Limitations of Liveness in Concurrent Software Systems

Marian V. Iordache
School of Engineering and Eng. Tech.
LeTourneau University
Longview, TX 75607-7001

Panos J. Antsaklis
Department of Electrical Engineering
University of Notre Dame
Notre Dame, IN 46556

December 16, 2010

Topic

We will say that a software system is **responsive** if for any reachable state any desirable transition (action) can eventually take place.

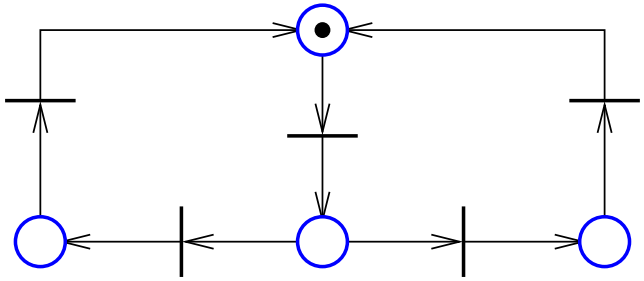
What properties of DES models of software guarantee that the software is responsive?

The paper shows that in general liveness does not guarantee responsiveness.

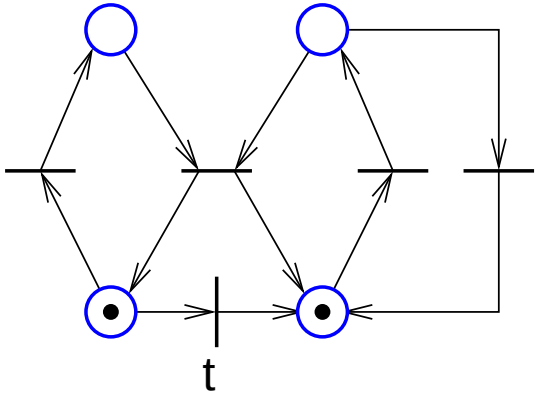
The paper also includes sufficient conditions under which liveness implies responsiveness.

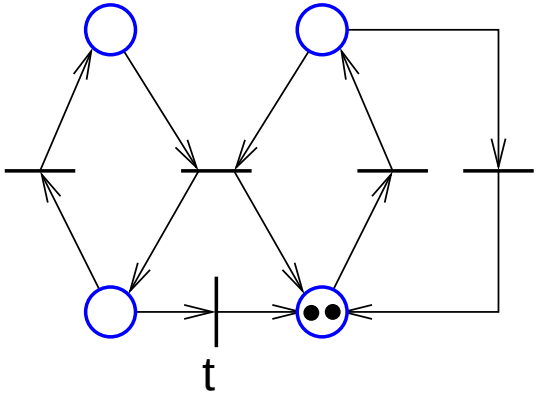
Liveness

A PN is live if for any reachable state, any transition can eventually take place.



