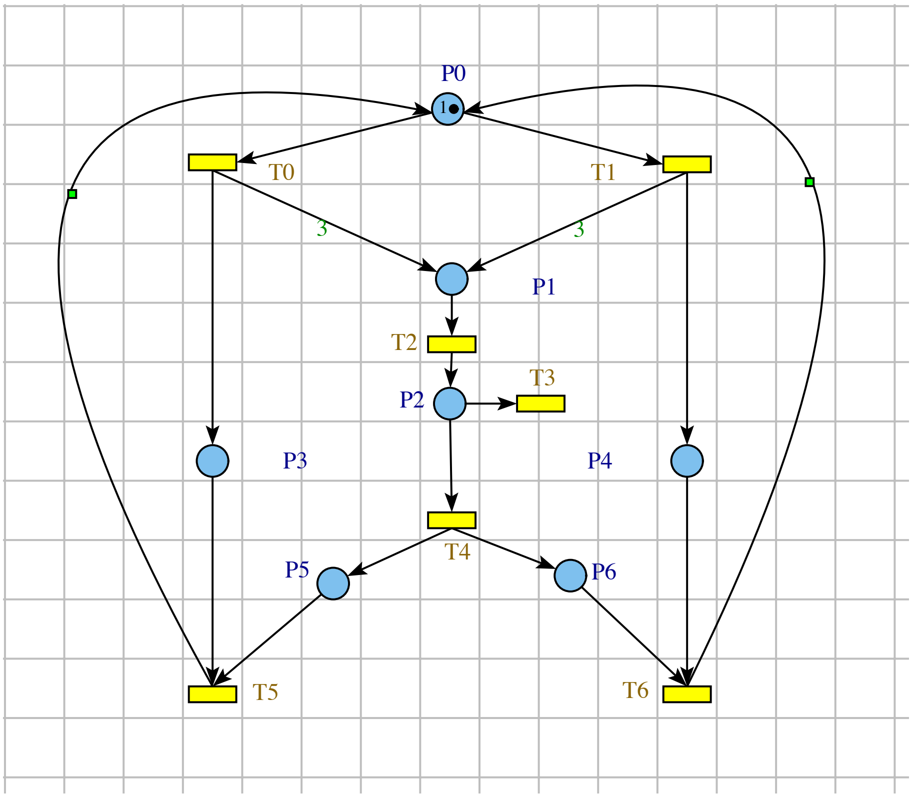
Examination of Petri Nets - Master ARIA

David Delfieu, 1h30, any documents

2015

1 Analysis of Petri nets (10 points)



1. State all the properties of this Petri net
2. Find the conservative components of this networ

Solution

Properties

- Event Graph : No (P_0), State Graph : No (T_5).
- Conflict (strutural and effective) in P_0, Free Choice and Simple Choice.
- Pure, without loops.
- Parallelism structural : T_{10}, T_5, ex effective parallelism T_2, T_3,T_4.
- Quasi-Alive ($T_0, T_2, T_2, T_2, T_3, T_3, T_3$) : Not alive, Not re-initialisable, Unbounded.

Place invariant

$$\left|\begin{array}{c} \text{-1 -1 0 0 0 1 1 } f_1 \\ \text{3 3 -1 0 0 0 0 } f_2 \\ \text{0 0 1 -1 -1 0 0 } f_3 \\ \text{1 0 0 0 0 -1 0 } f_4 \\ \text{0 1 0 0 0 0 -1 } f_5 \\ \text{0 0 0 0 1 -1 0 } f_6 \\ \text{0 0 0 0 1 0 -1 } f_7 \end{array}\right|$$

Table 1 – Incidence Matrix C : Pivot = C(3,4)

$$\left|\begin{array}{c} \text{-1 -1 0 0 1 1 } f_1 \\ \text{3 3 -1 0 0 0 } f_2 \\ \text{1 0 0 0 -1 0 } f_4 \\ \text{0 1 0 0 0 -1 } f_5 \\ \text{0 0 0 1 -1 0 } f_6 \\ \text{0 0 0 1 0 -1 } f_7 \end{array}\right|$$

Table 2 – C_1 : Pivot = $C_1(2,3)$

$$M(P_0) + M(P_3) + M(P_4) = M_0(P_0) + M_0(P_3) + M_0(P_4)$$

$$\left|\begin{array}{l} \text{-1 -1 0 1 1 } f_1 \\ \text{1 0 0 -1 0 } f_4 \\ \text{0 1 0 0 -1 } f_5 \\ \text{0 0 1 -1 0 } f_6 \\ \text{0 0 1 0 -1 } f_7 \end{array}\right|$$

TABLE 3 – C_2 : Pivot $= C_2(2,1)$

$$\left|\begin{array}{l} \text{-1 0 0 1 } f_1 + f_4 \\ \text{1 0 0 -1 } f_5 \\ \text{0 1 -1 0 } f_6 \\ \text{0 1 0 -1 } f_7 \end{array}\right|$$

