Colored Petri nets

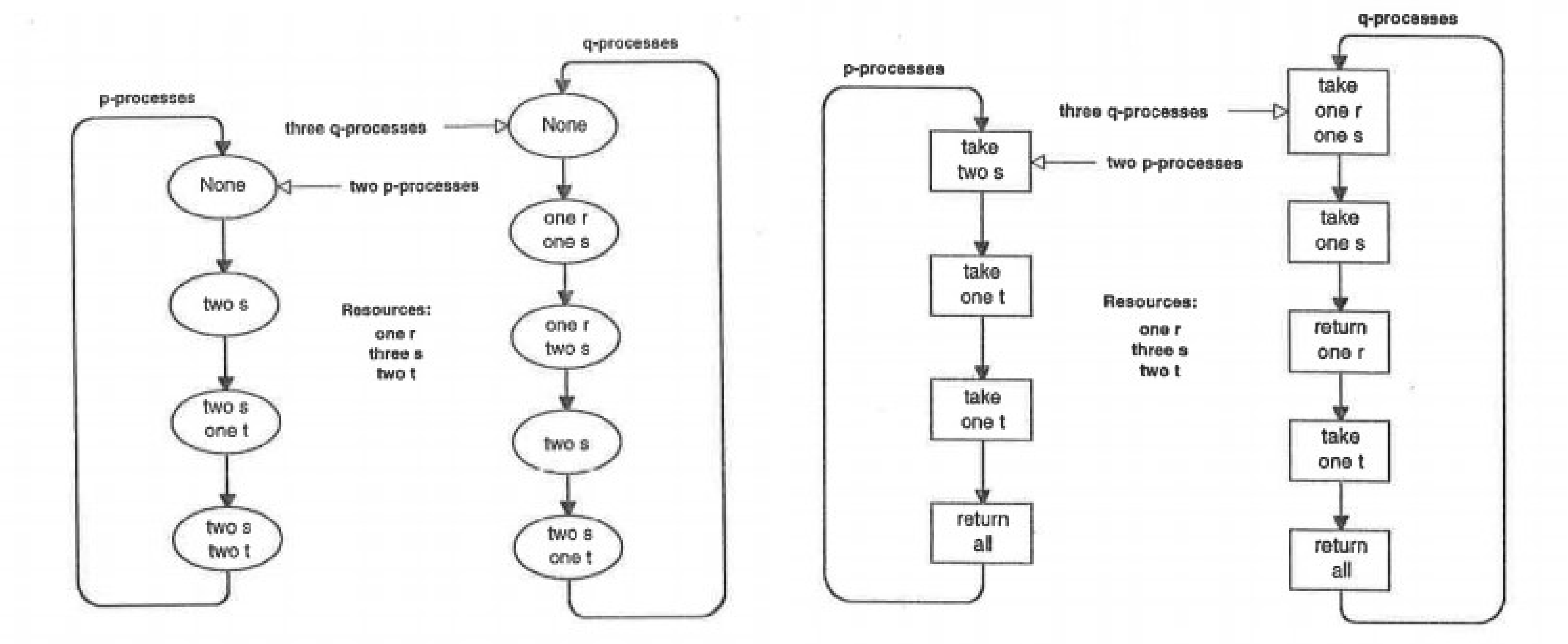
Miklós Gerzson

Introduction

- Petri nets: graphical and mathematical modelling tool for the description of dynamic systems
- system types: concurrent, asynchronous, distributed, parallel, nondeterministic, stochastic
- graphical representation: structural description and dynamic characterization
- mathematical description: state equations, algebraic equations
- analysis tool: behavioral and structural features of systems

