The ACTS Software and its Supervisory Control Framework

Marian V. Iordache
Department of Engineering
LeTourneau University
Longview, TX 75607-7001

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

December 13, 2012

Multiprogramming involves multiple tasks executed concurrently.

- Tasks may have to synchronize.
- Tasks may need to get exclusive access to shared resources.
- Tasks may not have the same precedence.
- Wait time should be bounded.

- Represent concurrency constraints in the supervisory contol framework.
- Solve the supervisory control problem.
- Implement supervisory policy in concurrency control code.

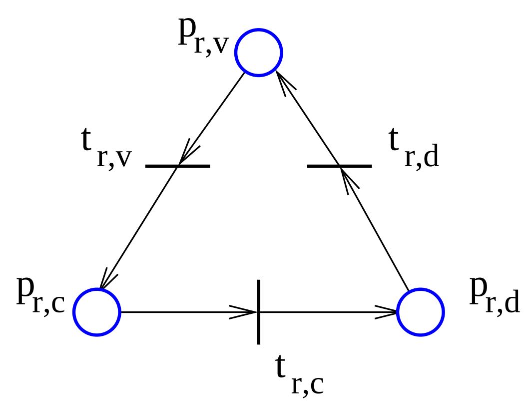
- ACTS: generates concurrency control code for concurrent programs.
- PN based supervisory control is implemented.

Synchronization problem (adapted from [Downey, 2008]):

- Shared data accessed by reader, inserter, and deleter processes.
- At any time, only one inserter may modify the data.
- At any time, only one deleter may modify the data.
- Readers and inserters may not access the shared data at the same time as deleters.

- Concurrent entities have a DES structure.
- A place represents an execution stage of the entity.
- Each place is associated with a segment of code.
- Concurrent entities with the same code correspond to different tokens of the same PN.

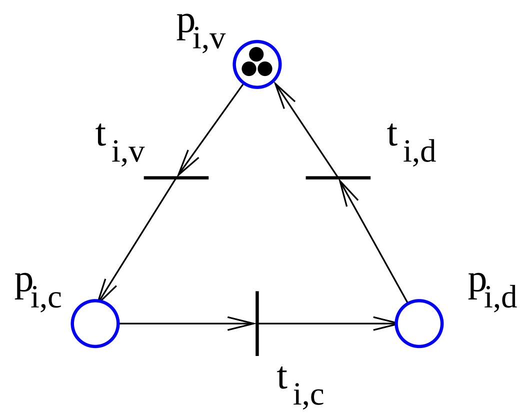
```
thread READER {
  places:  pv pc pd;
  transitions:  tv tc td;

  (pv, tv, pc); (pc, tc, pd);
  (pd, td, pv);
}
```



READER

- A DES is associated with a group of entities.
- The number of tokens equals the number of entities in a group.
- A DES does not have to preserve the number of tokens.

```
   // create identical DES for inserter entities
1. INSERTER = READER;
   // create identical DES for deleter entities
2. DELETER = READER;
   // define initial markings
3. initialize:  READER(pv:4,pd:1);
4. initialize:  INSERTER(pv:3);
5. initialize:  DELETER(pv:3);
```



INSERTER

The constraints are described by linear inequalities.

- Only one inserter may be in the critical section.

$$\mu_{i,c} \leq 1 \tag{1}$$

- Only one deleter may be in the critical section.
- Readers and inserters may not be in the critical section at the same time as a deleter.

$$6\mu_{d,c} + \mu_{r,c} + \mu_{i,c} \leq 6 \tag{2}$$

- A deleter should not wait indefinitely to access the critical section.
- Bounded wait requires more than just inequalities ...

```
   // Inequality constraints
1. INSERTER.pc <= 1
2. 6*DELETER.pc + READER.pc + INSERTER.pc <= 6
```

Specification — Example

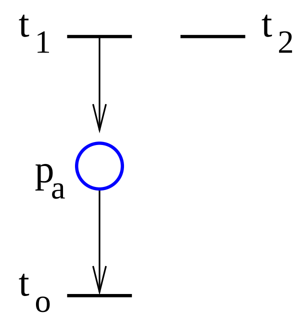
For bounded wait:

