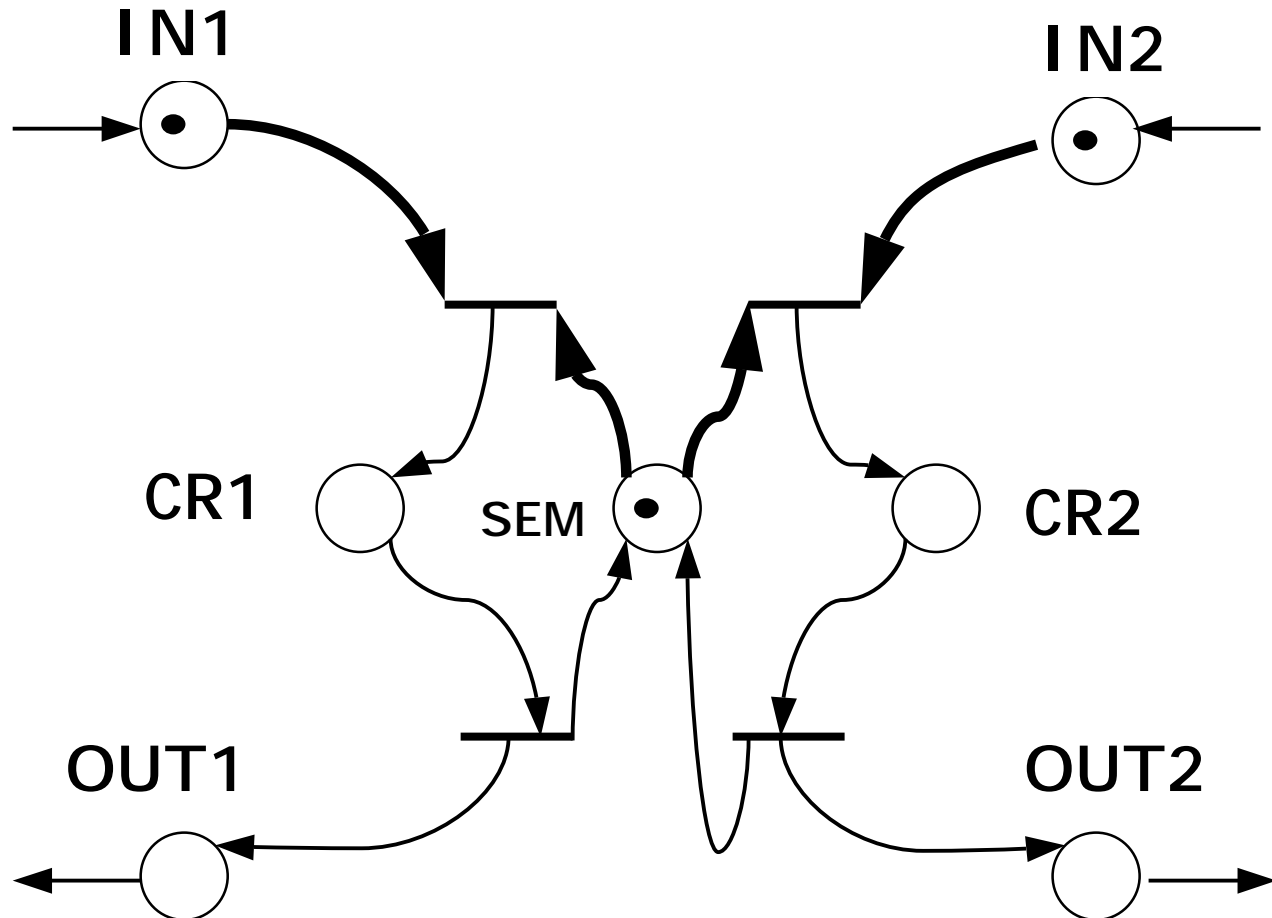
# *A Marked Petri Net Semaphore*

---



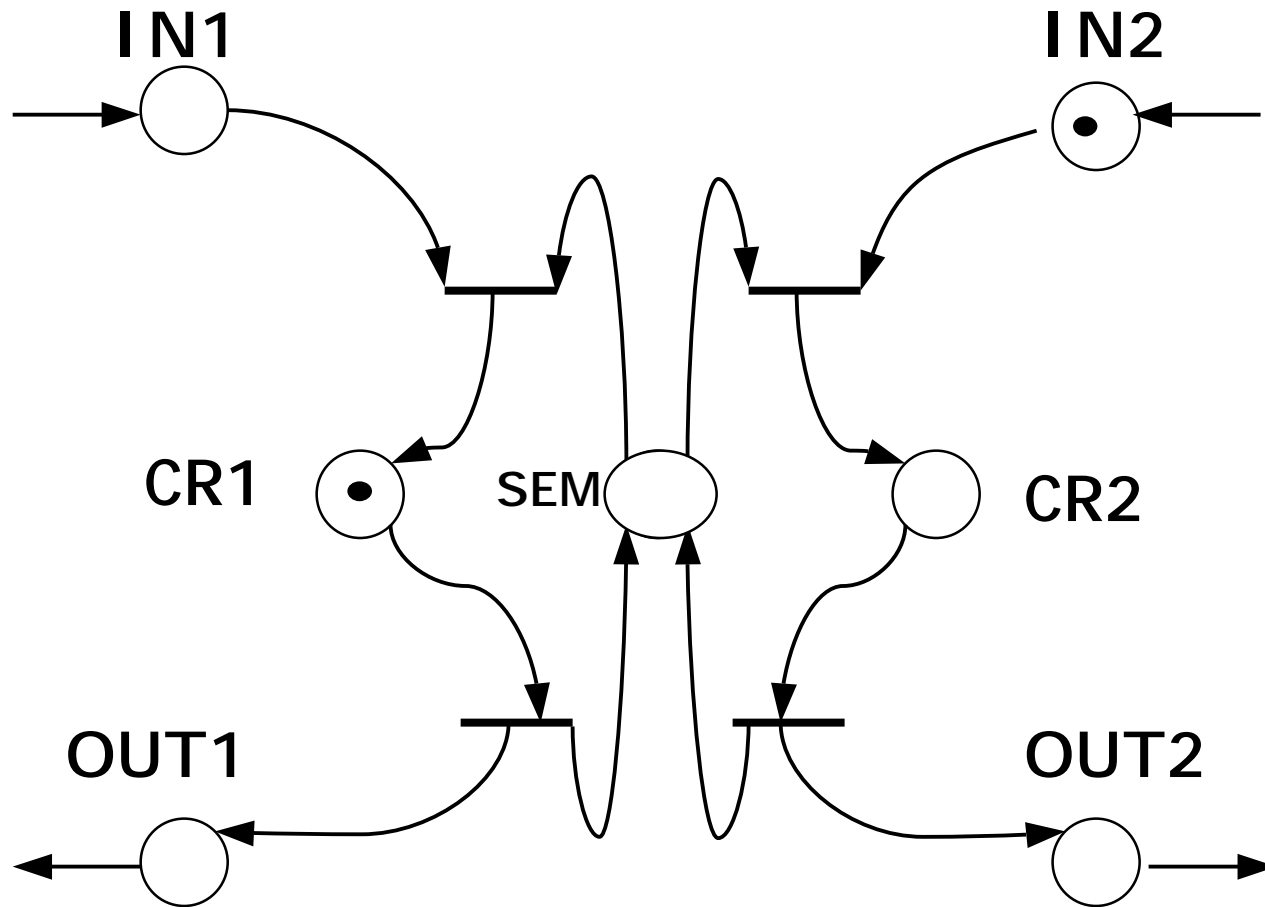
# *Enabled Transitions*

---



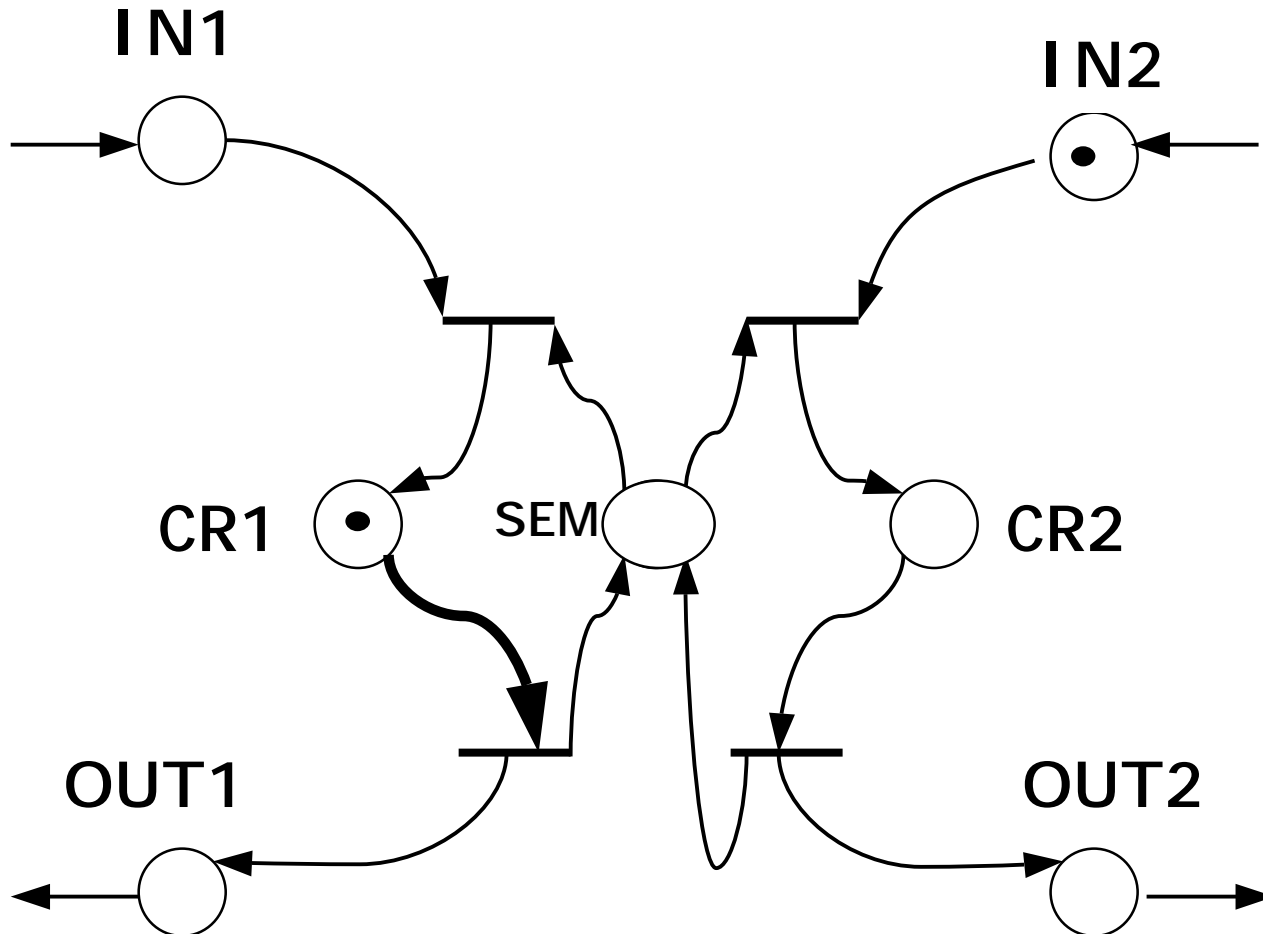
# *After Non-Deterministic Firing*

---



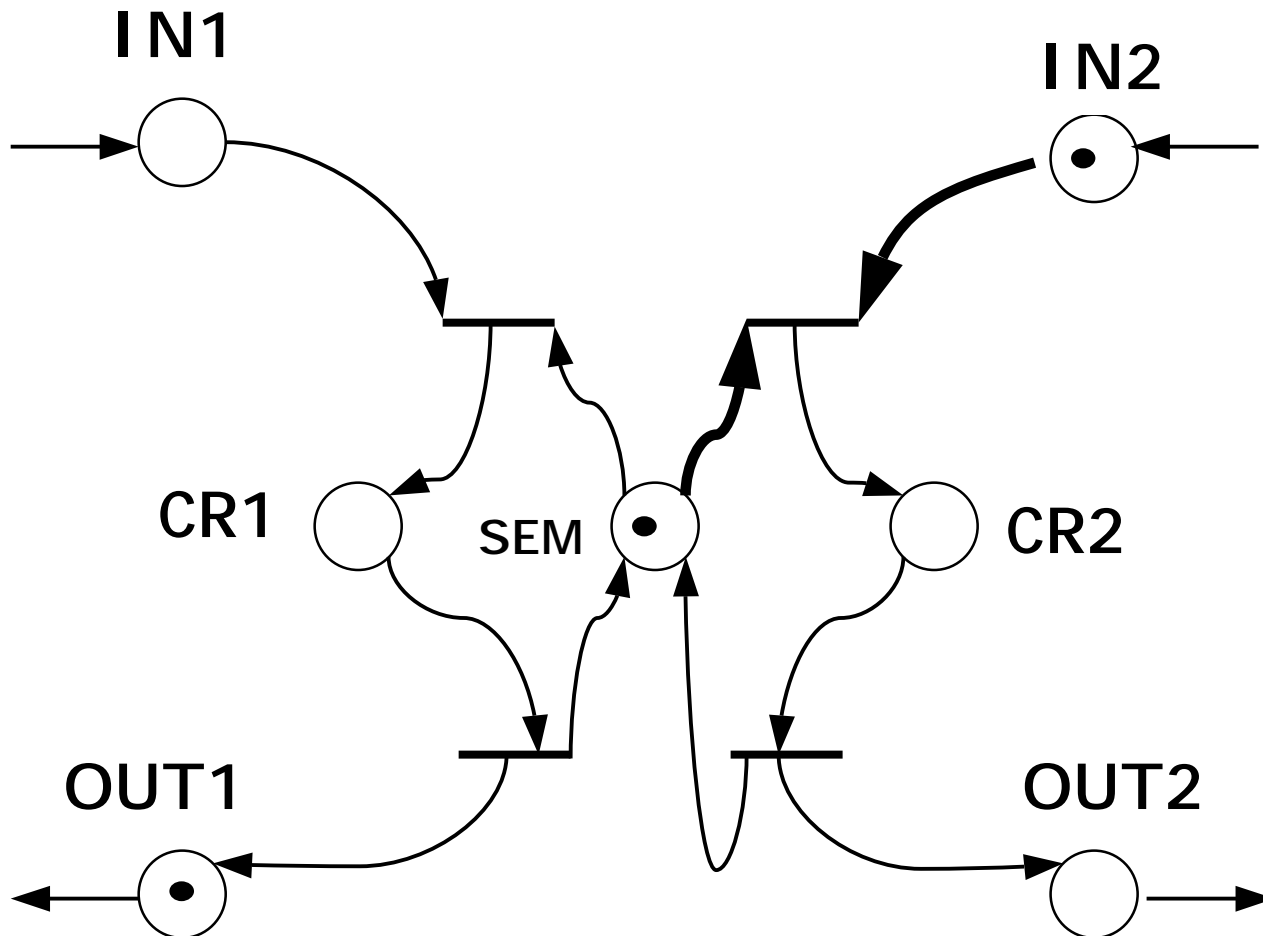
# *Enabled Transition*

---



# *After Firing*

---



# *Petri Net Static Analysis*

---

- **Invariants** are properties of a Petri net that hold (are true) in all markings.
- For example, the sum of all tokens in *CR1*, *CR2*, and *SEM* are equal to 1 in all reachable markings. That is,  
 $|CR1| + |CR2| + |SEM| = 1$