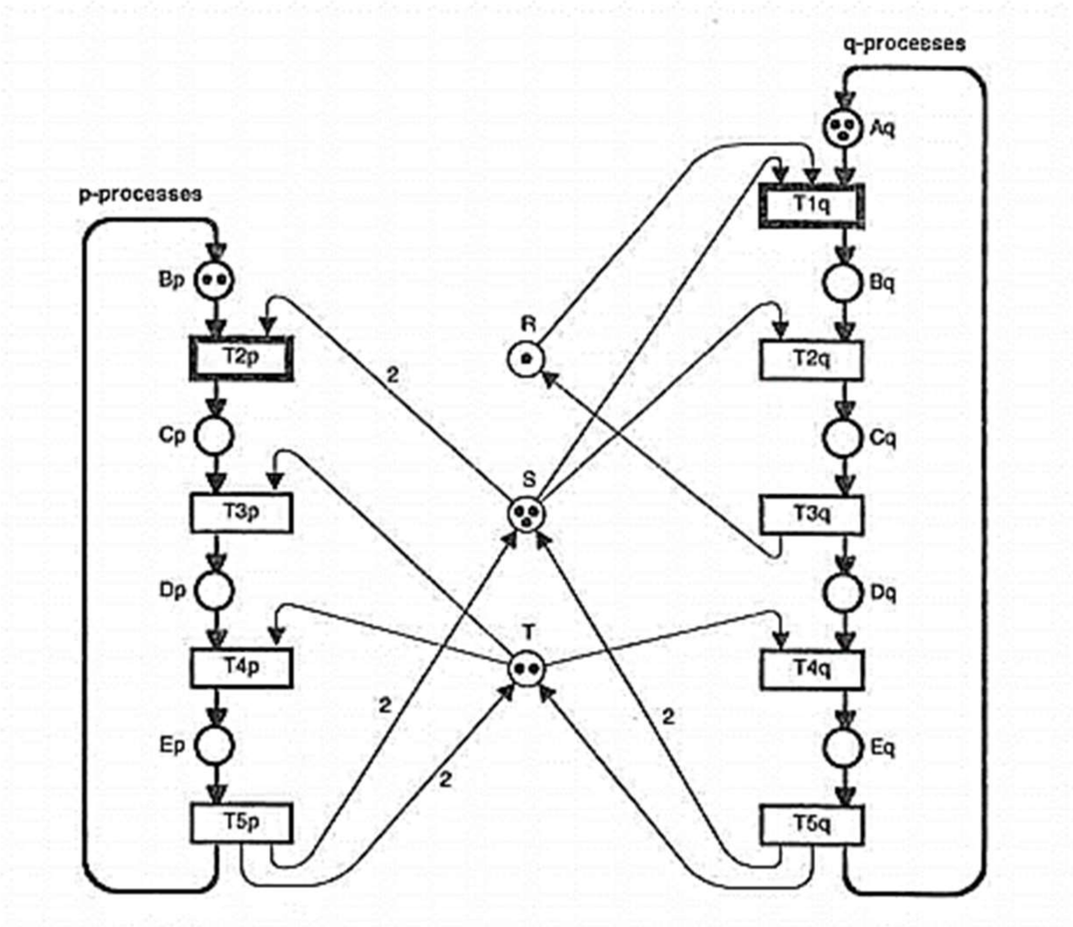
Introductory example

- Problem description



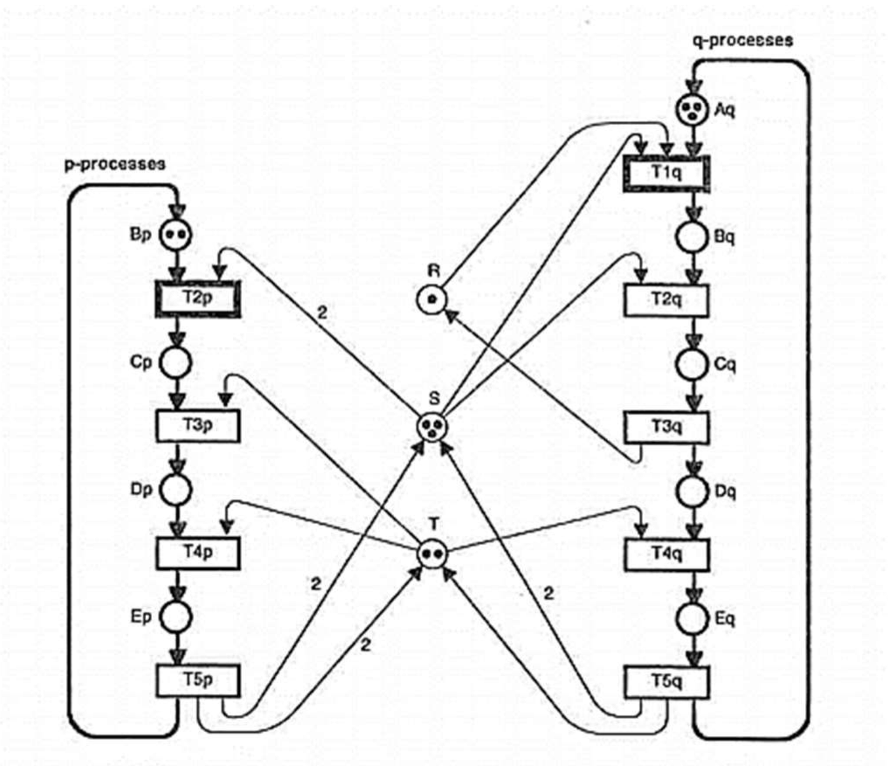
Introductory example