- define supervisor component
- synchronize t_1 and t_2 with $t_{r,v}$.
- require

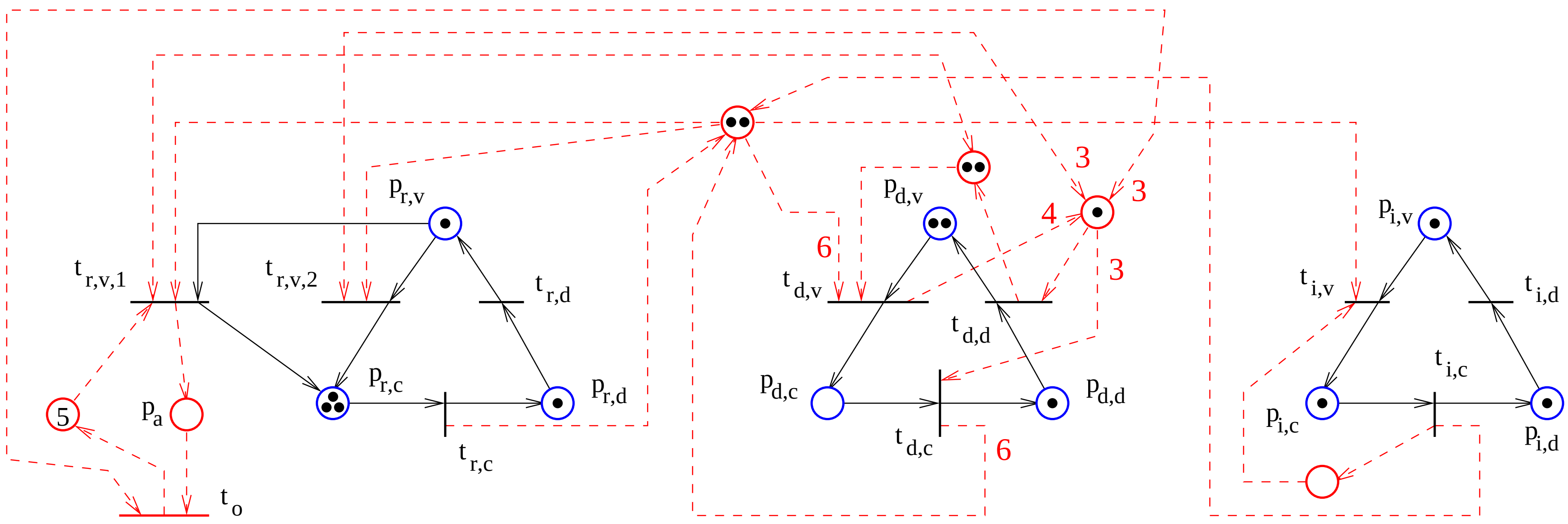
$$q_1 \leq \mu_{d,v} \quad (3)$$

$$\mu_a \leq 5 \quad (4)$$

$$3q_2 + 3q_o \leq 3 - \mu_{d,v} + 3\mu_{d,c} \quad (5)$$




```
supervisor sc { // Defines supervisor component
  places:  pa;
  transitions:  t0 t1 t2;
  (t1, pa); (pa, t0); }
sync sc.t1 sc.t2 READER.tv // Synchronizes transitions
sc.q.t1 <= DELETER.pv
sc.pa <= 5
3*sc.q.t2 + 3*sc.q.t0 <= 3 - DELETER.pv + 3*DELETER.pc
```


$p_{r,v}$
$t_{r,v,1}$
$t_{r,v,2}$
$t_{r,d}$
$p_{r,c}$
$p_{r,d}$
$t_{r,c}$
5
p_a
t_o
$p_{d,v}$
$t_{d,v}$
$t_{d,d}$
$p_{d,c}$
$p_{d,d}$
$t_{d,c}$
6
6
3
3
4
3
$p_{i,v}$
$t_{i,v}$
$t_{i,d}$
$t_{i,c}$
$p_{i,c}$
$p_{i,d}$

Properties

Note: PN $+$ user code $=$ HPN

A supervisory policy enforcing inequality constraints on the underlying PN will enforce the constraints also when applied to the HPN.