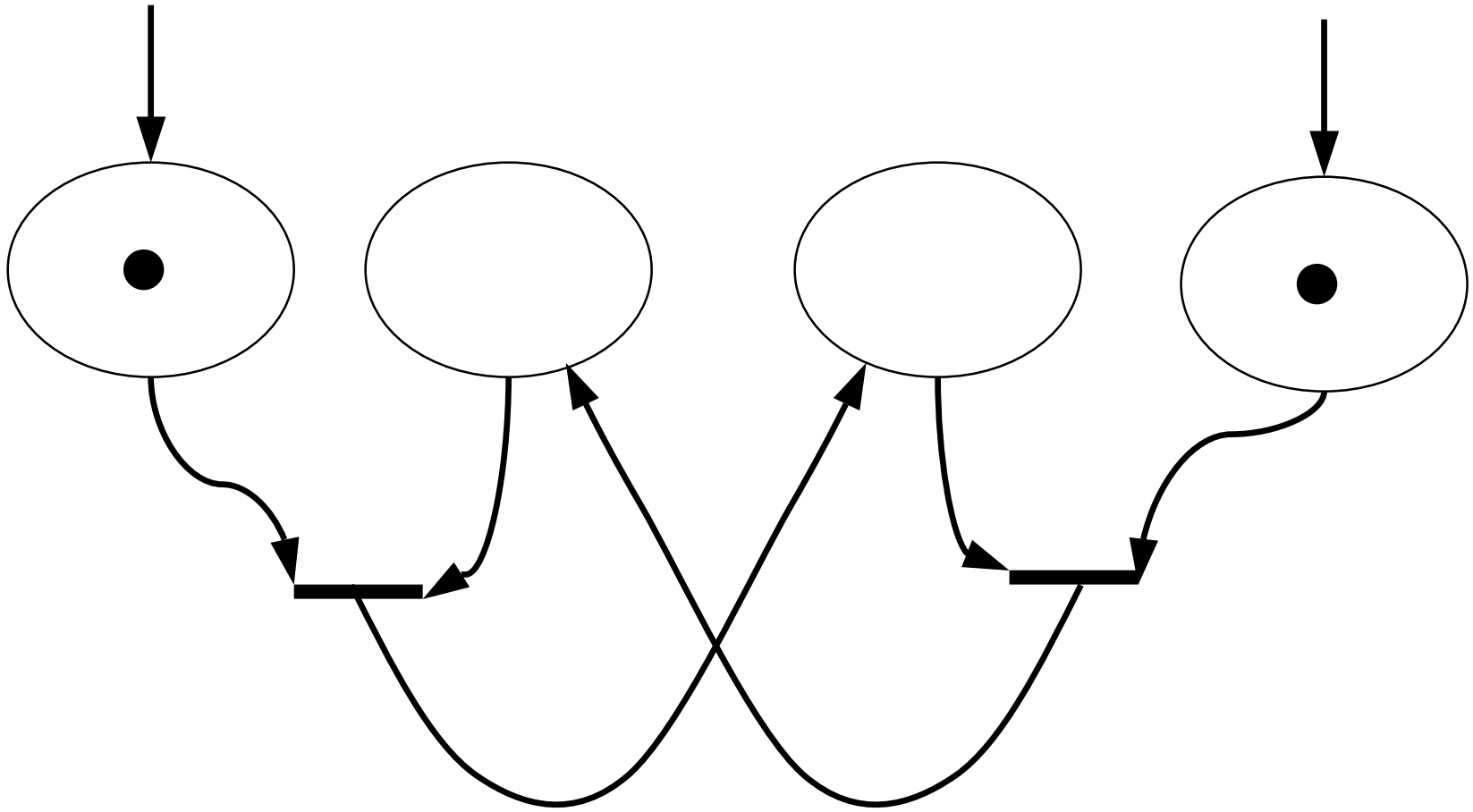
# *Deadlock and Starvation*

---

- A Petri Net with a given marking is in **deadlock** iff no transition is enabled in that marking.
- A Petri Net with a given marking is in **starvation** iff one or more transitions have been permanently disabled.
- A Petri Net is **live** if every transition can eventually be fired.

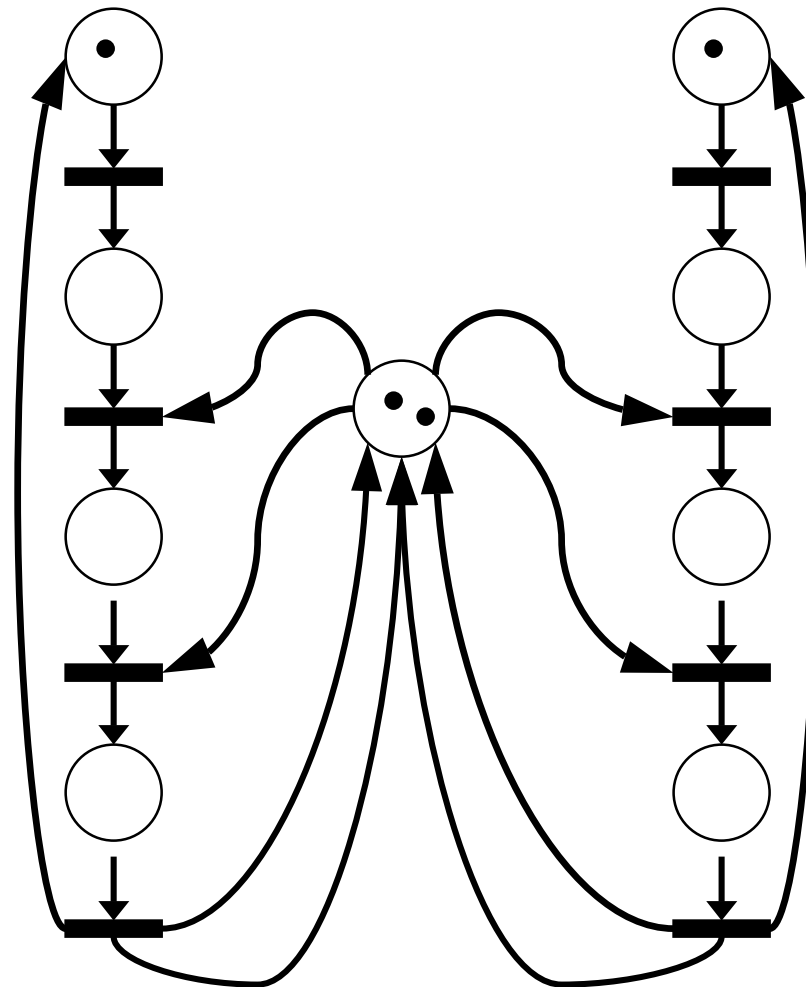
# *A Deadlocked Petri Net*

---



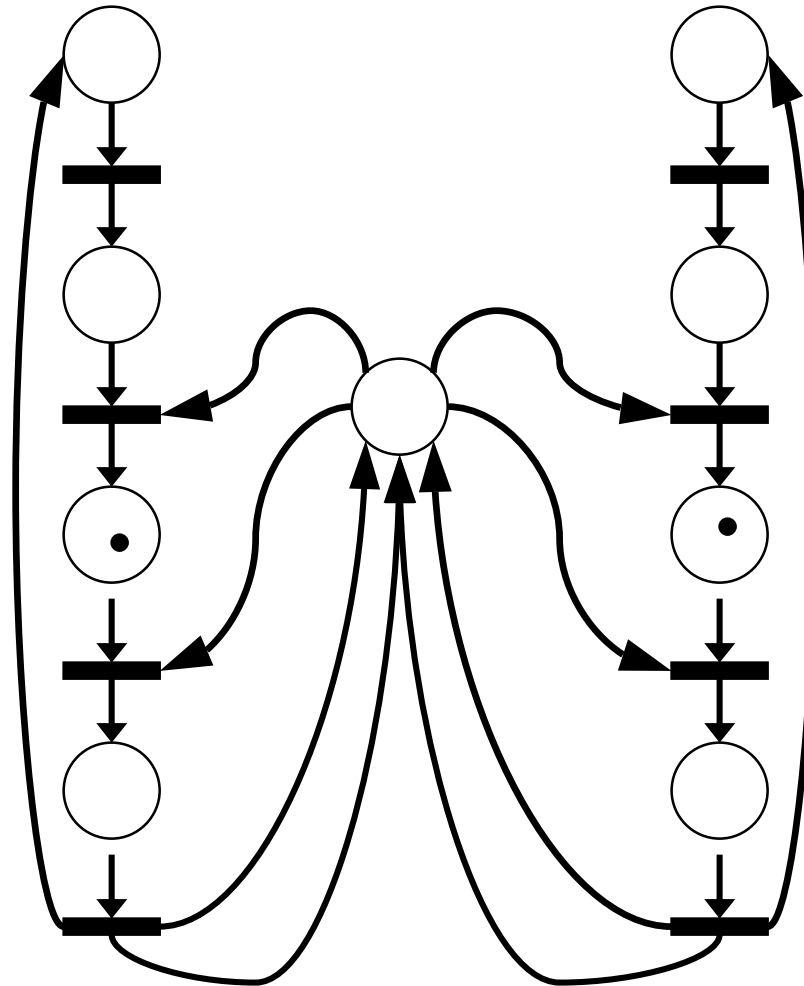
# *A Petri Net that can Enter a Deadlocked State*

---



# *A Deadlocked Petri Net*

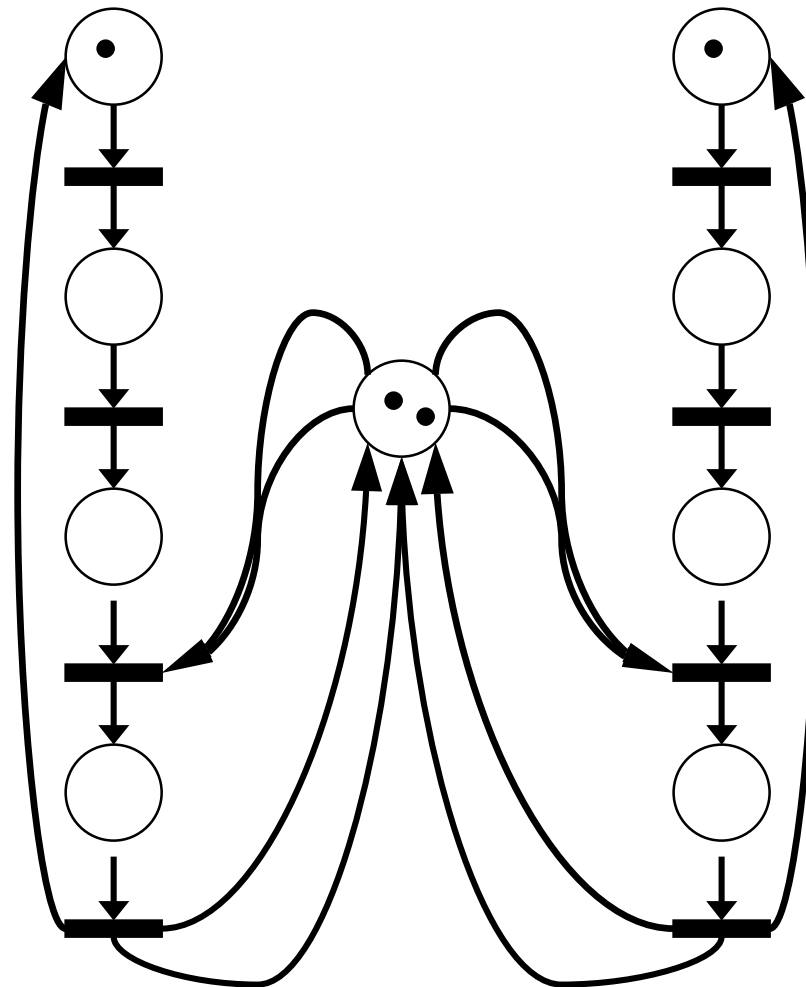
---



Software Design (Behavioral)

