Safety Analysis Using Petri Nets

IEEE Transactions on Software Engineering (1987)
Nancy G. Leveson and Janice L. Stolzy

Park, Ji Hun
2010.06.21

KAIST SE LAB
KAIST Software Engineering Laboratory

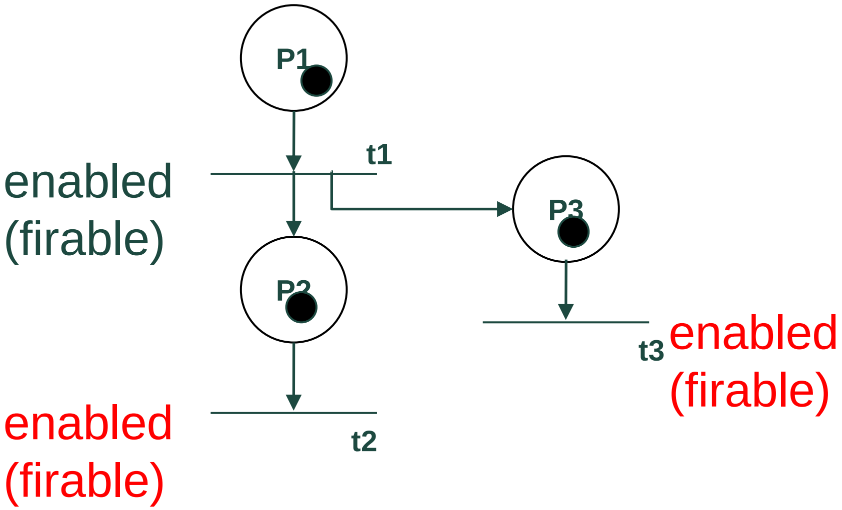
Contents

Introduction

- Motivation
  - Safety is important especially when it involves serious danger to human life and property
  - Software safety should be considered as a whole system including hardware and human, and they can be represented by Petri net
  - In real-time safety critical system, timing information is very important
- Goal of this paper
  - Suggest how to identify high-risk states and eliminate them
  - Suggest how to analyze failure using Petri net

Background (1/3)

- Petri net
  - Places P
  - Transitions T
  - Input functions I
  - Output functions O
  - Initial marking μ_0



$$P = \{P_1, P_2, P_3\}$$

$$T = \{t_1, t_2, t_3\}$$

$$\mu_0 = \{1,0,0\}$$

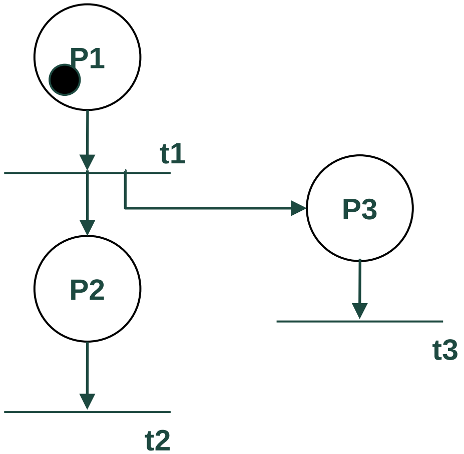
$$I(t_1) = \{P_1\} \qquad O(t_1) = \{P_2, P_3\}$$

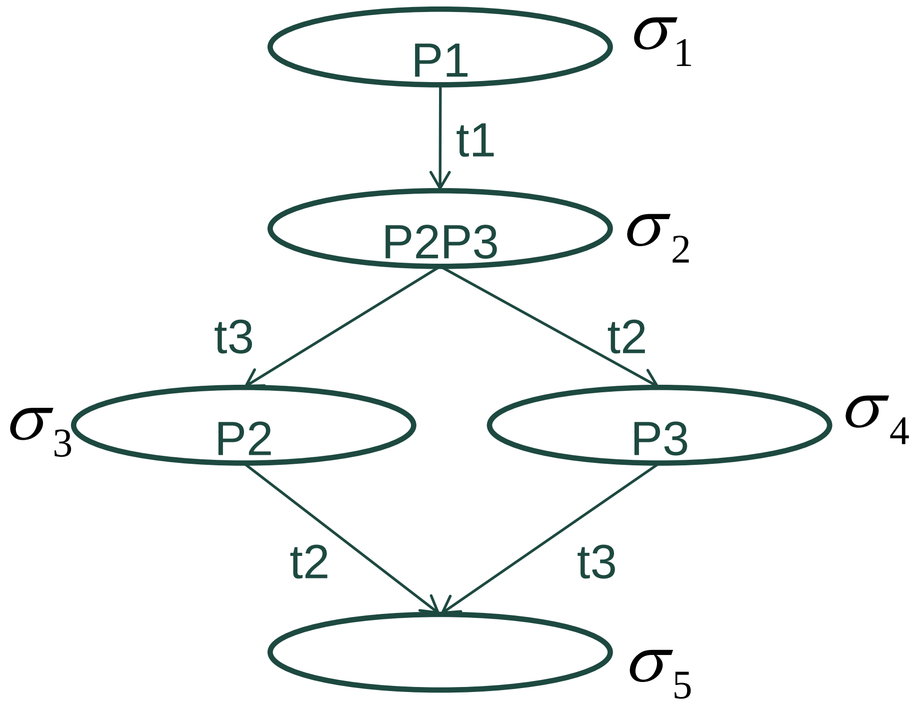
$$I(t_2) = \{P_2\} \qquad O(t_2) = \{\}$$

$$I(t_3) = \{P_3\} \qquad O(t_3) = \{\}$$

Background (2/3)

- Petri net(cont'd)
  - Reachability graph
  - Next-state function δ





$$\delta(\sigma_1, t_1) = \sigma_2$$

$$\delta(\sigma_2, t_3) = \sigma_3$$

$$\delta(\sigma_2, t_2) = \sigma_4$$

...