Assuming no uncontrollable transitions and no unobservable transitions, a least restrictive supervisory policy enforcing inequality constraints on the underlying PN will be least restrictive also when applied to the HPN.

Properties

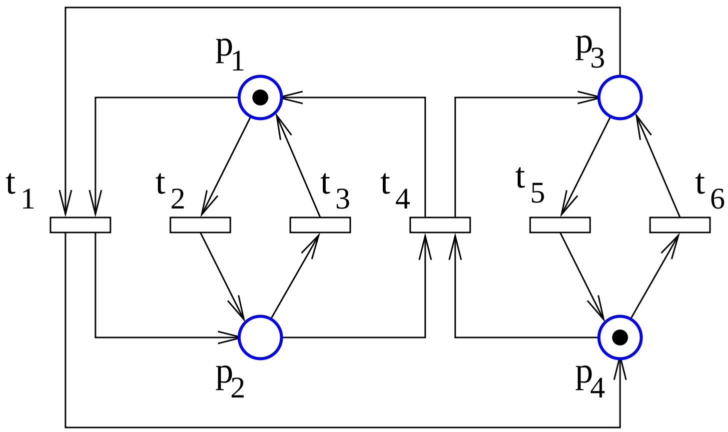
Uncontrollable and/or unobservable transitions may arise, for instance:

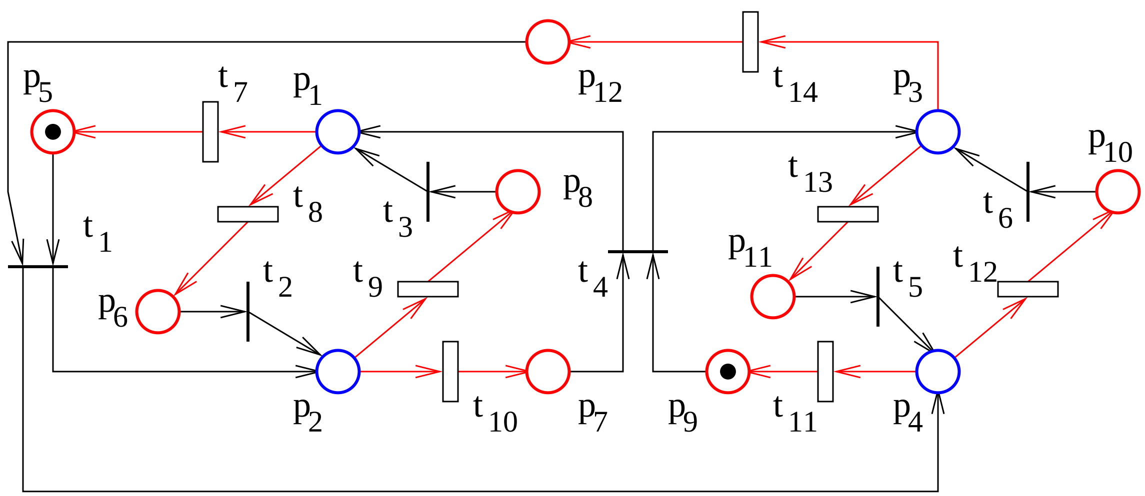
- as critical transitions that should not be delayed;
- in a decentralized context.

If the HPN has uncontrollable and/or unobservable transitions, a least restrictive supervisory policy enforcing inequality constraints on the underlying PN may not be least restrictive when applied to the HPN.

Properties

A PN model is said to be **normal** if it represents explicitly choice.





Properties

The bounded wait property is related to liveness.

If the underlying PN of an HPN does not represent explicitly choice, a supervisory policy preventing deadlock or enforcing T-liveness in the underlying PN may not prevent deadlock in the HPN.

Consider a HPN in which the underlying PN represents explicitly choice. A supervisory policy preventing deadlock in the underlying PN will prevent deadlock also in the HPN.

In general, additional conditions are required to guarantee bounded wait.

Liveness $<$ Responsiveness $<$ With bounded wait

Conclusions

SC can be applied to concurrency control.

ACTS: free open-source software applying SC to concurrent programming.

In general, traditional SC methods may be suboptimal in the context of uncontrollable and/or unobservable transitions.

Excepting special cases, traditional SC methods for liveness enforcement are insufficient.