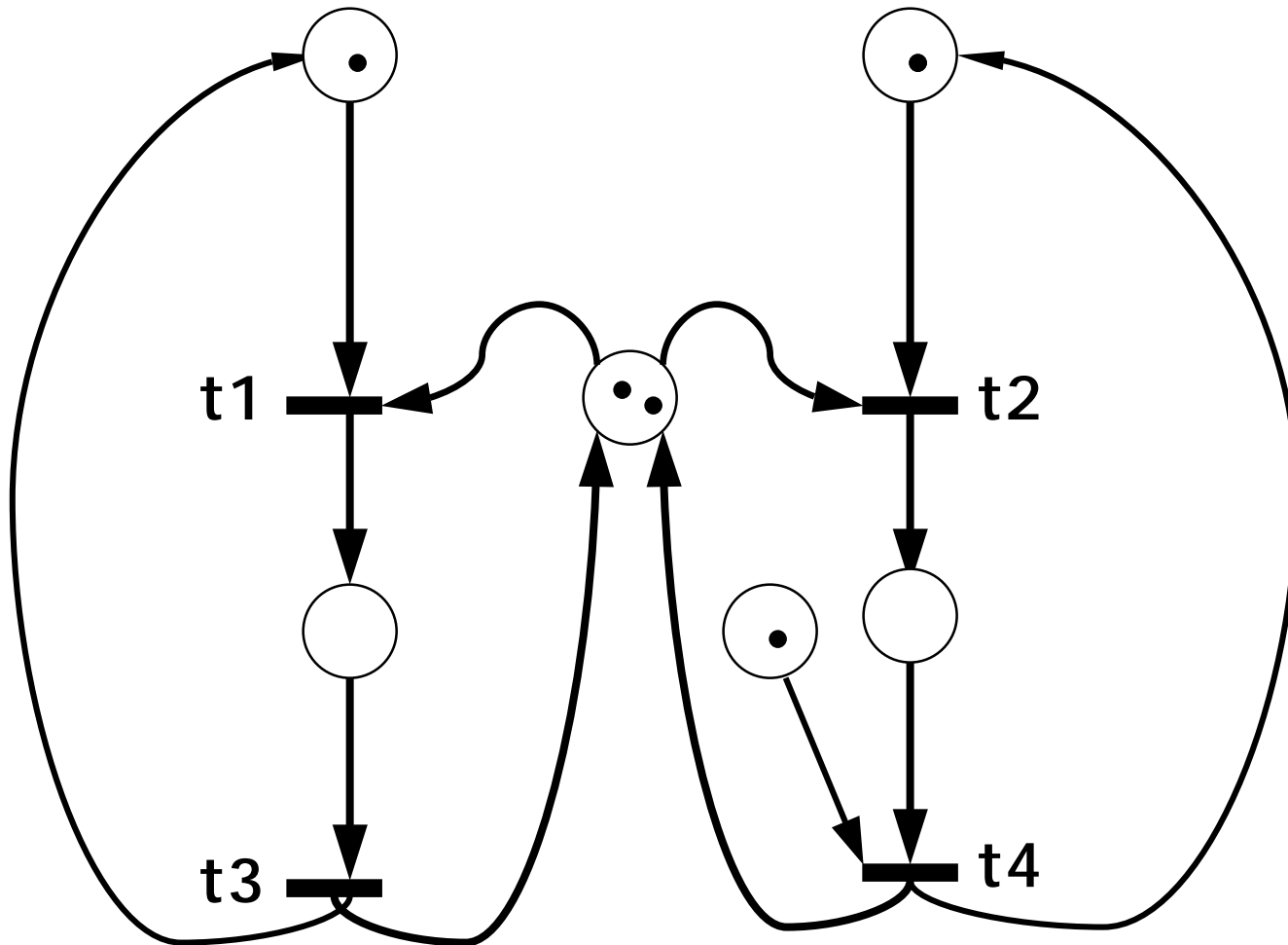
# *Modification into a Live Petri Net*

---



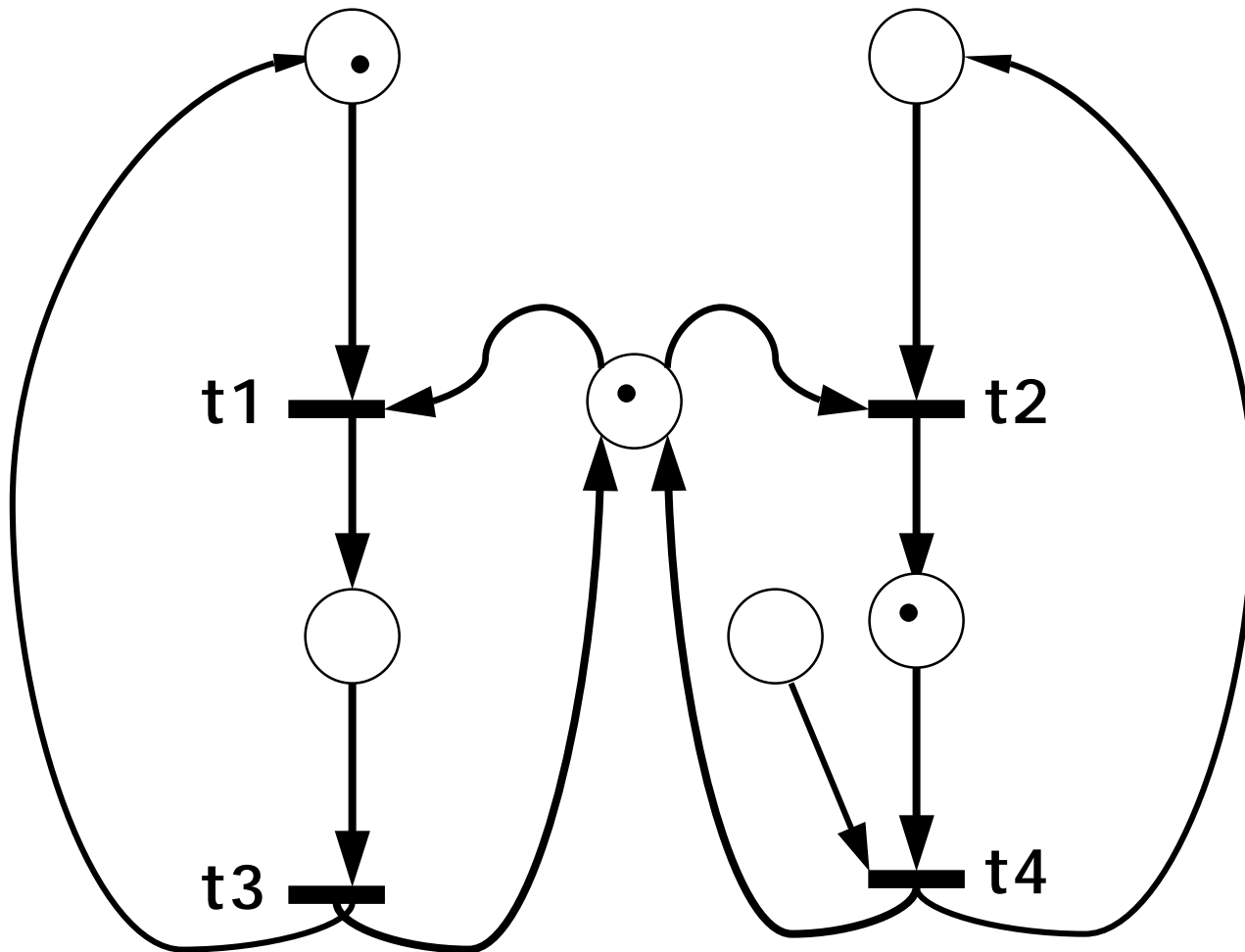
# *A Petri Net that can go into Starvation*

---



# *Starving Transitions t2 and t4*

---

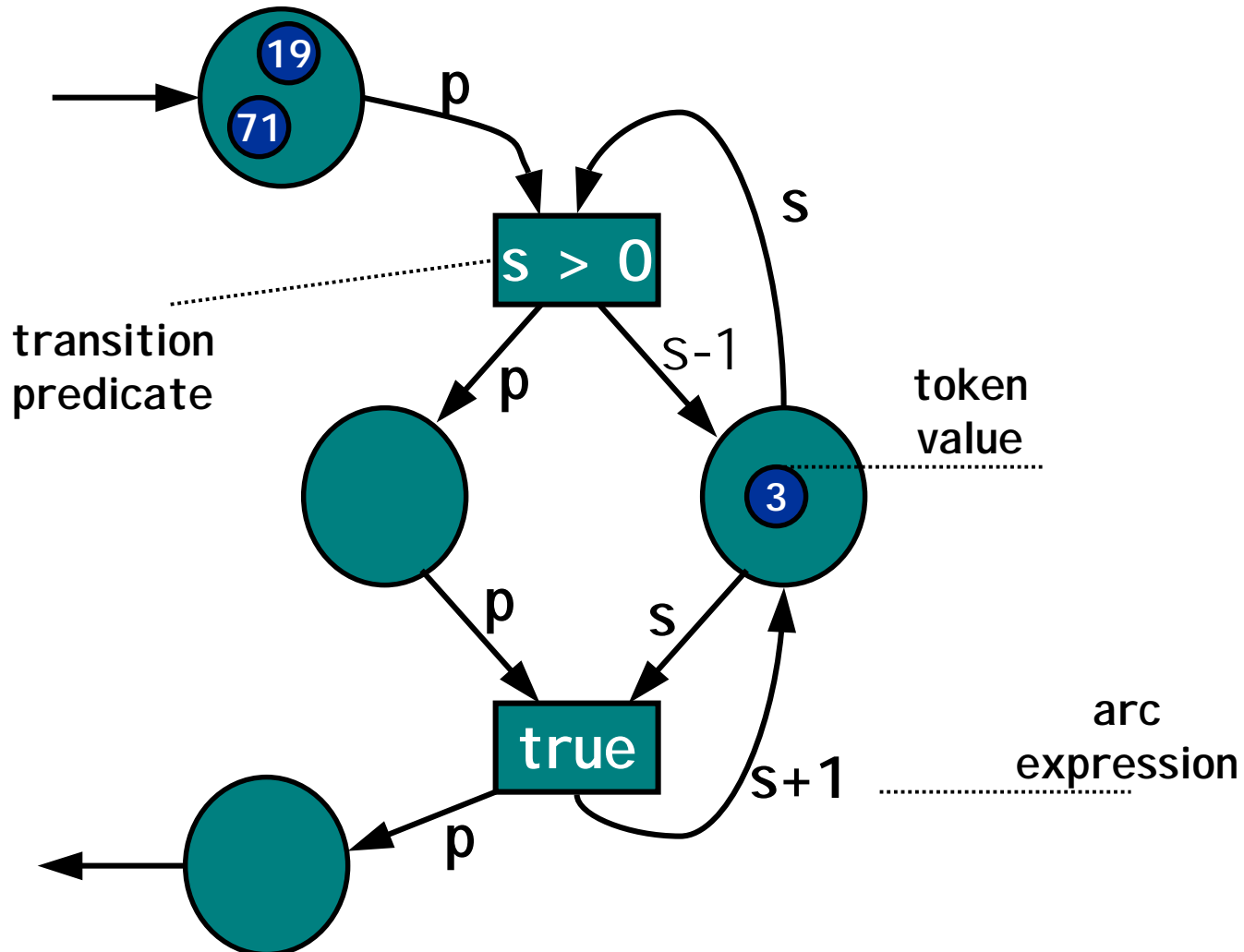


# *Shortcoming of Basic Petri Nets*

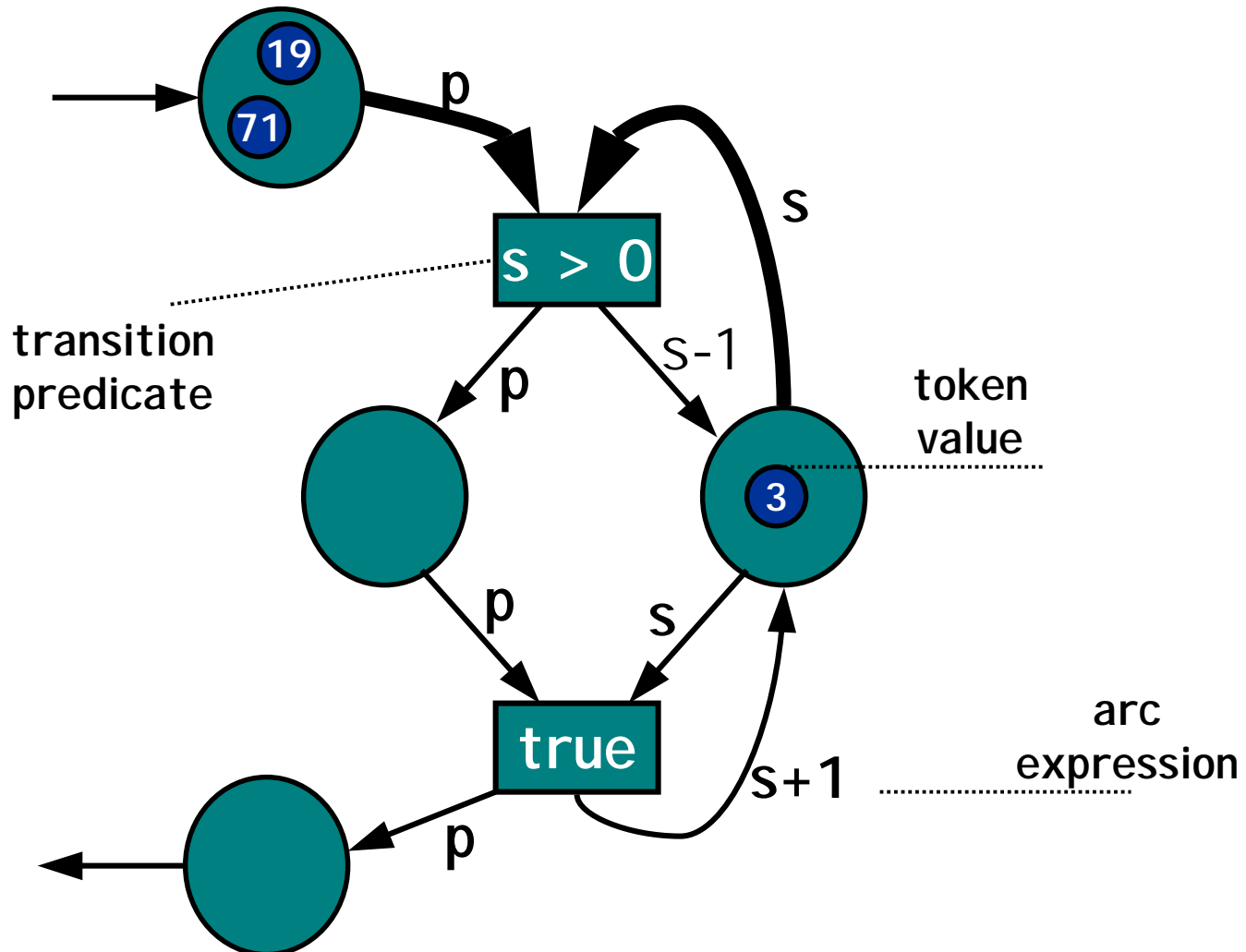
---

- **The Simplicity of building blocks leads to complexity in nets.**
- For example, a semaphore of  $N$  processes requires  $2N$  transitions and  $3N+1$  places.
- Would like:
  - *Enable* and *fire* as computations.
  - *Tokens* as data, not just control.
  - Ability to reduce high-level Petri nets to basic Petri nets for analysis.