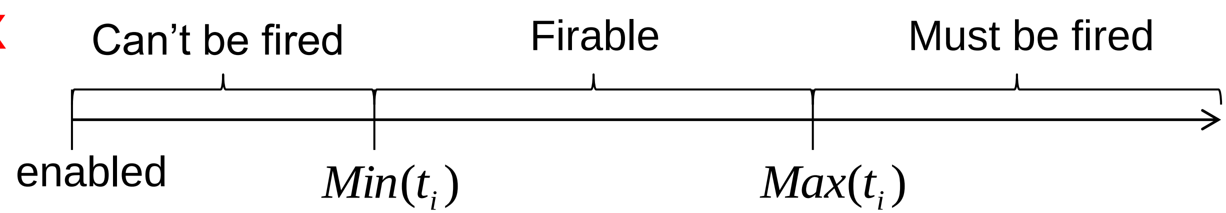
Background (3/3)

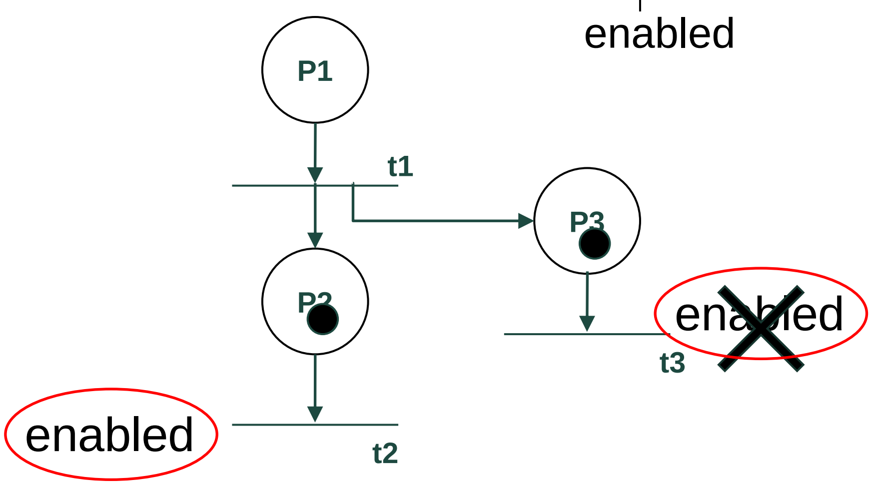
Time petri net

- Places P
- Transitions T
- Input functions I
- Output functions O
- Initial marking μ_0
- Reachability graph
- Next state function
- Min and Max

When the transition t_i is enabled,

- Must wait at least during $Min(t_i)$
- If wait more than $Max(t_i)$, It should be fired





$$Max(t_2) < Min(t_3)$$

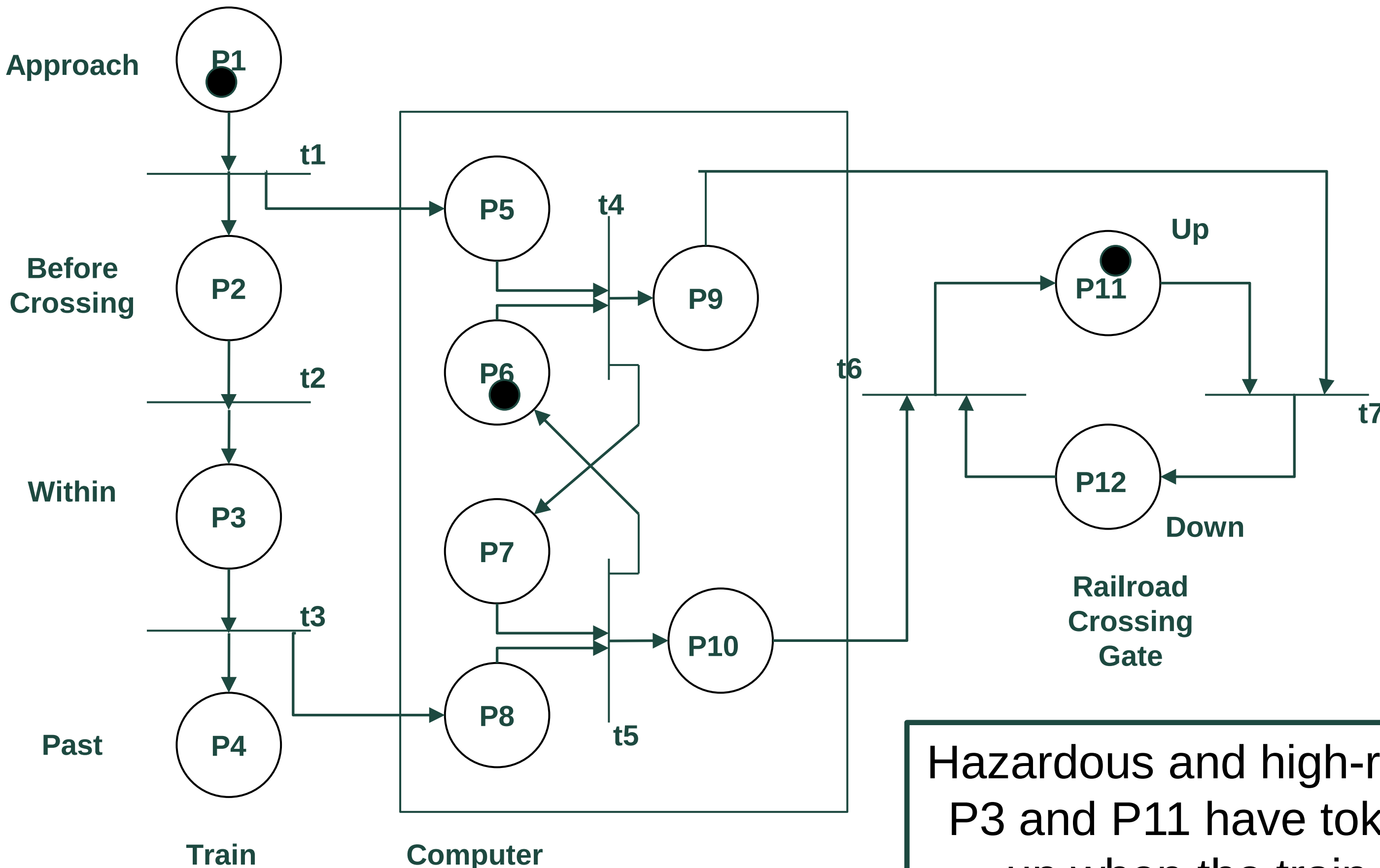
Safety analysis (1/6)