LIVE





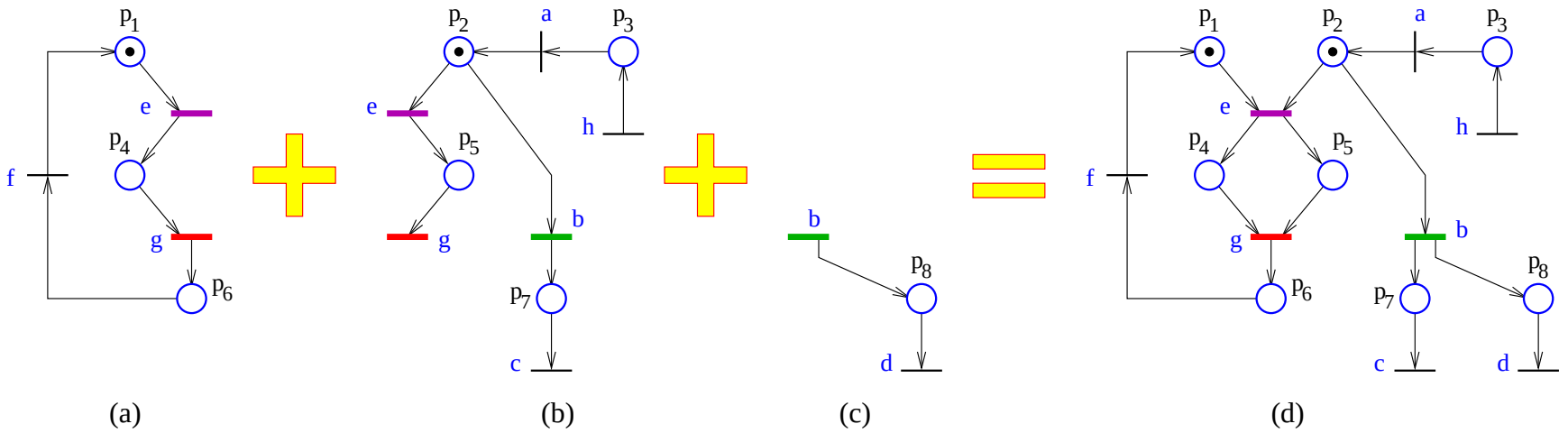
NOT LIVE

Software Model

We consider multithreaded software in which the thread structure is modeled by *state machines*.

Each state of a state machine represents an execution stage.

Due to synchronization between threads the overall model of the program is a PN.



A thread is represented by a token in its corresponding state machine.

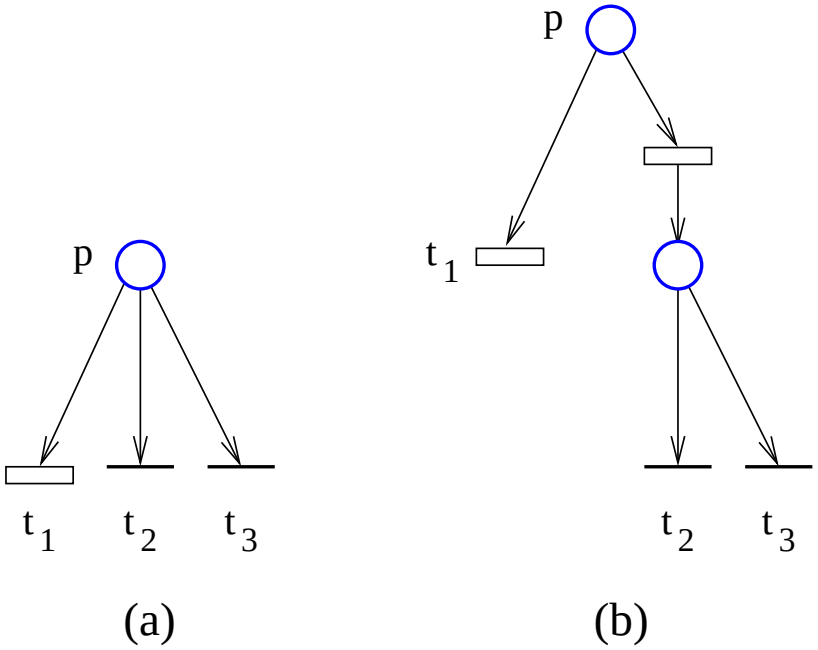
Each place corresponds to a stage in the execution of a thread.

When there is a choice for the next transition:

- If the code associated with a place selects the next transition, the choice is **deterministic**.
- If the thread imposes no constraints on the next transition, the choice is **nondeterministic**.

Deterministic choice cannot be revoked.

Without loss of generality, it is assumed that at every place choice is either completely deterministic or completely nondeterministic.



An arc (p, t) is **deterministic** (**nondeterministic**) if choice at p is deterministic (nondeterministic).

A hybrid PN (HPN) is used to represent software systems.

A HPN is a PN in which each place is associated with a segment of code.

A transition is enabled if enabled by both the underlying PN and the code associated with its input places.