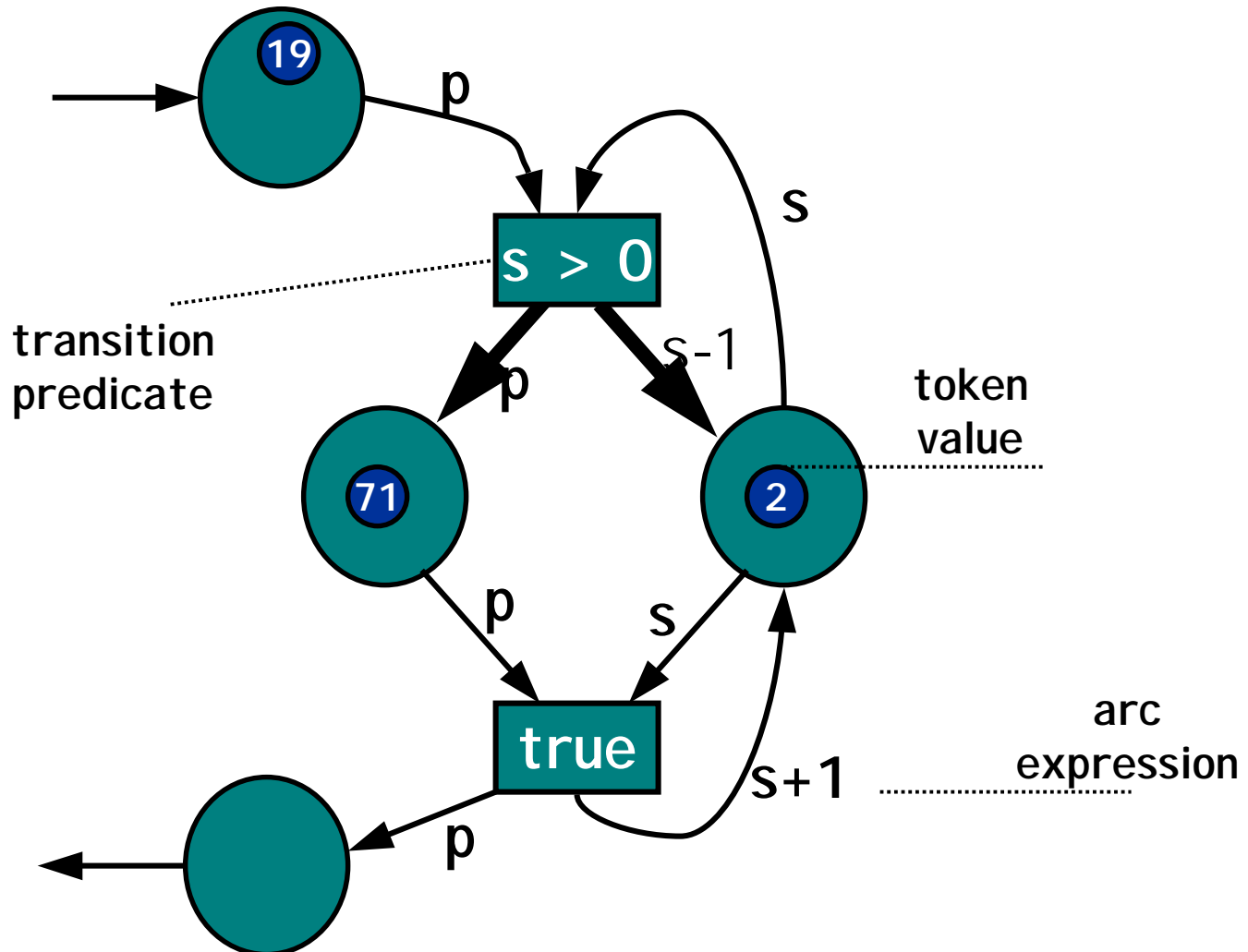
# Higher-Level Net Semaphore



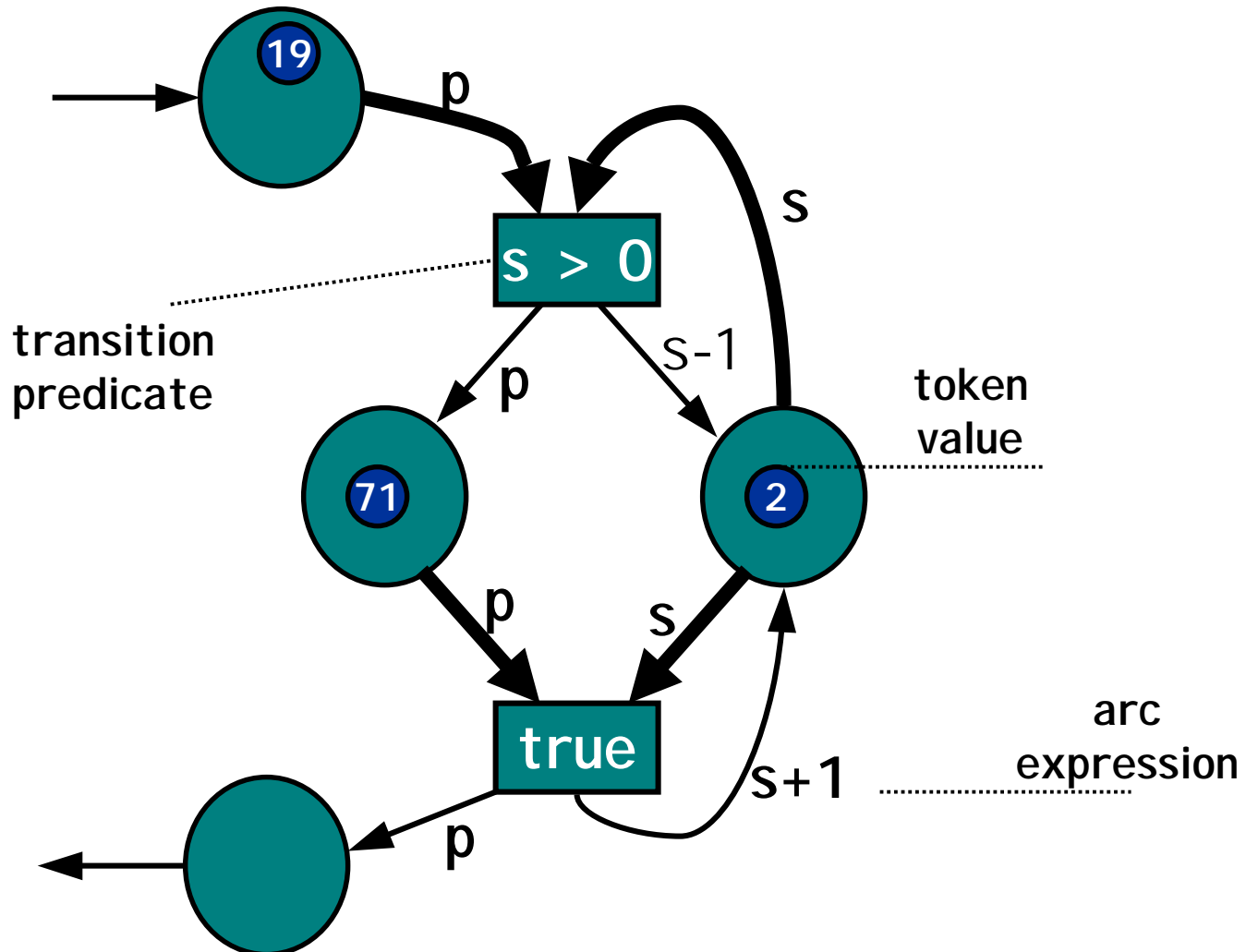
# Enabled Transition



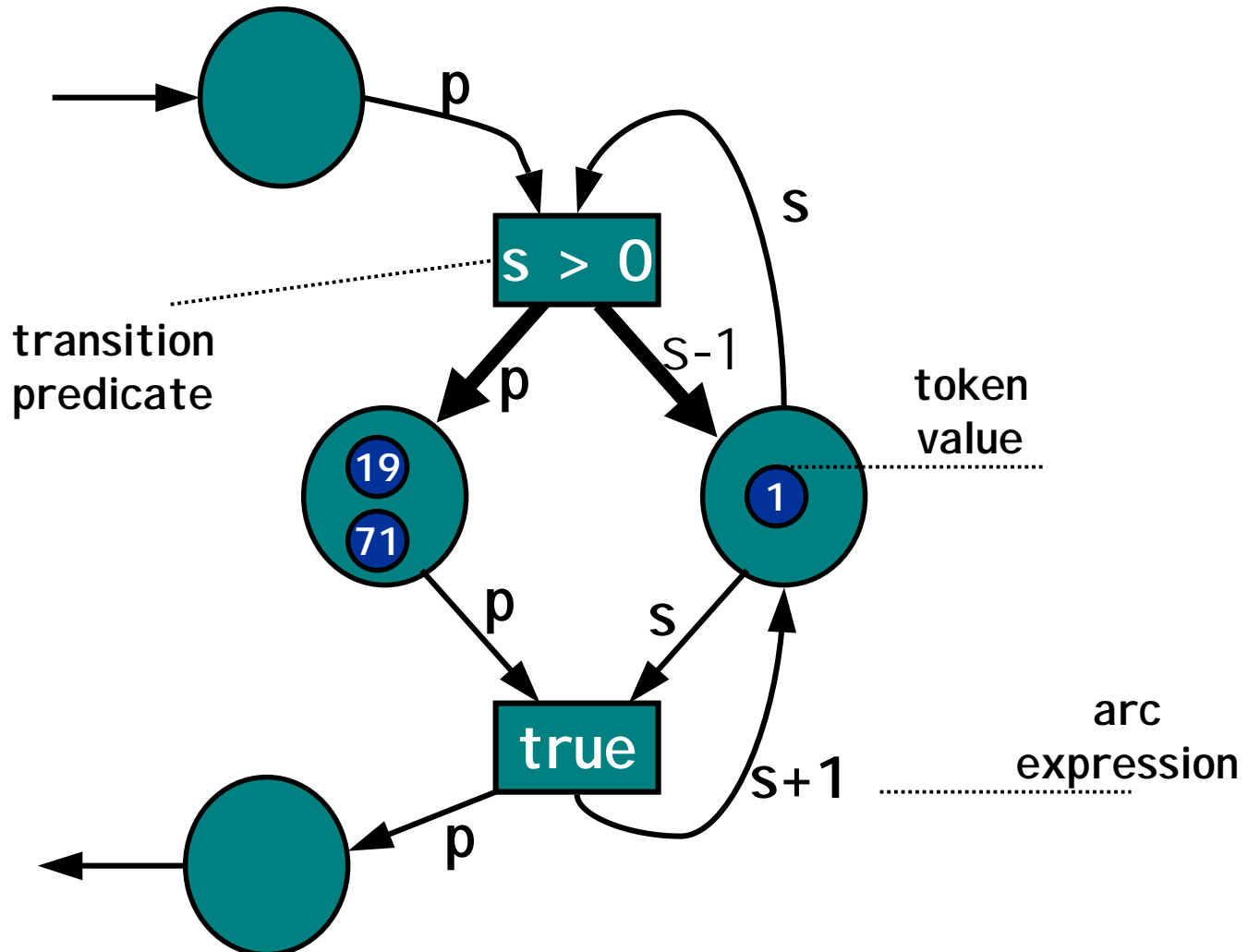
# After Firing



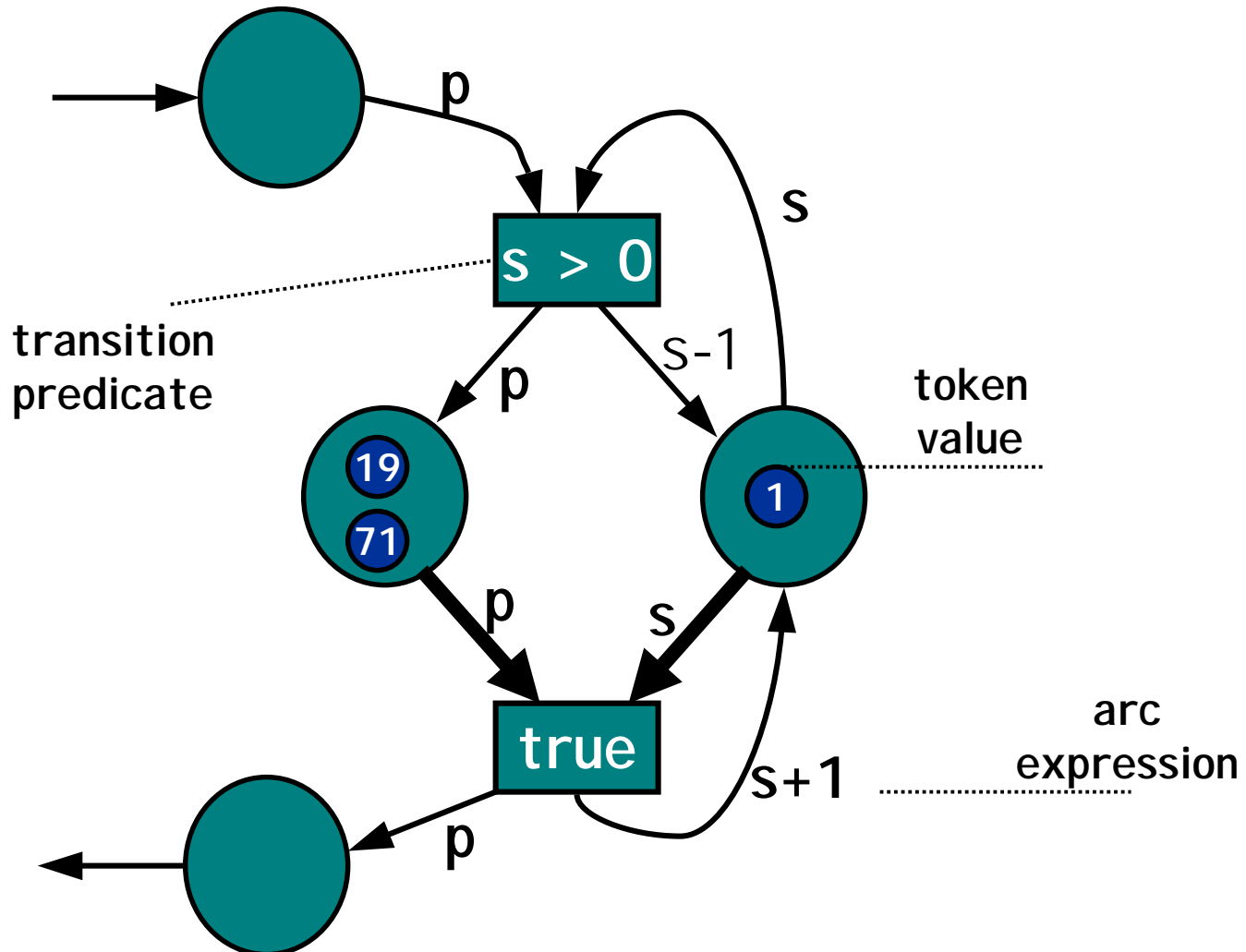
# Enabled Transitions



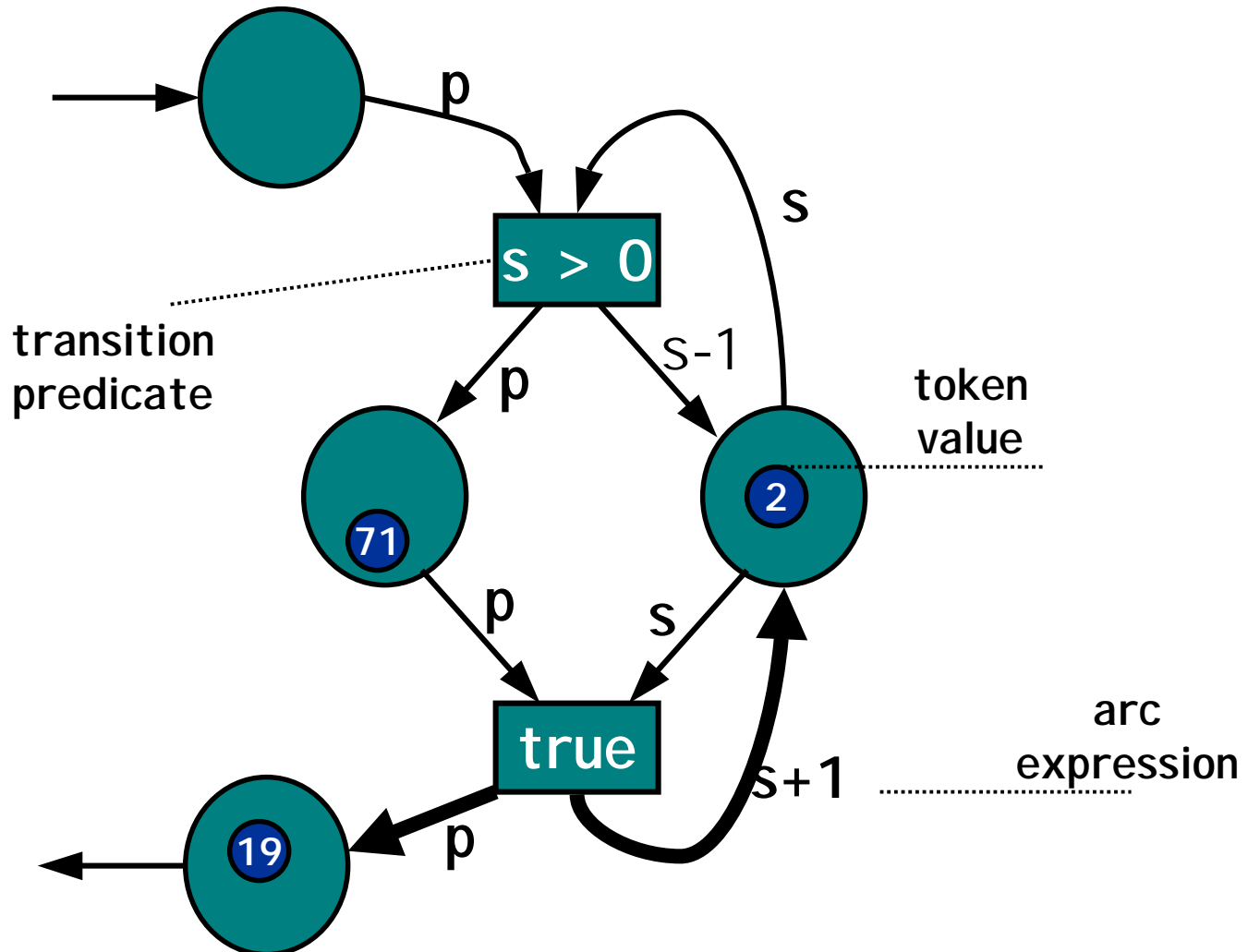