- Mishap and hazard
  - Mishap : An unplanned event or series of events that results in death, injury or damage to property or equipment
  - Hazard : A set of conditions which could cause a mishap

- Properties of hazard
  - Severity : High-risk and low-risk
  - Probability : Not considered in this paper

Safety analysis (2/6)

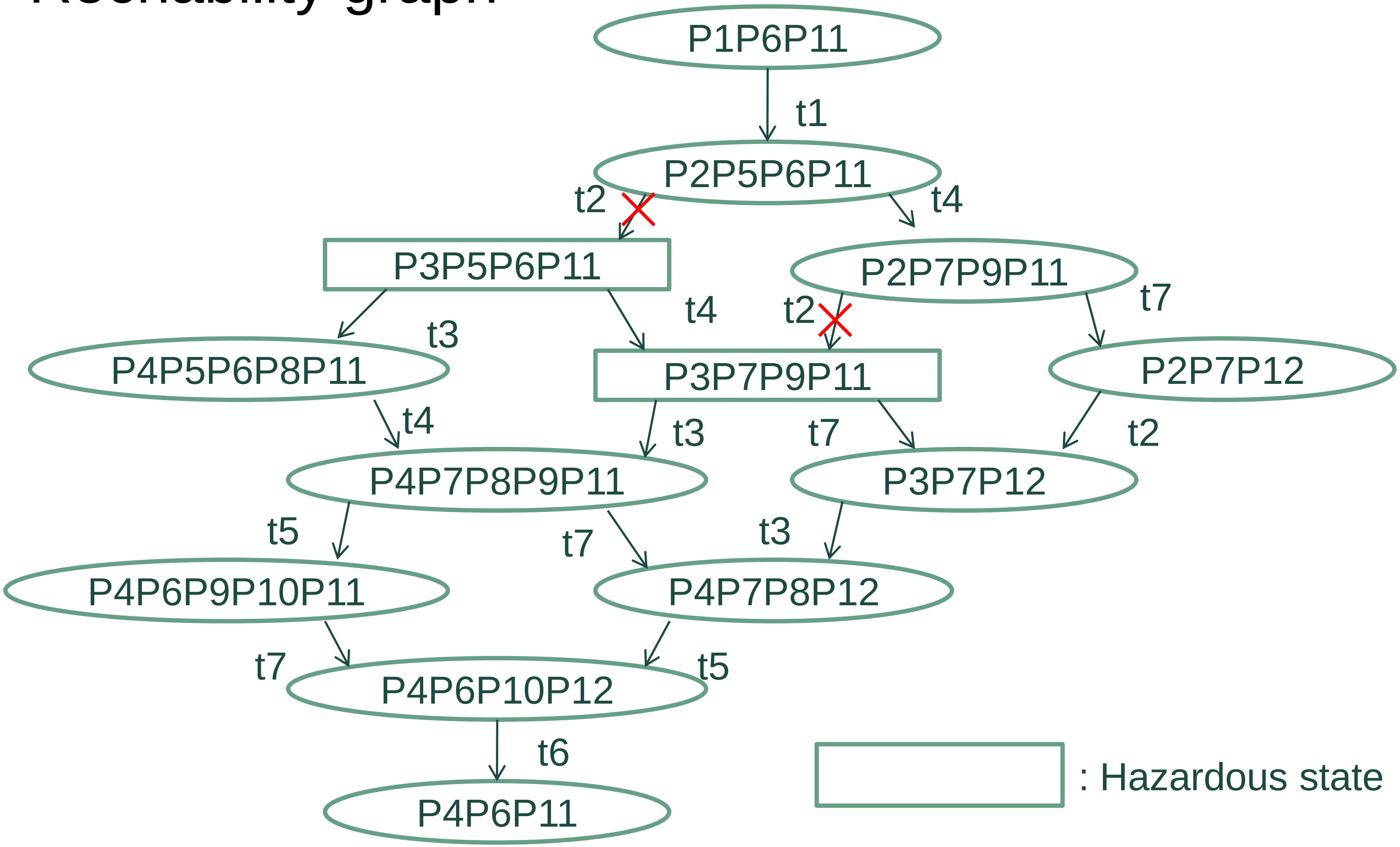
❖ Example of safety-critical system



Hazardous and high-risk when both P3 and P11 have tokens : Gate is up when the train is passing