TABLE 4 – C_3 : Pivot $= C_3(3,3)$

$$\left|\begin{array}{l} \text{-1 0 1 } f_1 + f_4 \\ \text{1 0 -1 } f_5 \\ \text{0 1 -1 } f_7 \end{array}\right|$$

TABLE 5 – C_4 : Pivot $= C_4(3,2)$

$$\left|\begin{array}{l} \text{-1 1 } f_1 + f_4 \\ \text{1 -1 } f_5 \end{array}\right|$$

TABLE 6 – C_5 : Pivot $= C_5(2,1)$

2 Racket(10 points)

2.1 Exercises (4 pts)

```
(define l1 '(a b c d e f))
(define l2 '(b a g h i  j))
```

1. Write the *inter* function that return the intersection of two sets :

```
(Inter l1 l2) = '(a b)
```

2. Write the *Union* function that return the union of two sets :

```
(Union l1 l2) = '(a b c d e f g h i j)
```

3. Write the *comp* function that return the elements of set l_1 which does not belong to set l_2 :

```
(comp l1 l2)='(c d e f)
```

Solution

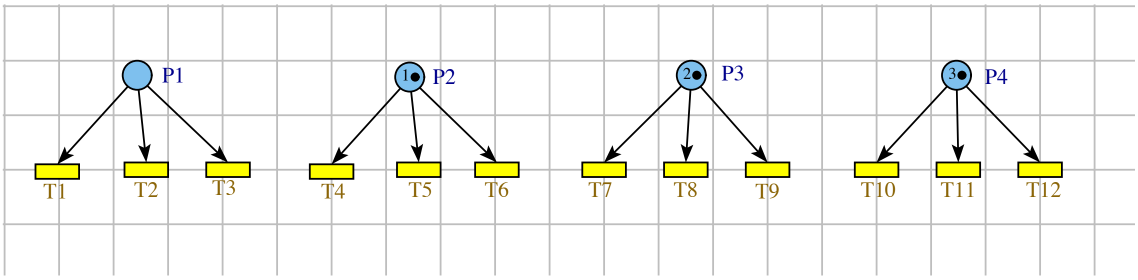
```
(define l1 '(a b c d e f))
(define l2 '(a b g h i j ))
(define (inter l1 l2)
  (for/list ([e1 l1] #:when (member e1 l2)) e1))

(define (union l1 l2)
  (cond
    [(empty? l1) l2]
    [(member (car l1) l2) (union (cdr l1) l2)]
    [else (cons (car l1) (union (cdr l1) l2))]))

(define (comp l1 l2)
  (cond
    [(empty? l2) l1]
    [else (for/list ([e l1] #:when (not (member e l2))) e)]))
```

2.2 Effective conflicts (6 pts)

1. For these four conflicts, detail the possible execution scenarios. What are the cases where a conflict is effective ?
2. From these 4 cases, determine the cases where the conflicts are either partially effective or completely effective.



3. Give a formal definition of these two concepts.
4. Give the racket code of two functions that will both return a boolen result :

   (define (CEffective M c) ...)
   return #t : if the conflict is completely effective in the M marking.

   (define (PEffective M c) ...)
   return #t : if the conflict is partially effective in the M marking.

As an indication, I give you some indications and the different functions I used for coding :

- A conflict c is defined by : $c = < p, \{t_1, t_2, ..., t_n\} >$
- (mark p M) : M is a marking, p is a place of the Petri nets. M(p) gives the marking of the place p in the marking M.
- (for/sum (for-clause ...) body-or-break ... body)
  For/sum is an iterative addition function all the results of each body are added and the final result is returned.
  Example :
  `> (for/sum ([i '(1 2 3 4)]) i)`
  `10`
- W contains all the weights of the arcs and (readW (list p t)) gives the value of the arc (p t) in the Petri nets.

Solution

```
(define c1 '(P1 T1 T2 T3))
(define c2 '(P2 T4 T5 T6))
(define c3 '(P3 T7 T8 T9))
(define c4 '(P4 T10 T11 T12))

(define (CEffective M c)
  (cond
    [(empty? c) #t]
```

```
    [ else (>= (mark (car c) M) (for/sum ([t (cdr c)]) (readW (list (car c) t))))]))

(define (PEffective M c)
  (cond
    [(empty? c) #t]
    [ else  (for*/or ([t1 (cdr c)] [t2 (cdr c)]
                         #:when (not (equal? t1 t2)))
                         (>= (mark (car c) M)
                             (+ (readW (list (car c) t1))(readW (list (car c) t2)))))]))

> (CEffective M0 c1)
#f
> (CEffective M0 c2)
#f
> (CEffective M0 c3)
#f
> (CEffective M0 c4)
#t
> (PEffective M0 c1)
#f
> (PEffective M0 c2)
#f
> (PEffective M0 c3)
#t
>
```