# After Firing



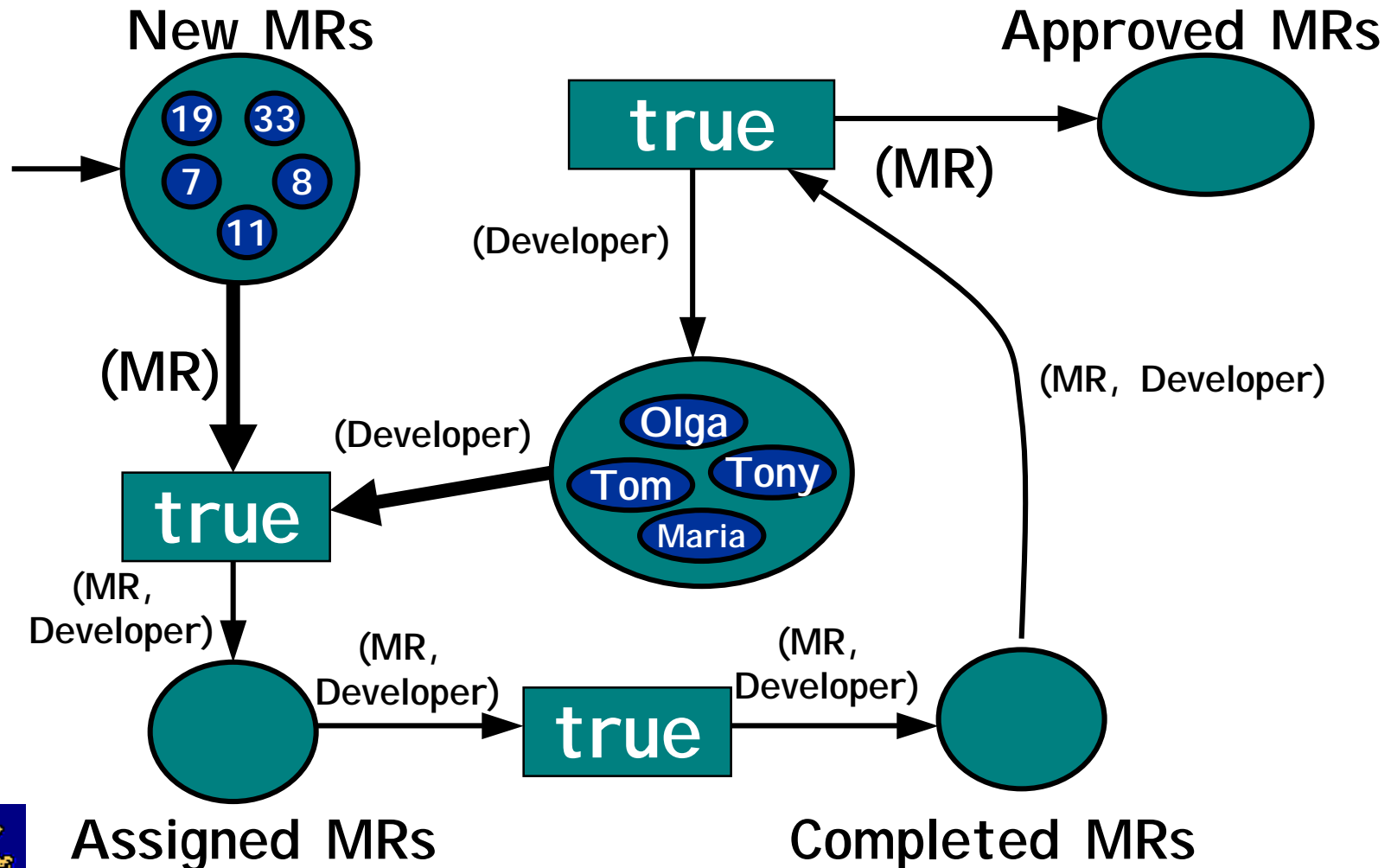
# Enabled Transition



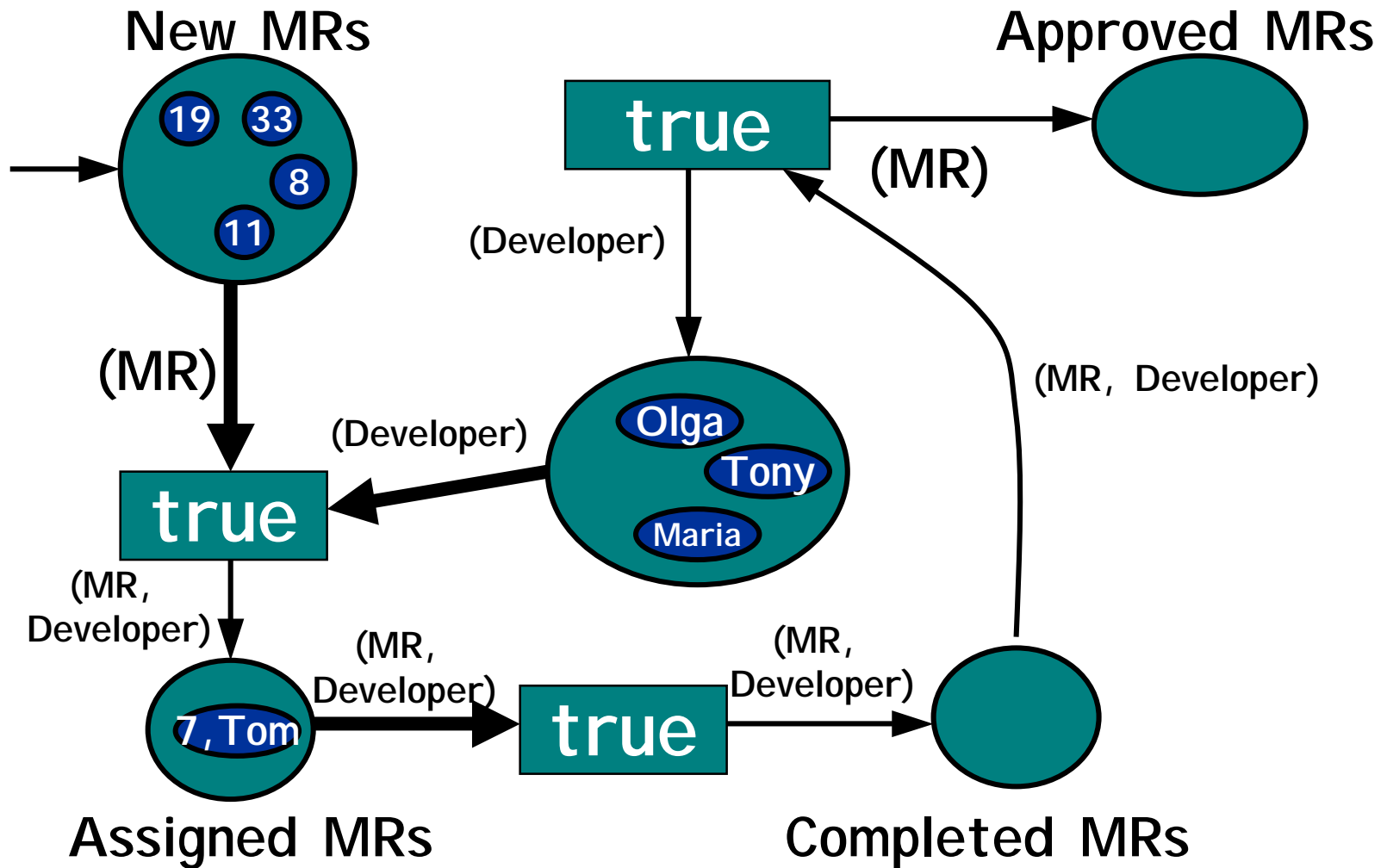
# After Firing



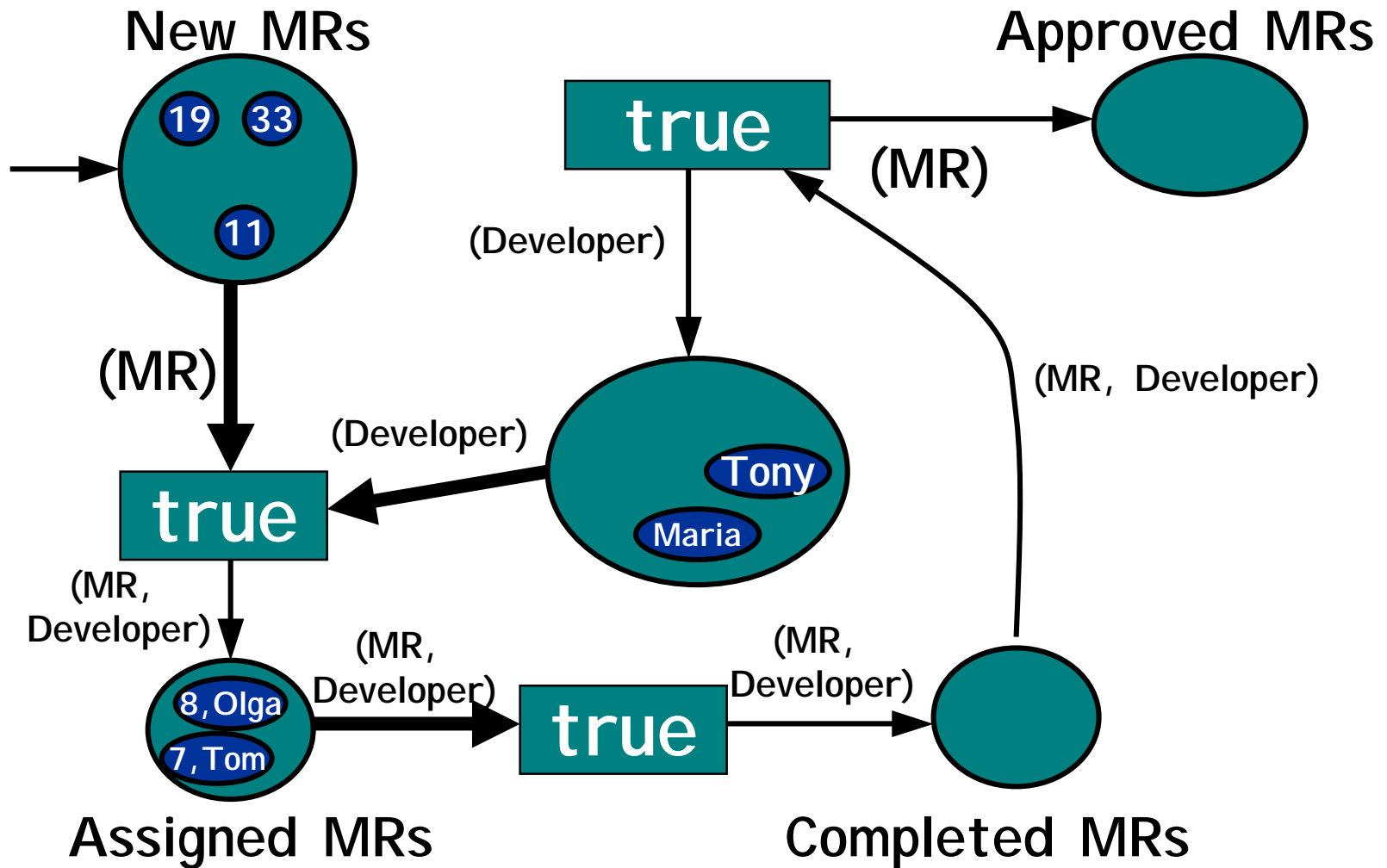
# A Software Change Process



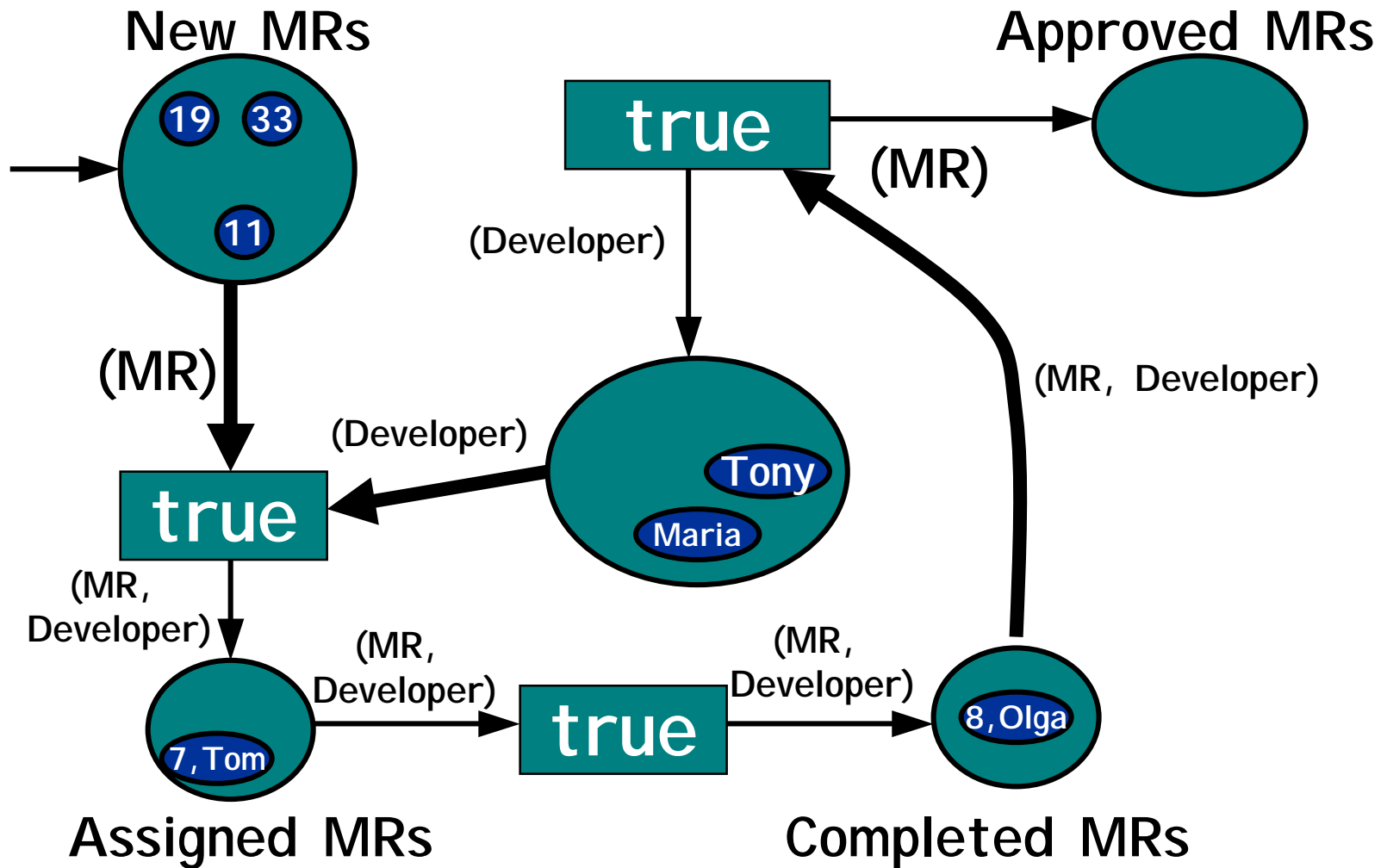
# After Firing (New Assigned MR)