Safety analysis (3/6)

- Rechability graph



Safety analysis (4/6)

- Identifying high-risk state

Problem of creating full reachability graph

Size of the graph is impractically large for a complex system

⇩

Backward analysis

Testing whether the high-risk states are reachable
Using Inverse Petri net which is inversed each transition's input places with output places

Problem of Backward analysis

Useful only considering small number of high-risk states
Possibly as large as or even larger than original graph

⇩

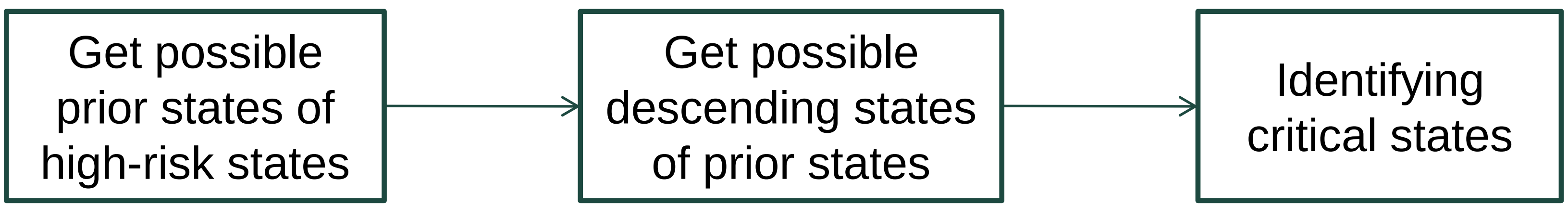
The author's solution

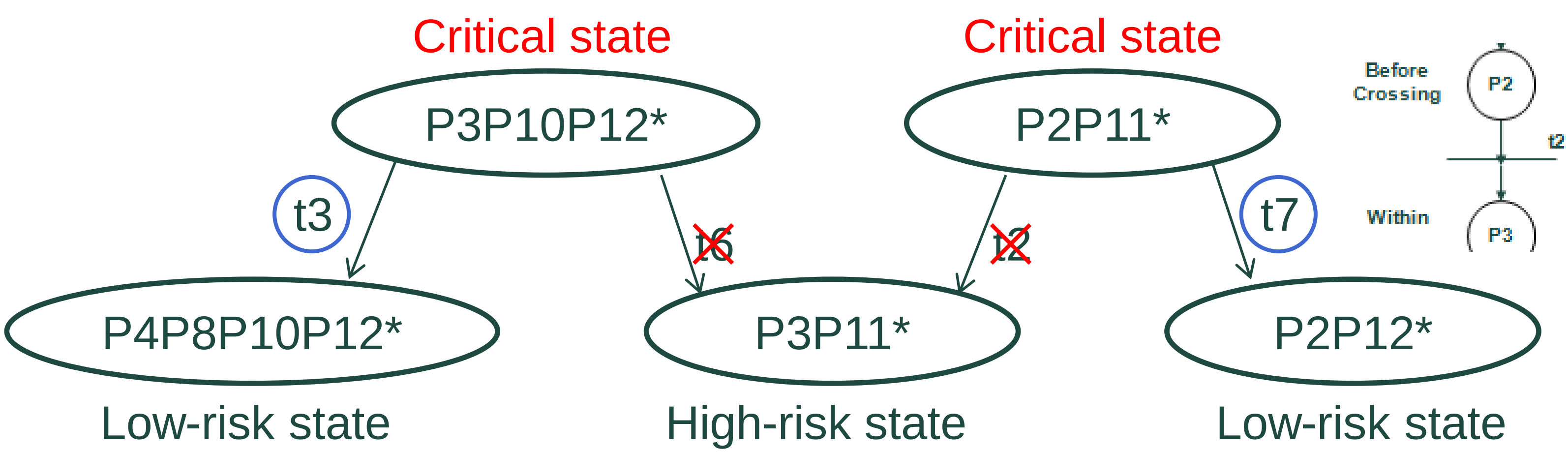
Using particular type of state named 'critical state'
Don't need entire backward reachability graph

Safety analysis (5/6)

- Critical states
  - Low-risk states which has both transitions toward high-risk states and low-risk states
  - By selecting for low-risk states way, high-risk states can be avoided