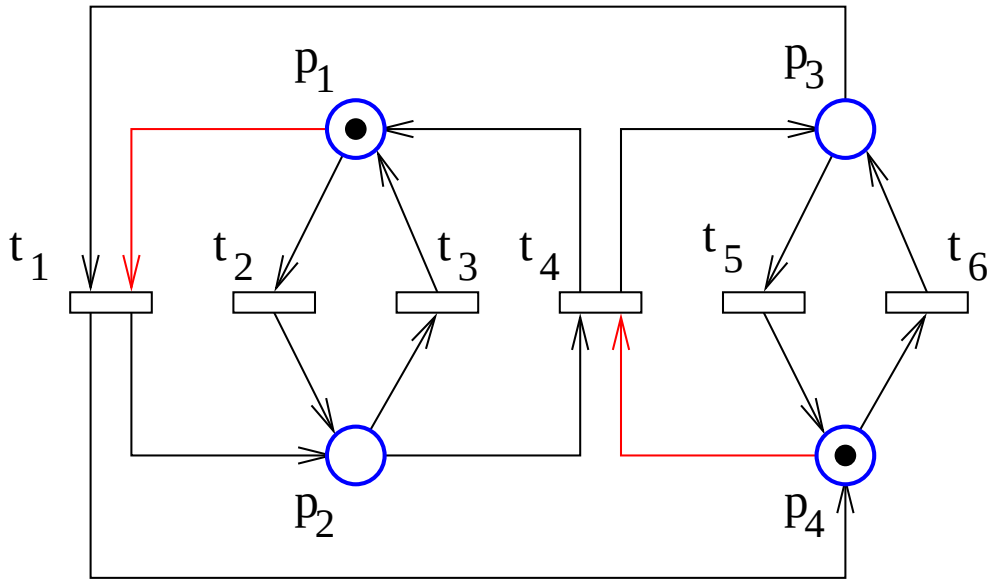
Enabled transitions are fired immediately.

A state of the HPN consists of the marking of the underlying PN and the state of the program.

We consider a restricted form of responsiveness defined as follows:

An HPN is responsive *if from any reachable state any transition can be eventually fired.*

Liveness Does not Imply Responsiveness



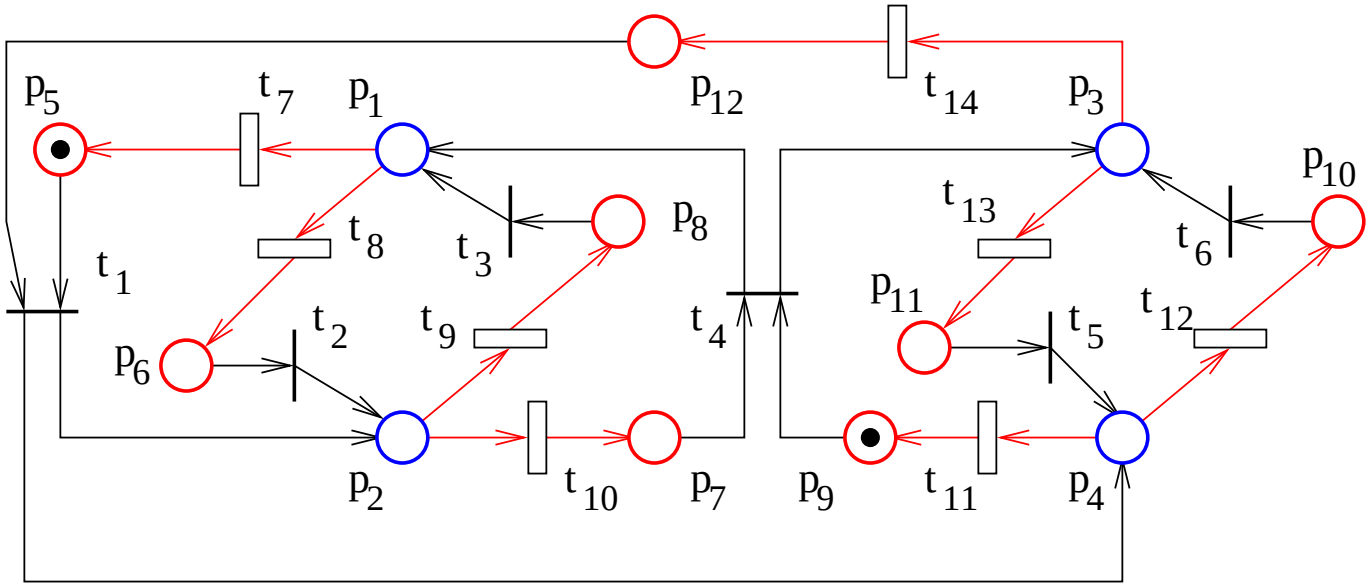
All choice is deterministic.