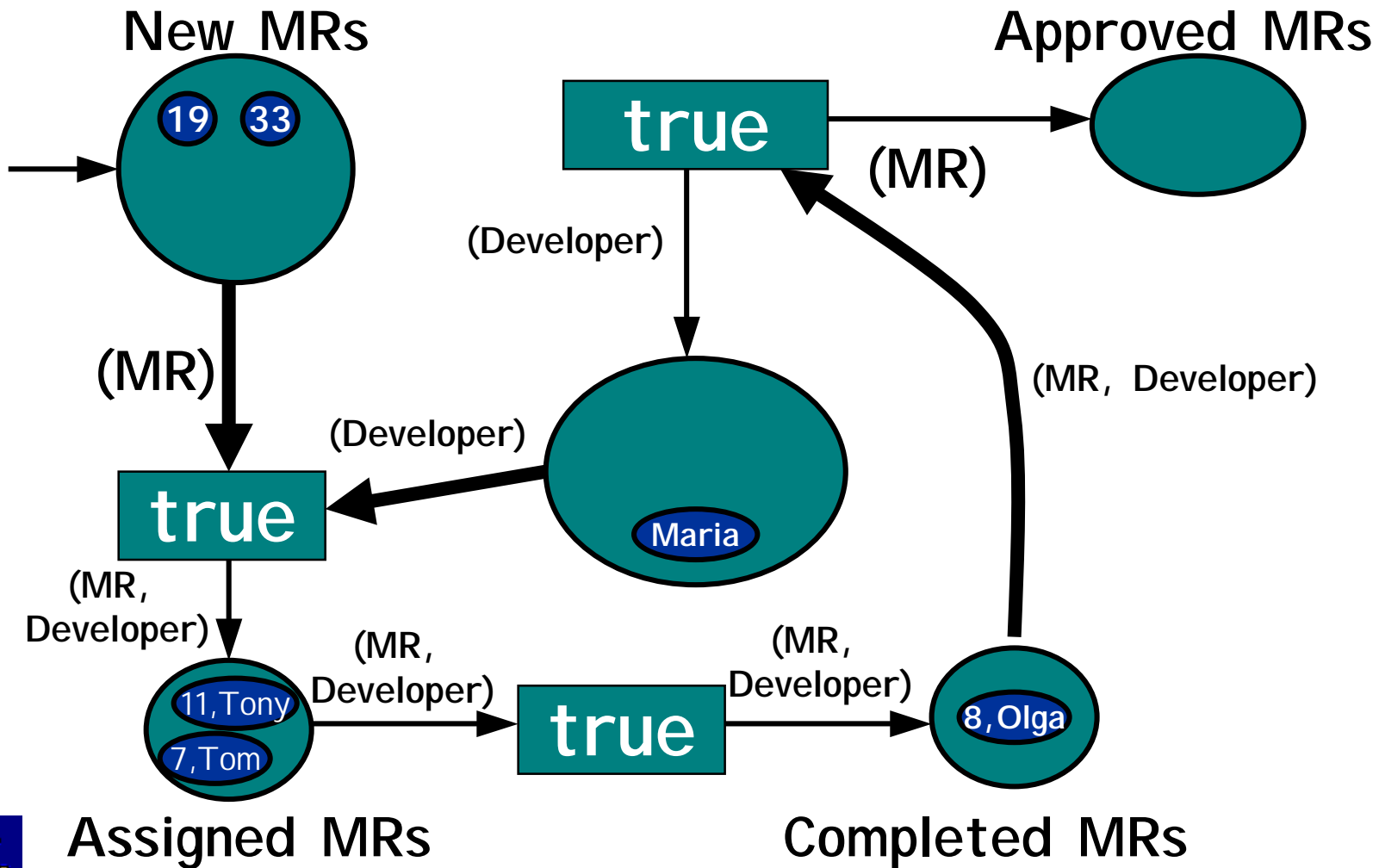
# After Firing (New Assigned MR)



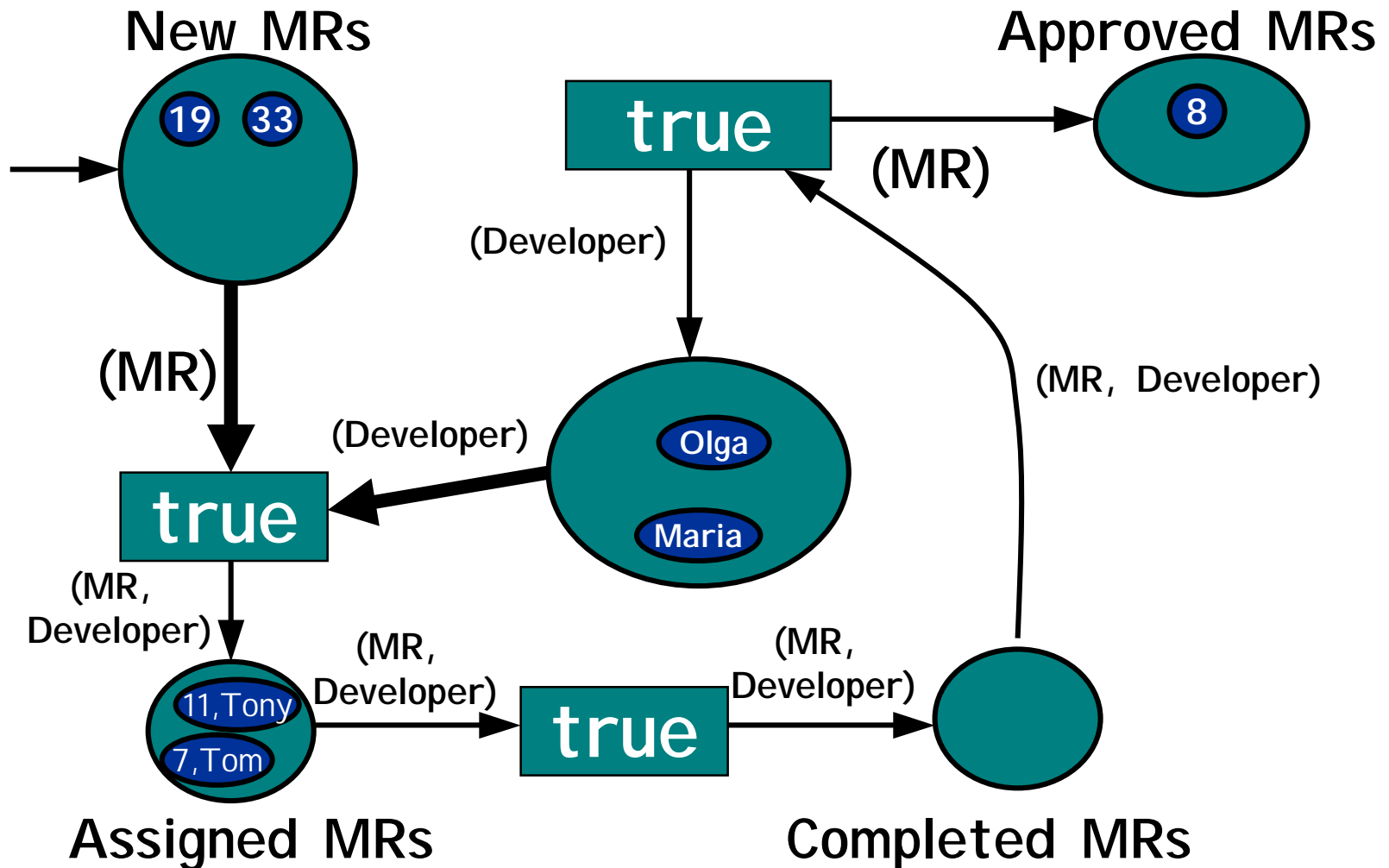
# After Firing (New Completed MR)



# After Firing (New Assigned MR)



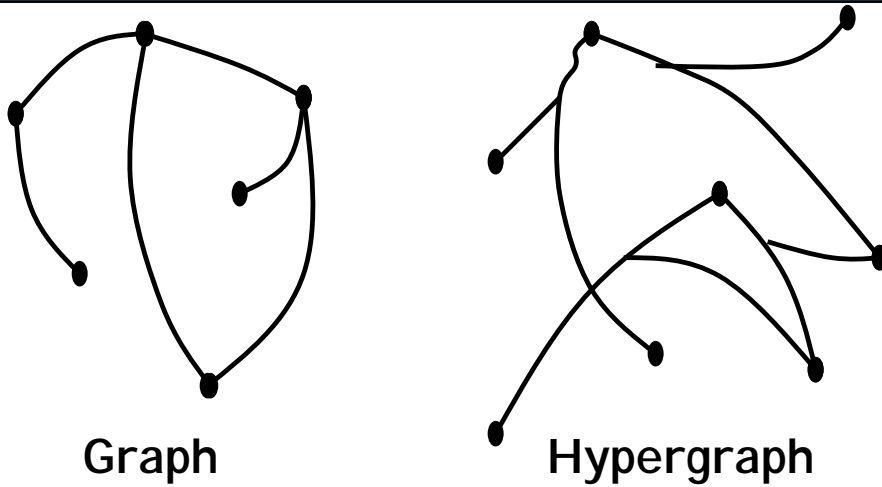
# After Firing (New Approved MR)



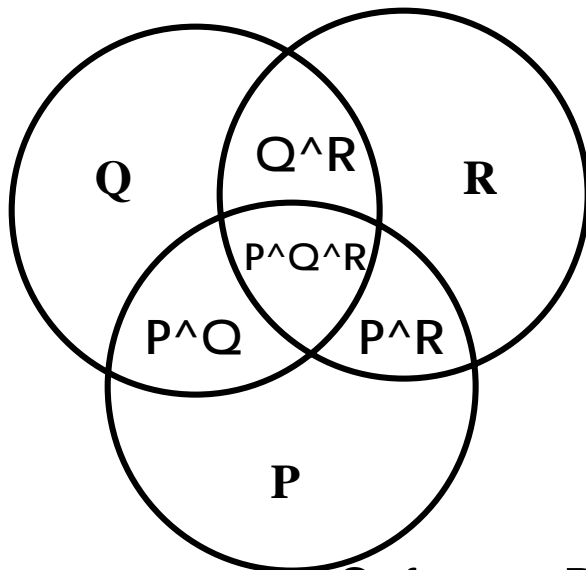
# *Higraphs and Statecharts*

---

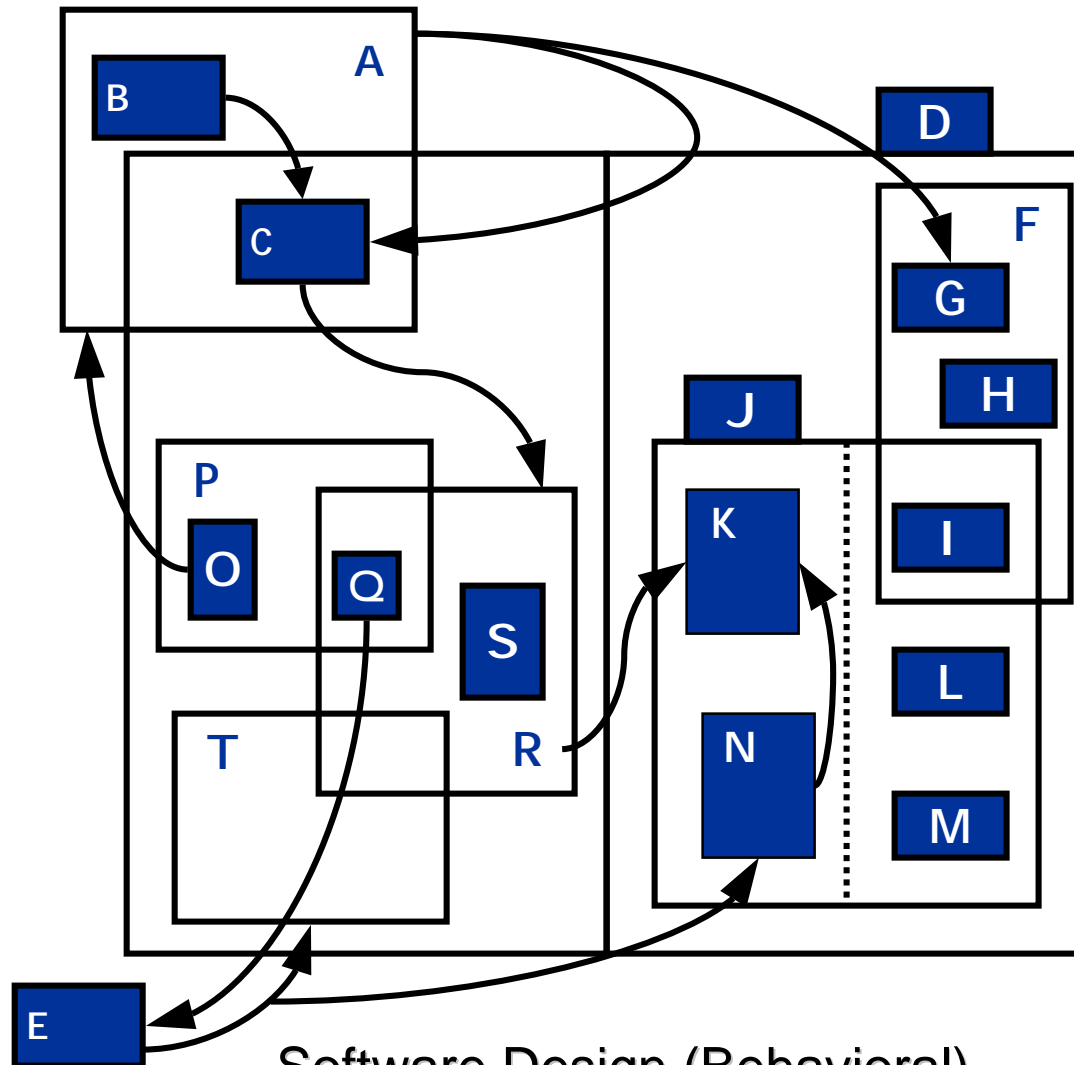
# Higraphs



- Higraphs are based on:
  - Euler graphs
  - hypergraphs
  - Venn diagrams



# *Higraphs Supports Cartesian Products.*



# *Formal Definition of a Higraph*

---

$H = (B, \Sigma, \pi, E)$ , where

$B$  is a finite set of elements (blobs)