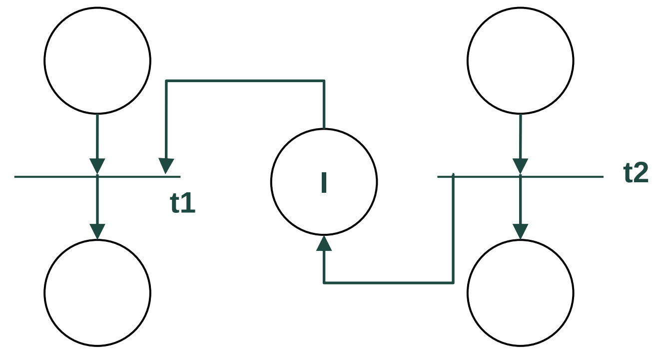
Algorithm



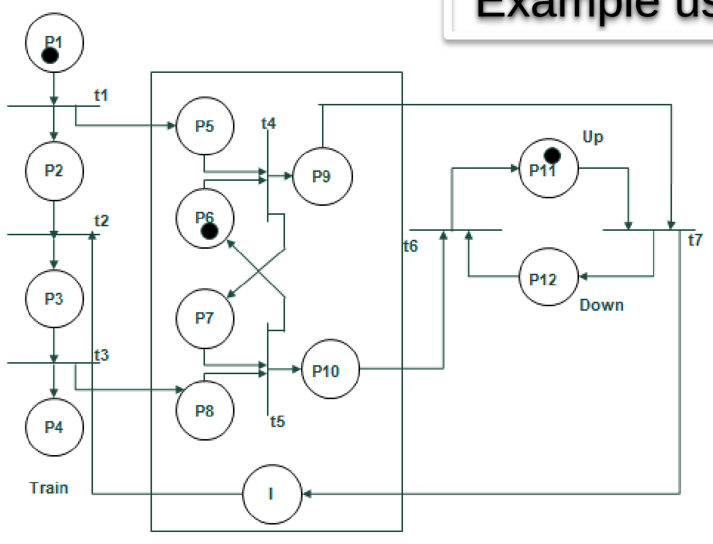


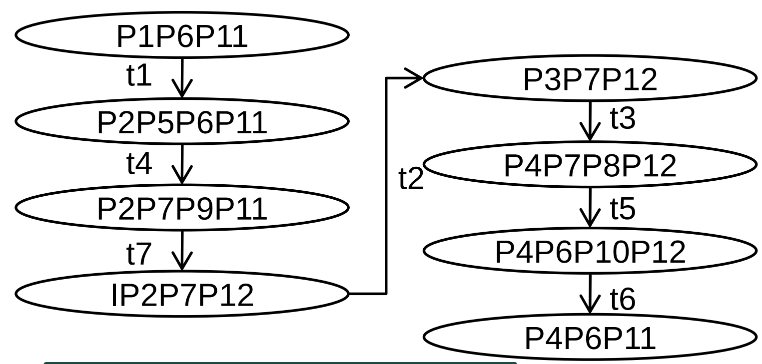
Safety analysis (6/6)

- Eliminating high-risk state
  - Inter lock
    - One event always precedes another events
  - Time constraint
    - $Max(t_2) < Min(t_1)$
    - Determined using reachability graph



Example using interlock





No hazardous state!!

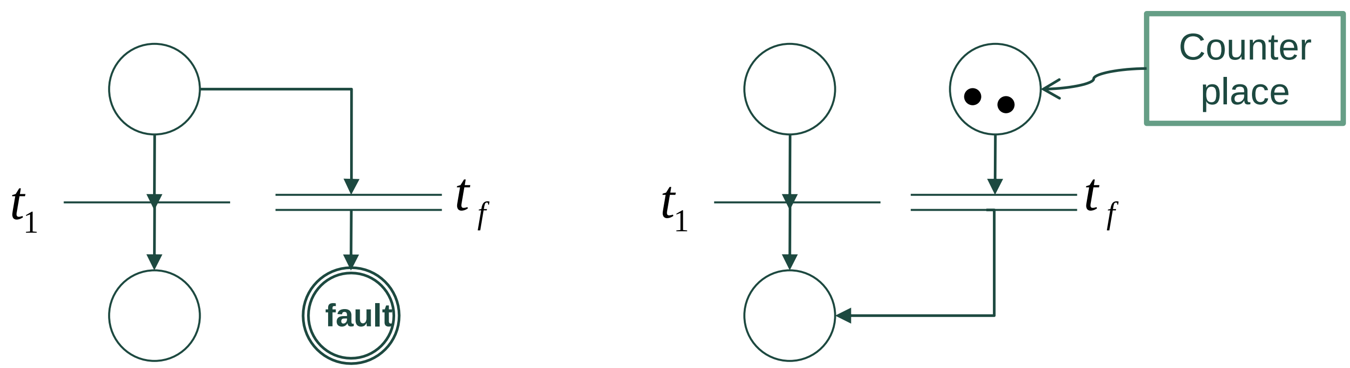
Adding failures to the analysis (1/9)

- Type of control failures
  - A required event that does not occur
  - An undesired event
  - An incorrect sequence of required events
  - Timing failures in event sequences

- IEEE definition of failure (IEEE Std1633-2008)
  - The inability of a system or system component to perform a required function within specified limits

Adding failures to the analysis (2/9)

- Representation of control failure
  - Previous work – Loss of tokens
    - Hard to know circumstance of the failure
  - Author's suggestion – Failure transition and place
    - Legal transition(T_L) and Failure transition(T_F)
    - Legal place(P_L) and Failure place(P_F)



Desired event does not occur

Undesired event occurs

Adding failures to the analysis (3/9)

- Representation of control failure(cont'd)
  - Legal and faulty state
    - Legal state

$$\sigma \text{ is legal state, iff from initial state } \sigma_0$$

$$\exists path(\text{sequence of transition})\ s,\ s \in T_L^*,\ \delta^*(\sigma_0, s) = \sigma$$