The choice is highlighted in red at each marked place.

The HPN is deadlocked though the PN is live.

Liveness Does not Imply Responsiveness

To fix this problem deterministic choice has to be modeled explicitly.

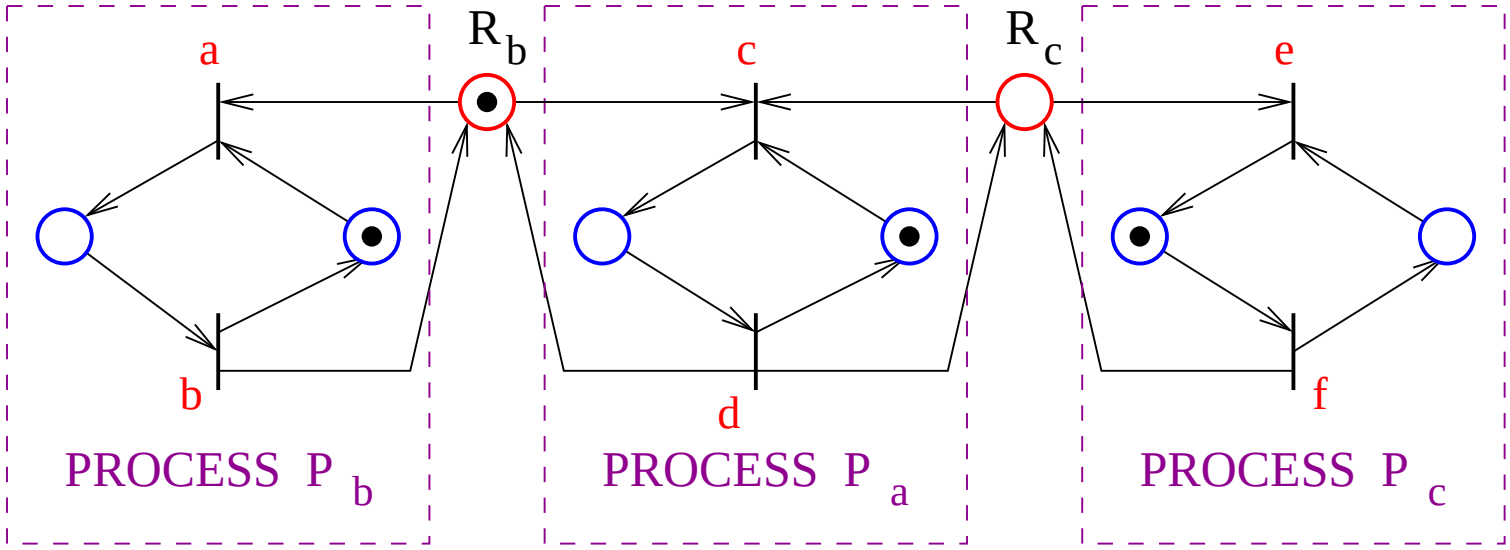


Changes are highlighted in red.

This time the PN is not live.

Liveness Does not Imply Responsiveness

Event timing may also create a problem.