$\Sigma$  is the sub-blob function  $\Sigma: B \rightarrow 2^B$

$\pi$  is the partitioning function

$$\pi: B \rightarrow 2^{B \times B}$$

$$(2^{B \times B} = 2^B \times \dots \times 2^B)$$

$E$  is the set of edges  $E \subseteq B \times B$

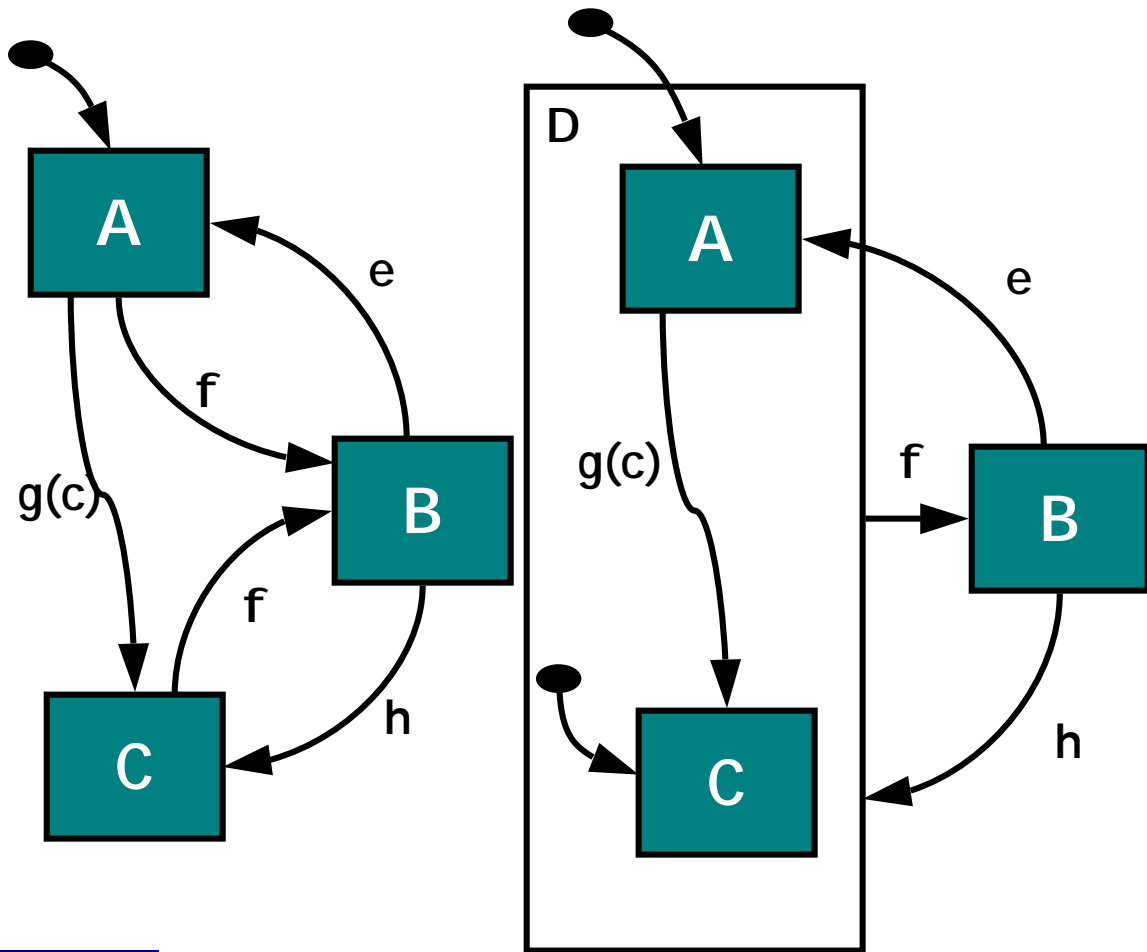
# *Specialized Higraphs:*

## *State Charts*

---

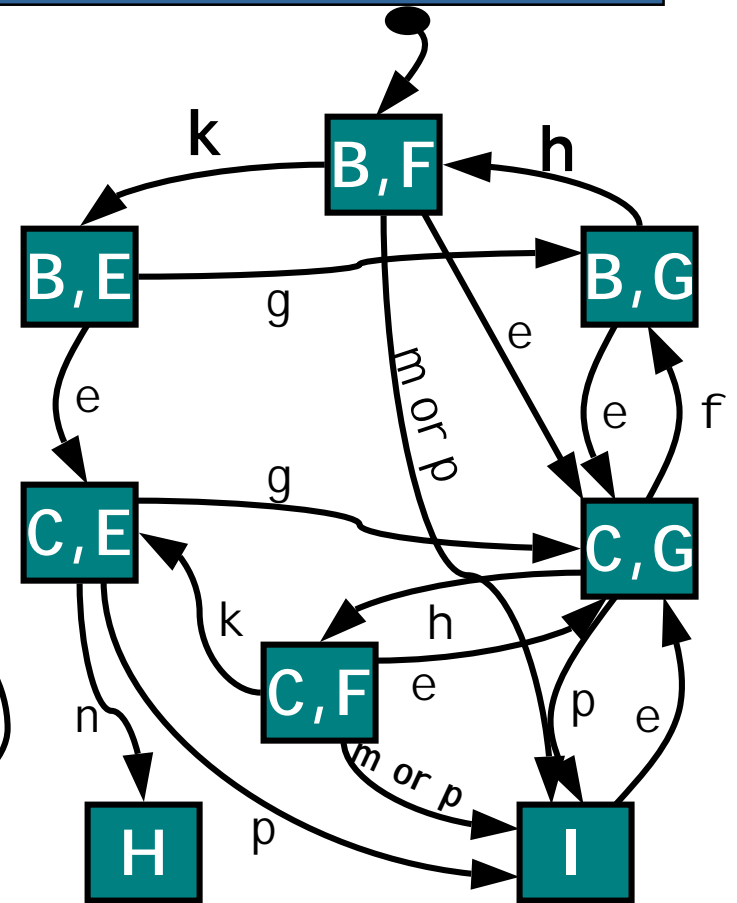
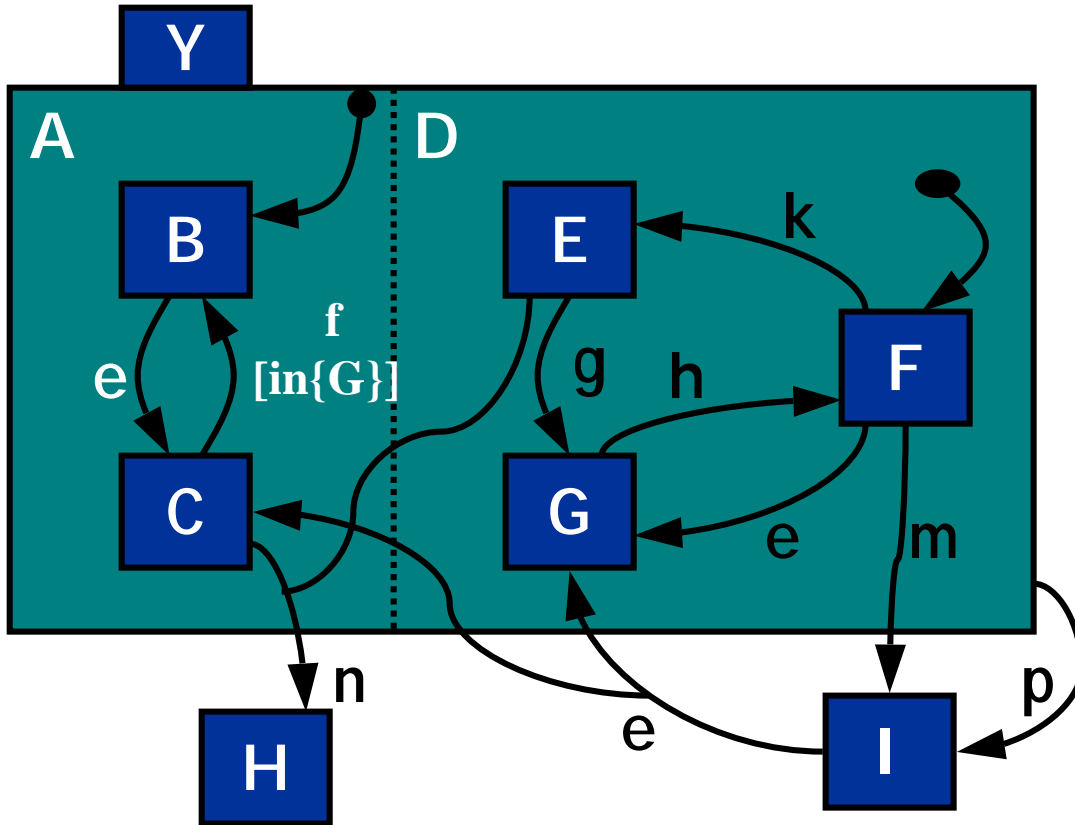
- **State Charts** are a higraph-based extension of standard state-transition diagrams, where blobs represent states and arrows represent transitions.
- *State Charts = state diagrams + depth + orthogonality + broadcast communication*

# Depth of State Charts

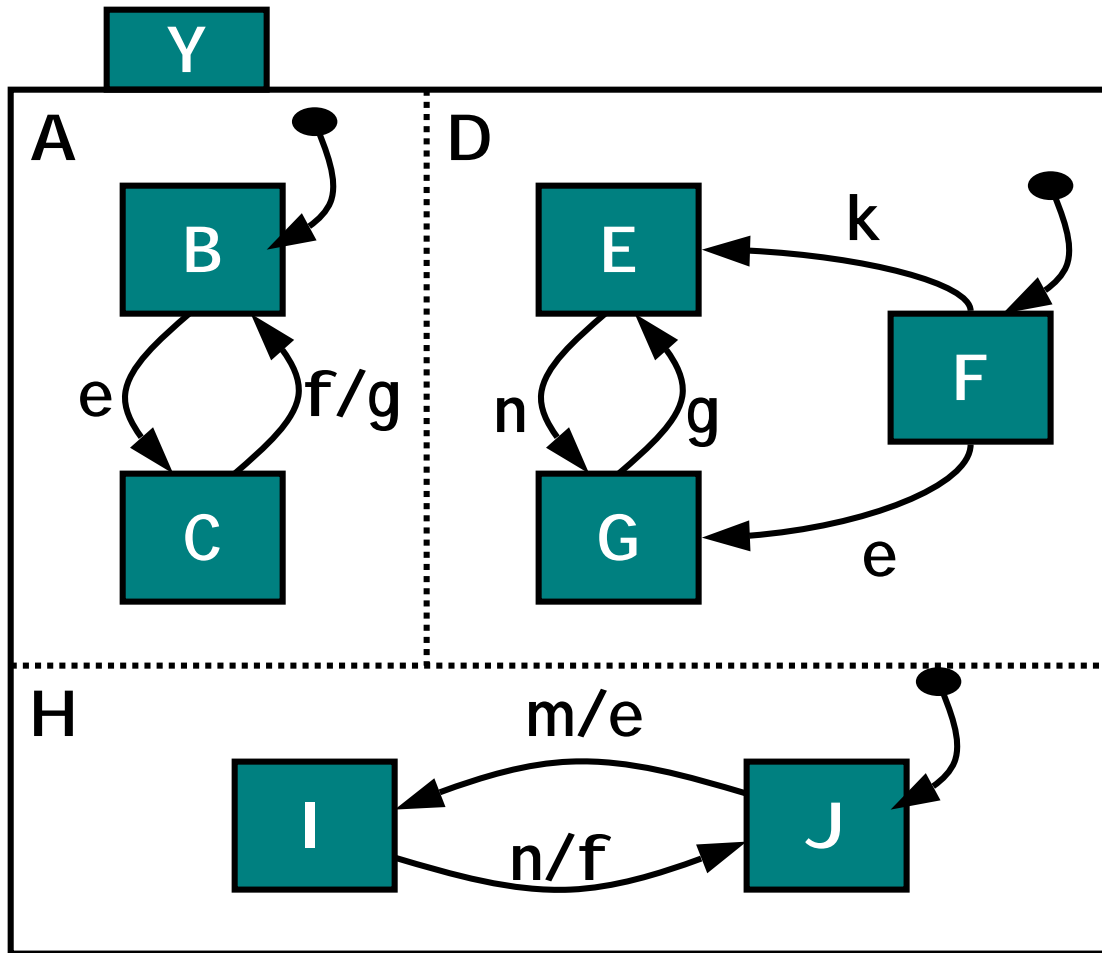


- $e, f, g, h$ : events that trigger the transitions
- $g(c)$ : is the transition by event  $g$  when condition  $c$  is true.
- Being in state  $D$  means being in one of states  $A$  or  $C$ .
- The  $f$  arrow leaving  $D$  applies to both  $A$  and  $C$ .
- $A$  is the default state.
- $C$  is the default state when in  $D$ .