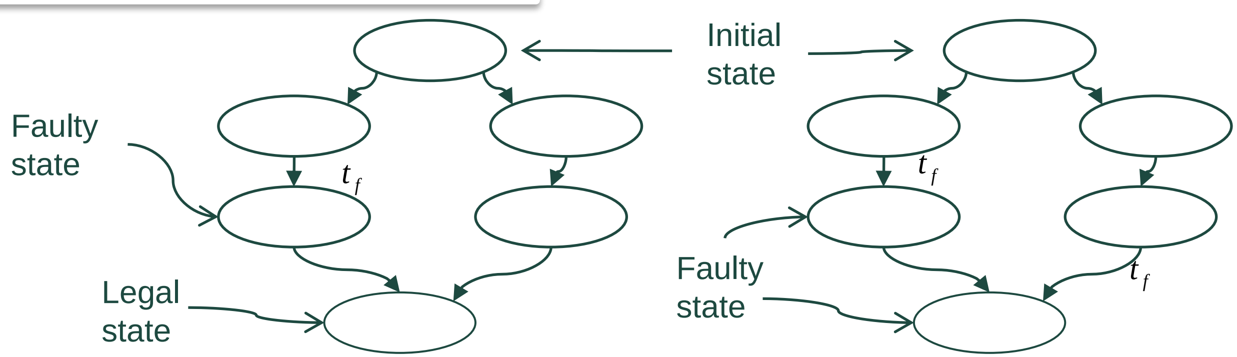
    - Faulty state

$$\sigma \text{ is faulty state, iff from initial state } \sigma_0$$

$$\forall path(\text{sequence of transition})\ s,\ \delta^*(\sigma_0, s) = \sigma,$$

$$\exists t_f \in T_f \text{ and } t_f \in s$$

Fault reachability graph

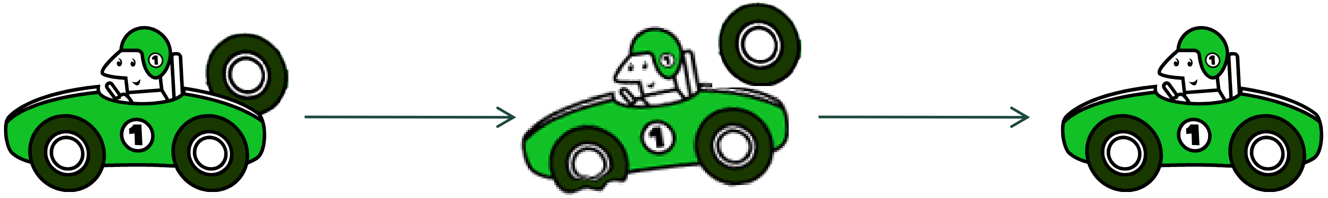


Adding failures to the analysis (4/9)

- Qualities of design associated with failure
  - Recoverability
    - After failure, the control of process is not lost and will return to normal execution within an acceptable amount of time
  - Fault-tolerance
    - The system continues to provide full performance and functional capabilities in the presence of faults
  - Fail-safe
    - The system limits the amount of damage caused by failure and functional requirement could be not satisfied

Adding failures to the analysis (5/9)

- Recoverability
  - Definition
    - Number of faulty states are finite
    - There are no terminal faulty node
    - There are no directed loops including *only* faulty states
    - The sum of maximum times on all paths *from the failure transition to correct state* is less than a predefined acceptable amount of time
  - Problem
    - Once a permanent failure has occurred, the state cannot return to normal unless some repair action has taken place



Normal state(with spare tire) Failure(flat tire) Recovered but not normal (no spare tire)

Adding failures to the analysis (6/9)

Correct behavior path

- Definition
  - Path in reachability graph which contains no failure transition

$$\delta(\sigma_{i-1}, t_i) = \sigma_i, \ for\ i = 1..n\ and\ t_i \in T_L$$

Fault-tolerant

- Definition
  - A correct behavior path is a subsequence of every path from initial to any terminal state
  - Sum of maximum times on all paths is less than predefined acceptable amount of time

$$for\ \ path\ t_1 \ldots t_n\ \ from\ \sigma_0\ to\ \sigma_n,$$
$$\sum Max(t_j) < T_{acceptable}\ \ for\ \ j = 1 \ldots n$$

Adding failures to the analysis (7/9)

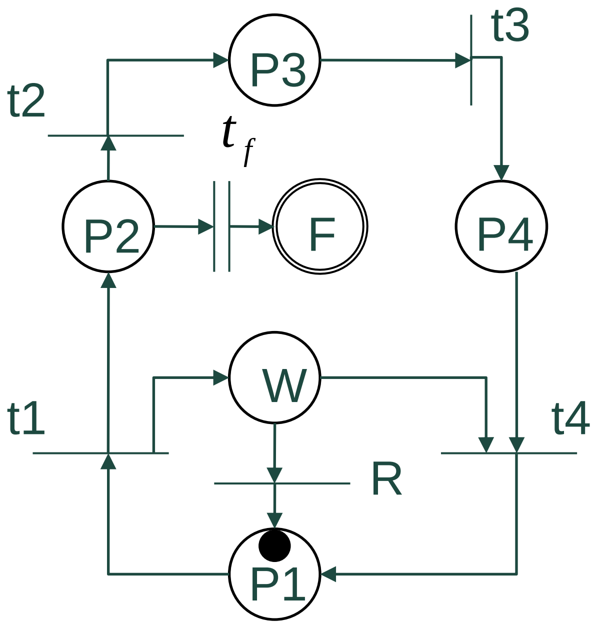
- Fault-tolerant(cont'd)
  - Correct behavior path : $t_1 t_2 t_4 t_8 t_{10}$
  - Initial to final path : $t_1 t_f t_2 t_R t_4 t_8 t_{10}$

    Takes time no more than $T_{acceptable}$

  - Meaning of 'Fault-tolerant'
    - Even if some initial to terminal path has failure transition, the system should be recovered and perform adequately
    - Even if there is failure transition, sum of execution times is less than predefined time

Adding failures to the analysis (8/9)

- Example of fault-tolerant system



- When failure occurs, R could fire then it puts token in P1
- R is firable any time after firing of t1
  - Time constraint is needed

$$Min(R) \geq Max(t_2) + Max(t_3) + Max(t_4)$$

Adding failures to the analysis (9/9)