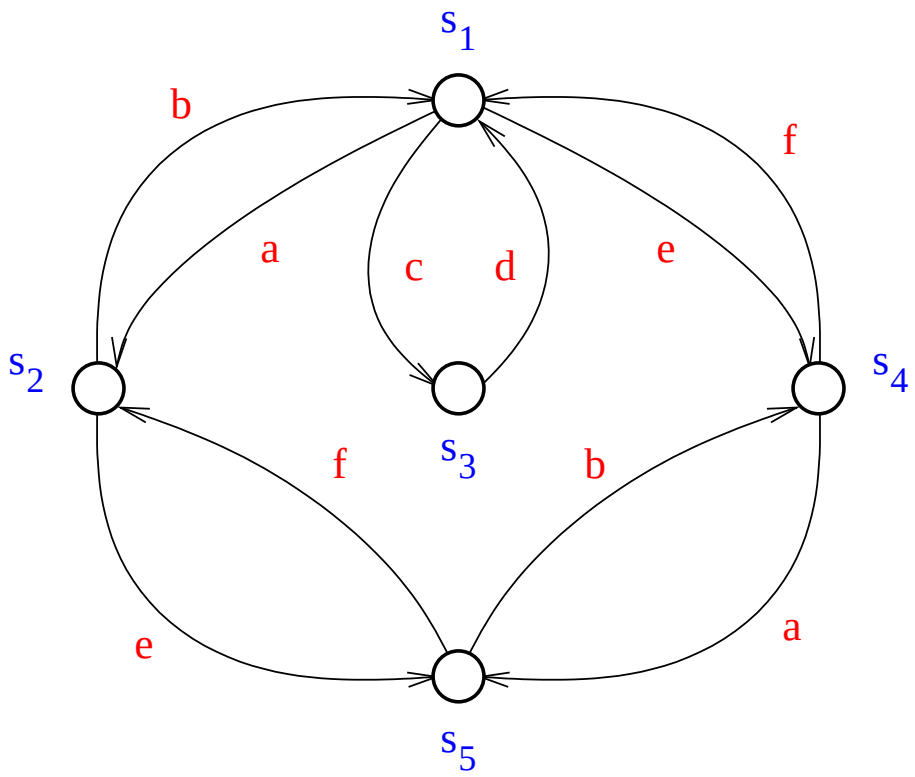


The PN is live but the HPN may not be responsive.

Indeed, events may occur in an order such that c is never enabled.

Liveness Does not Imply Responsiveness

Here is the equivalent automaton (the reachability graph).



Events may occur in an order such that s_1 is never reached.

Assumption 1 For every place p that outputs deterministic transitions, for every transition $t \in p\bullet$, the number of consecutive choices at p that do not select t is finite.

Assumption 2 The code associated with any place p is executed in finite time.

Assumption 3 Places with nondeterministic choice have "asymmetric choice". That is, for any two nondeterministic arcs (p_1, t) and (p_2, t) connected to the same transition t, either $p_1\bullet \subseteq p_2\bullet$ or $p_1\bullet \supseteq p_2\bullet$.

Main Results

An HPN is said to be **PT-ordinary** if all arcs from places to transitions have unity weight.

A PN $\mathcal{N}'$ is the **normalized** version of a PN $\mathcal{N}$ if it represents explicitly deterministic choice.

Theorem. *Consider a PT-ordinary HPN. Under the assumptions 1–3, if there is a reachable state from which a transition can never be fired, then the normalized PN is not live.*

Main Results

Normalization can also be defined for the closed-loop of a supervisor with the software system.

Corollary. *A supervisor enforces responsiveness if all of the following conditions are met.*

1. *The plant satisfies the assumptions 1 and 2.*
2. *The supervisor can be represented by a PN.*
3. *The normalized closed-loop PN is live.*
4. *The normalized closed-loop PN is PT-ordinary and satisfies the assumption 3.*

Final Remarks

In general, liveness is not sufficient for software-system responsiveness.

There are sufficient conditions under which liveness can guarantee responsiveness.

A possible approach to the design of responsiveness enforcing supervisors is to design liveness enforcing supervisors that satisfy the sufficient conditions.

In particular, a possible approach for future work is to use fairness constraints to guarantee that liveness enforcement will ensure responsiveness.