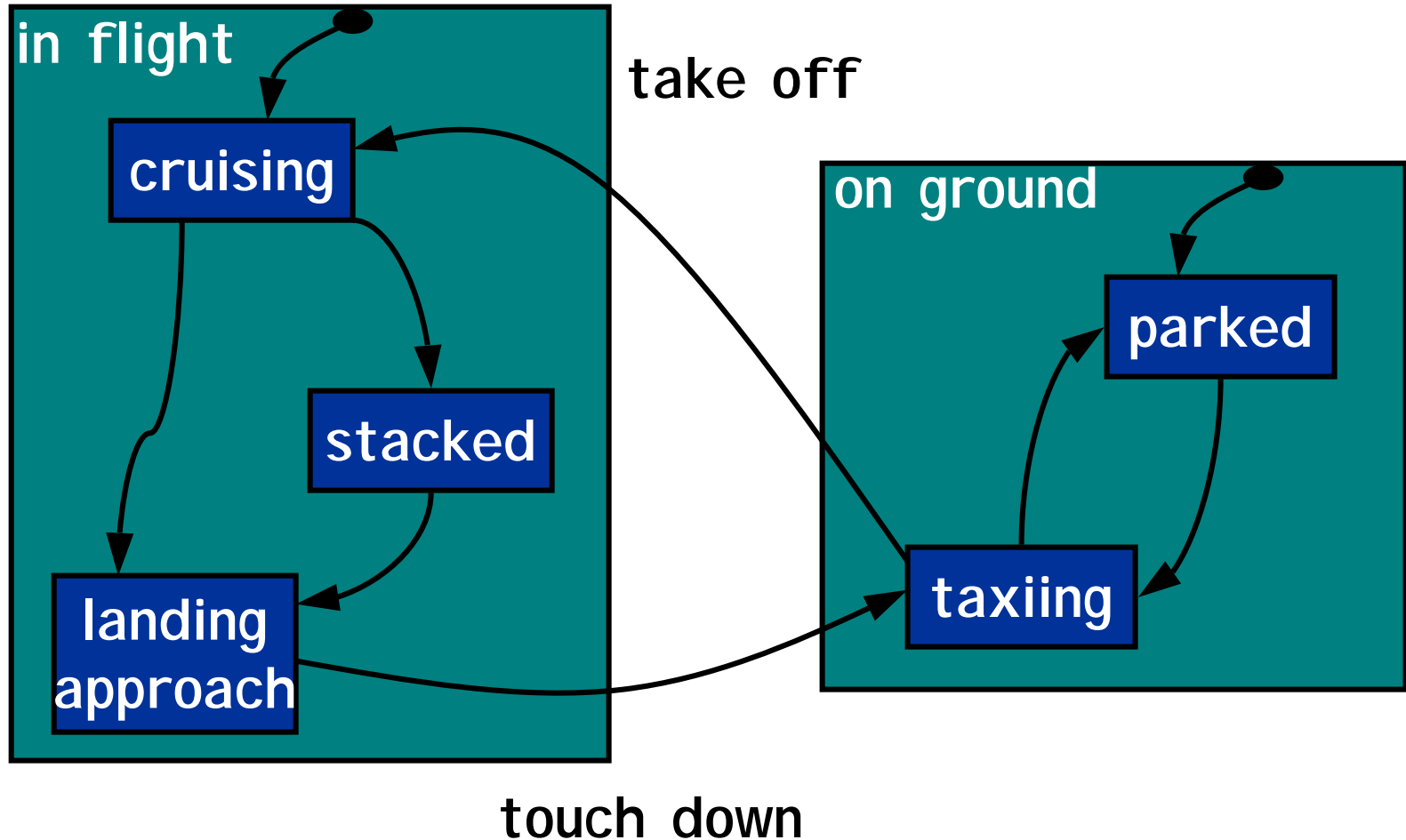
# Orthogonality of State Charts



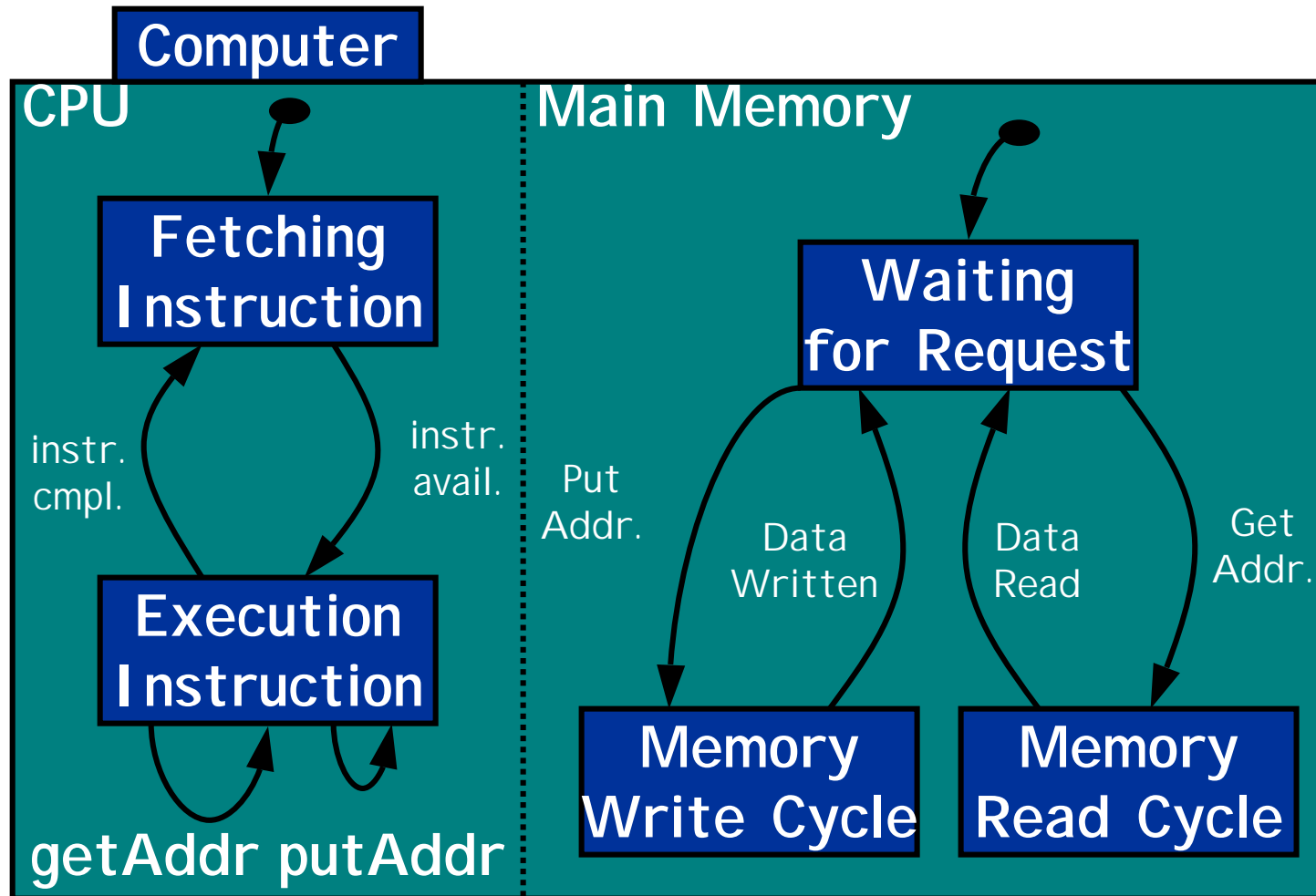
# Broadcasting of State Charts



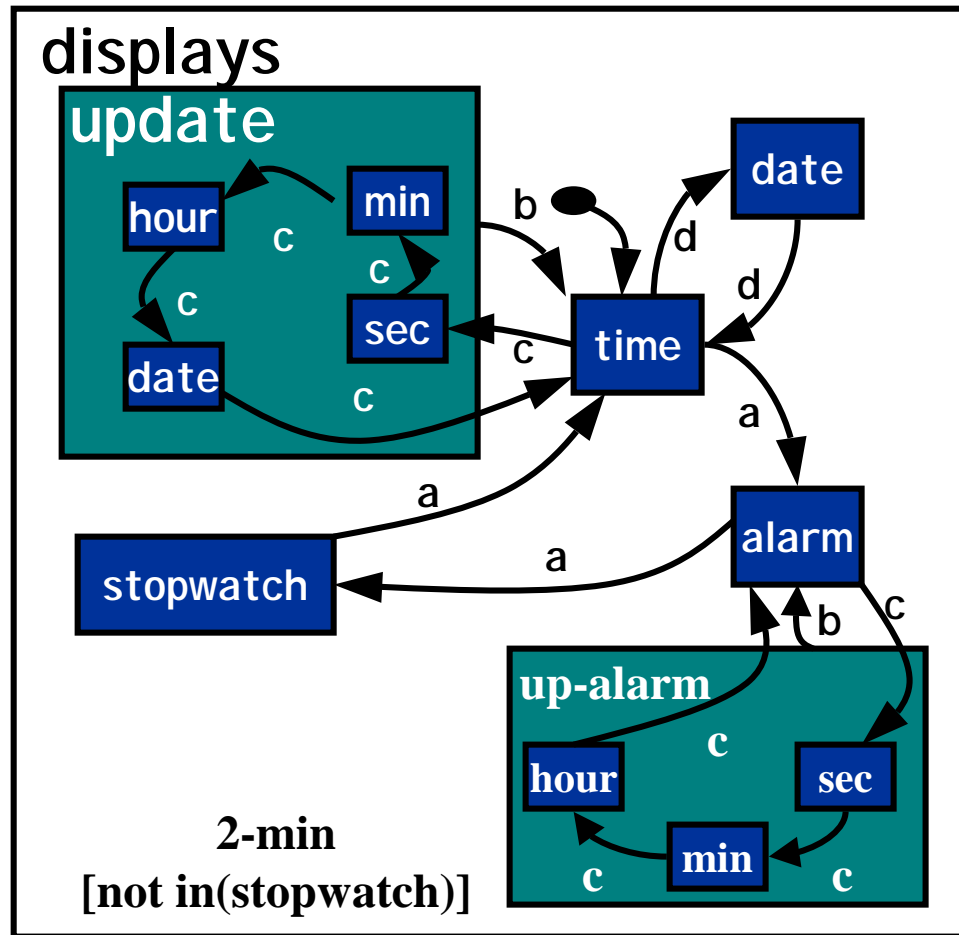
# *State Chart Describing ATC*



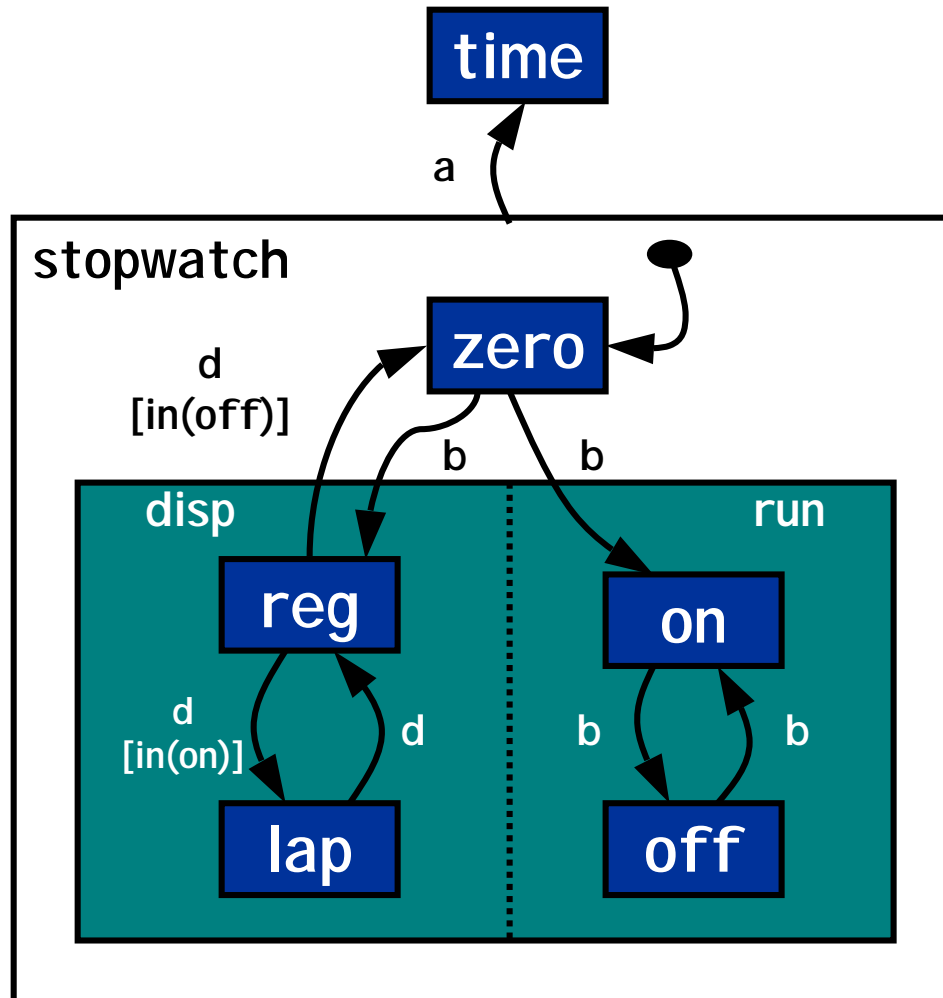
# State Chart Describing a Computer



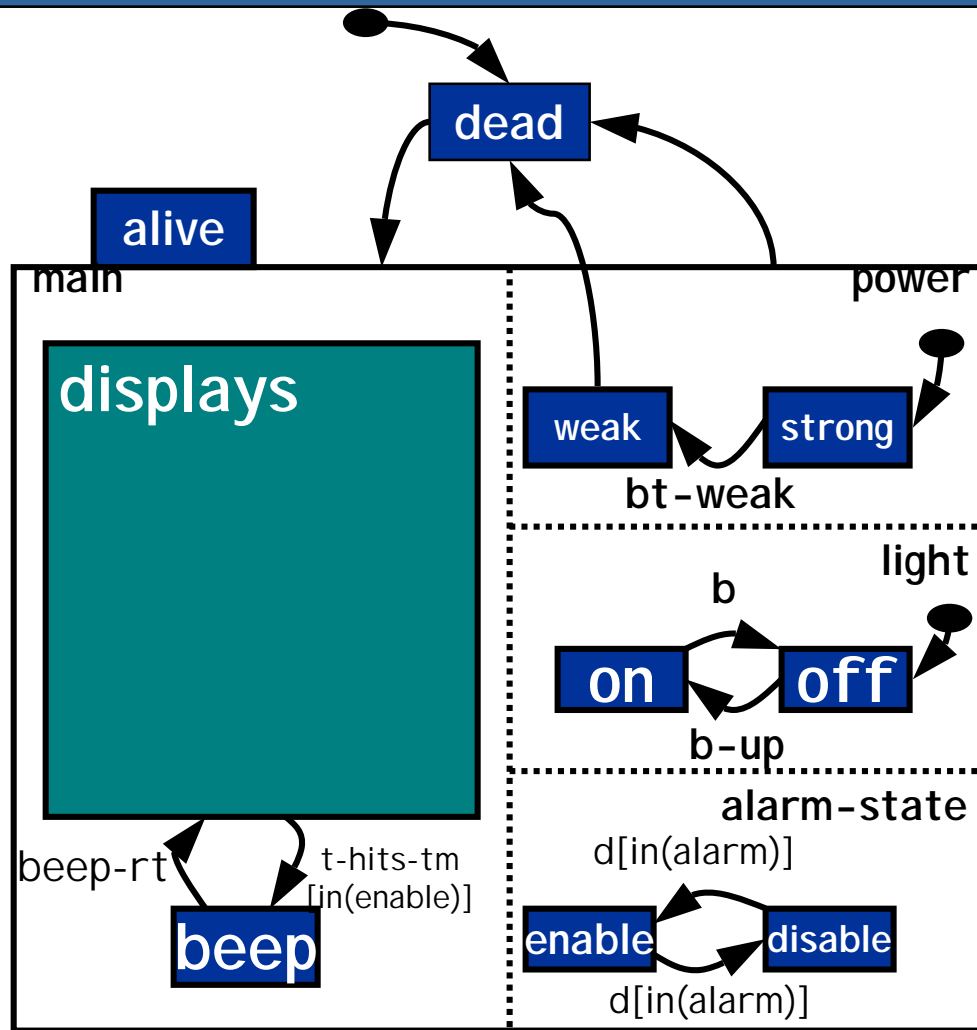
# Display State of Digital Watch



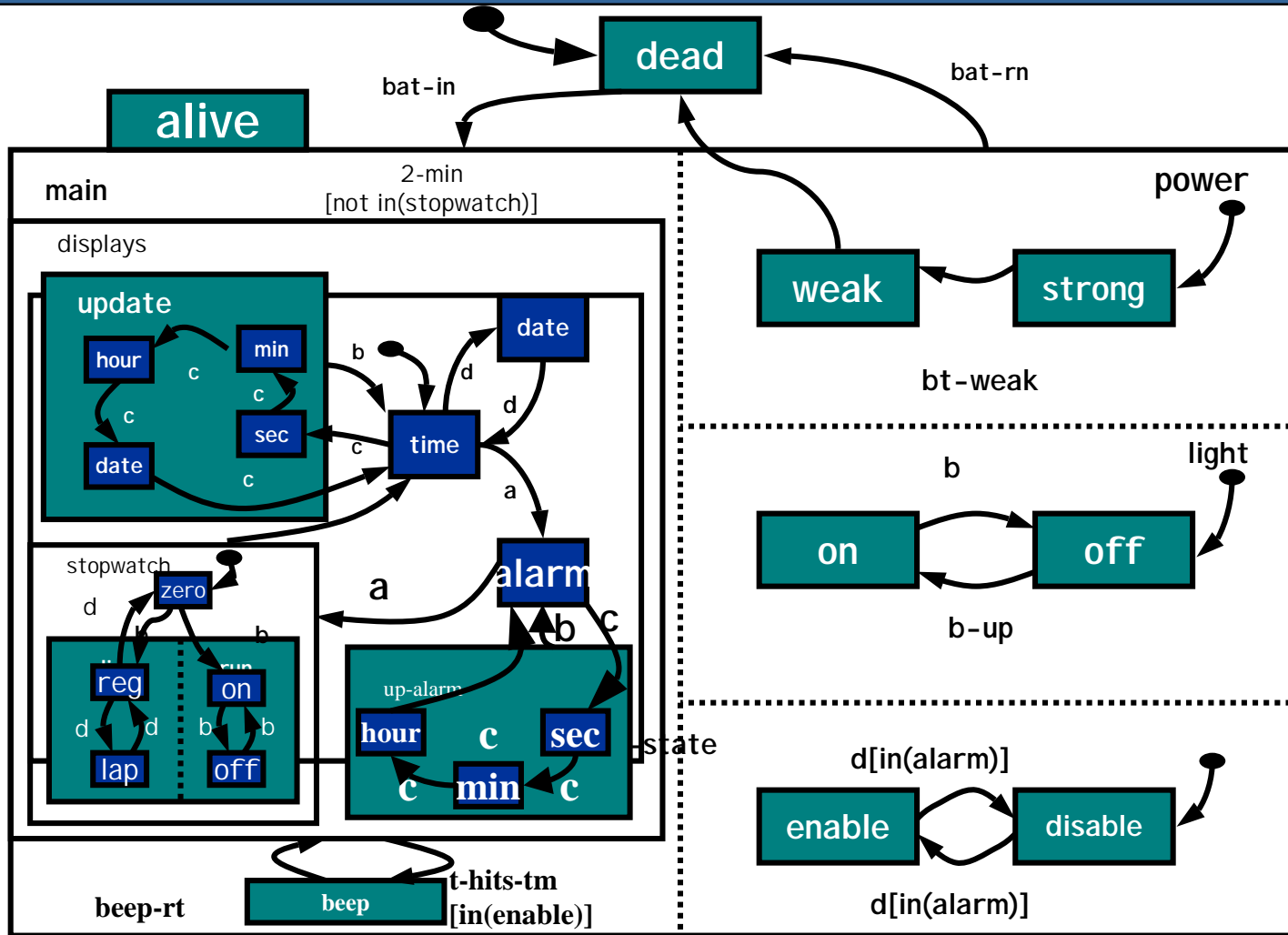
# Stopwatch State of Digital Watch



# High-Level Description of Digital Watch



# State Chart of Digital Watch



# References

---

- [Rosenblum94] D. Rosenblum, A. L. Wolf, Formal Software Engineering, Tutorial SIGSOFT'94 FSE, New Orleans, Dec., 1994.
- [Budgen94] D. Budgen, Software Design, Addison-Wesley, 1994.
- [Ghezzi91] C. Ghezzi, M. Jazayeri, D. Mandrioli, Fundamentals of Software Engineering, Prentice-Hall, 1991.
- [Harel88] D. Harel, On Visual Formalisms, CACM, Vol. 31, No. 5, 1988.