- Fail-safe
  - Definition
    - All paths from a failure F contain only low-risk states

$$\forall \sigma_f \ \textit{and sequences}\ s_1\ \textit{such that}\ \delta^*(\sigma_0, s_1 F) = \sigma_f$$

$$\neg\exists\ \textit{sequence}\ s_2\ \textit{and}\ \sigma_h \in \textit{high}-\textit{risk states}\ \delta^*(\sigma_f, s_2) = \sigma_h$$

  - Property
    - The system may never get back to a legal state
  - Possible way to design the system
    - The system may be n-fault-tolerant and n+1 fail-safe
    - The system may be fault-tolerant but not fail-safe

Example of safety analysis (1/3)

- Analysis approach

➢ Consider only those failures with the most serious consequences

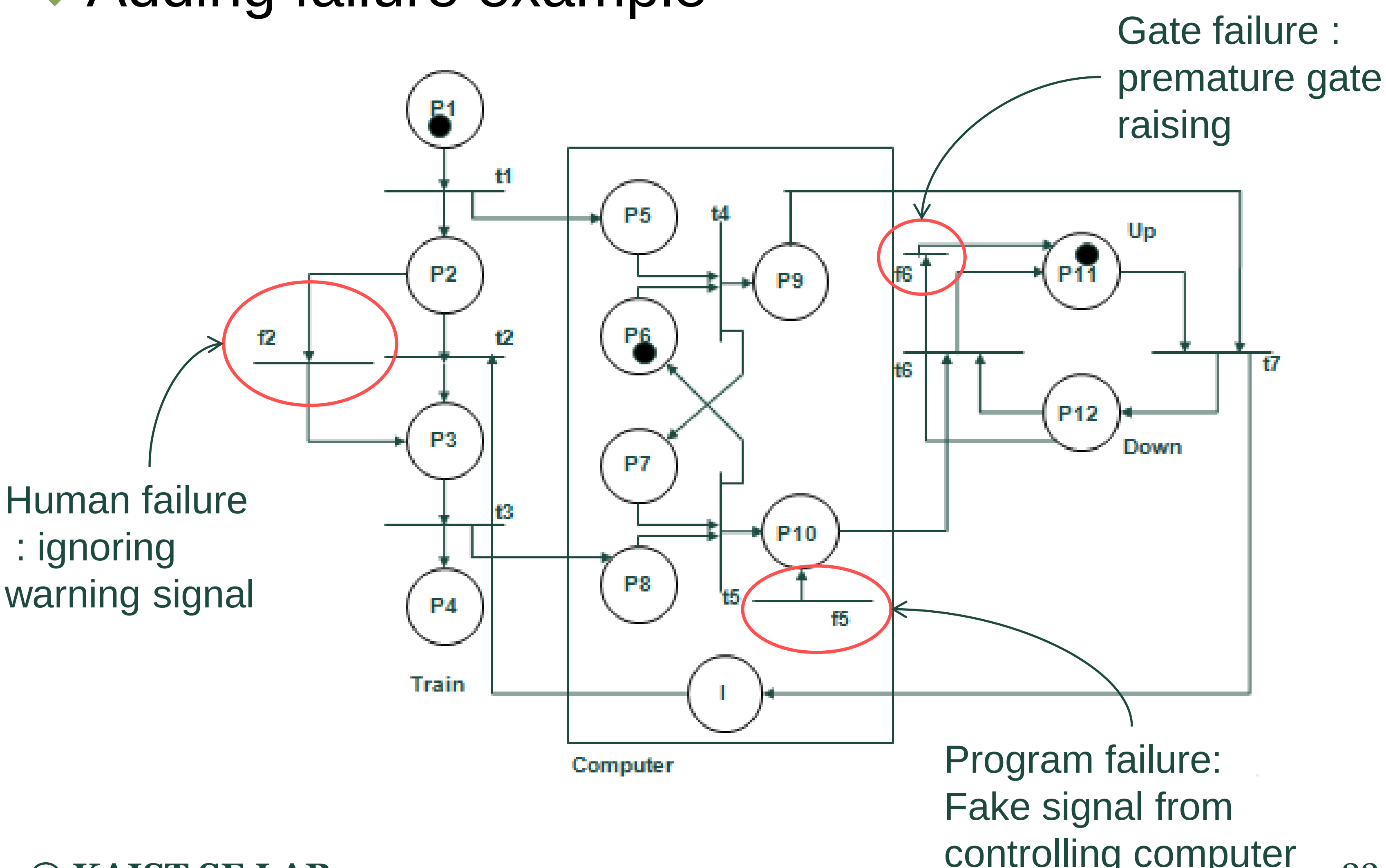
➢ Add fault-detection and recovery devices to minimize the risk of a mishap (fault-tolerant)

➢ If risk can not be lowered, (e.g., unacceptable probability it fails or uncontrollable variables such as human error involved)

➢Add hazard-detection and risk-minimization mechanisms (fail-safe)

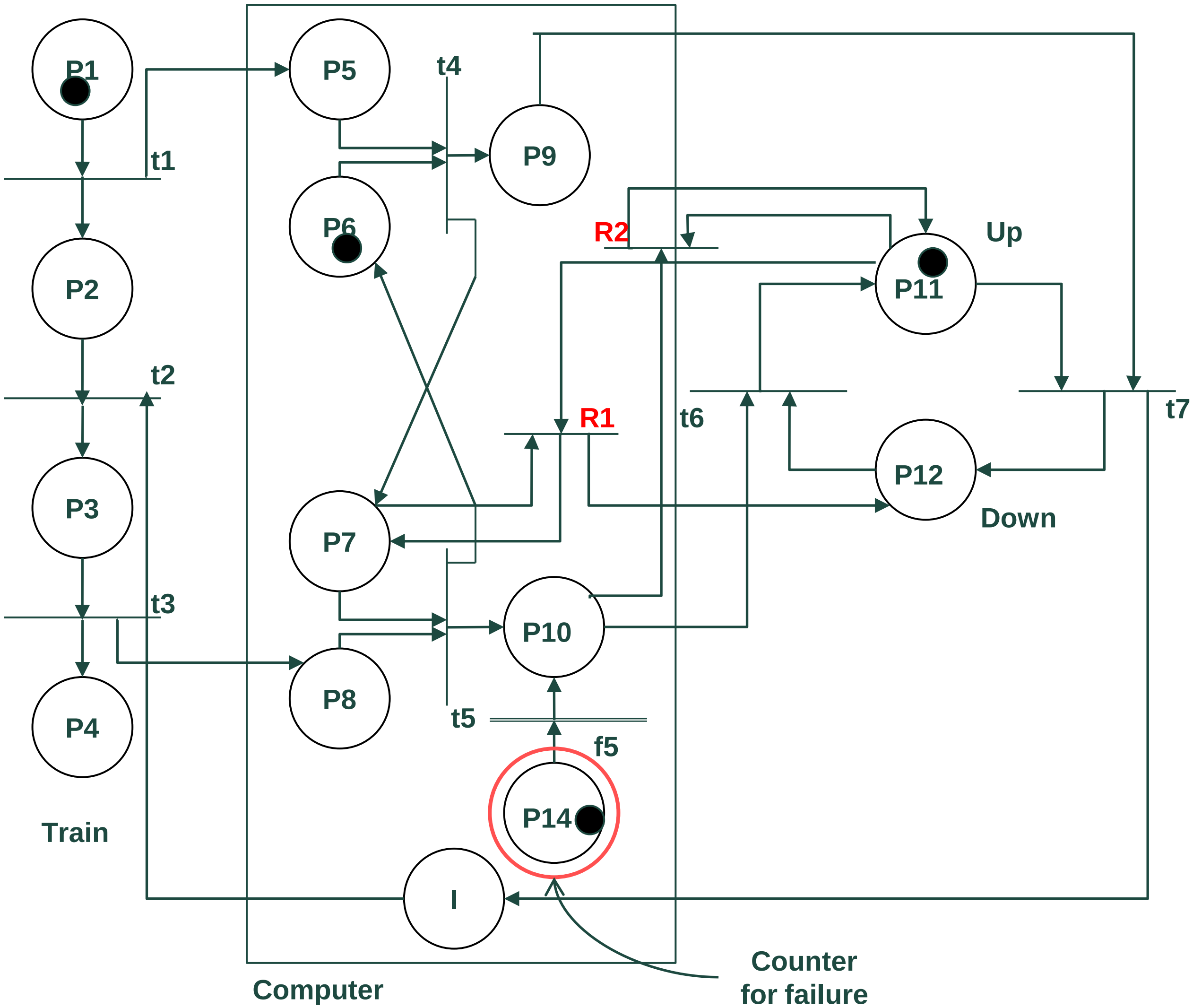
Example of safety analysis (2/3)

❖ Adding failure example



Example of safety analysis (3/3)

- Failure analysis example with recovery transition



R1 : lower gate when it should be down

R2 : ignore spurious control signal

Conclusion

- Contribution
  - Suggest ‘critical state’ algorithm eliminating high-risk states without generating whole reachability graph
  - Suggest model to analysis failure using Petri net

- Future work
  - Considering probability of hazard occurring not only its severity
  - Verifying formally whether the algorithm really generate high-risk free design

Discussion

- Limitation
  - Because of the time, the meaning of each words are little bit different
  - In the failure analysis, how to represent of time-associated failure is not suggested
  - There is no example of fail-safe mechanism
  - Lack of formal verification

Thank You

Q & A

KAIST SE LAB
KAIST Software Engineering Laboratory

About author

- She was a computer science professor of UC Irvine, University of Washington
- Now she is professor of MIT
- Authority on software safety(safety critical real time system)
- [safe ware : System safety and computers] is published 1995

Definition of terms

- Failure
  - Nonperformance or inability of the system or component to perform its intended function for a specified time under specified environmental conditions
- Accident
  - An undesired and unplanned event that result in a specified level of loss
- Hazard
  - A state or set of conditions of a system that will lead inevitably to an accident(loss event)

From Safeware(1995, NG. Leveson)

Recoverability

- Recoverability
  - Formal definition
    - Number of states are finite

$$cardinality(\sum_F) < \infty$$

    - There are no terminal faulty node

$$for \forall \sigma \in \sum_F, \exists t \in T \; such \, that \; \delta(\sigma, t_i) = \sigma'$$

    - There are no directed loops including *only* faulty states

$$\neg\exists \; sequence \; t_1 ... t_n \; such \; that \;\; for \;\; \sigma_i \in \sum_F,$$

$$\delta(\sigma_i, t_i) = \sigma_{i+1} \;\; for \;\; i = 1..n-1 \, and \;\; \sigma_1 = \sigma_{n+1}$$

    - The sum of maximum times on all paths *from the failure transition to correct state* is less than a predefined acceptable amount of time

$$for \; \forall path \; (t_1 ... t_n) \;\; from \; \sigma_1 \in \sum_F \; to \; \sigma_2 \in \sum_L$$

$$\sum Max(t_j) < T_{acceptable} \;\; for \;\; j = 1..n$$