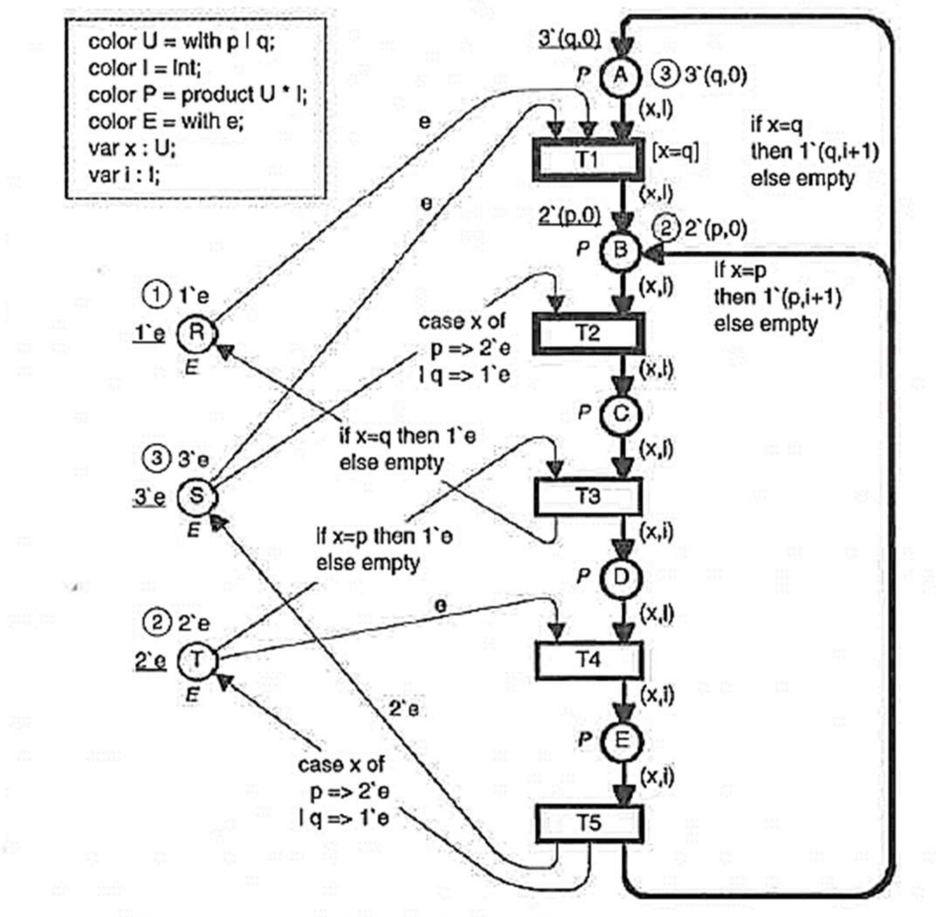
- Low level Petri net of example



Introductory example

- The first step to get the CPN model





Introductory example

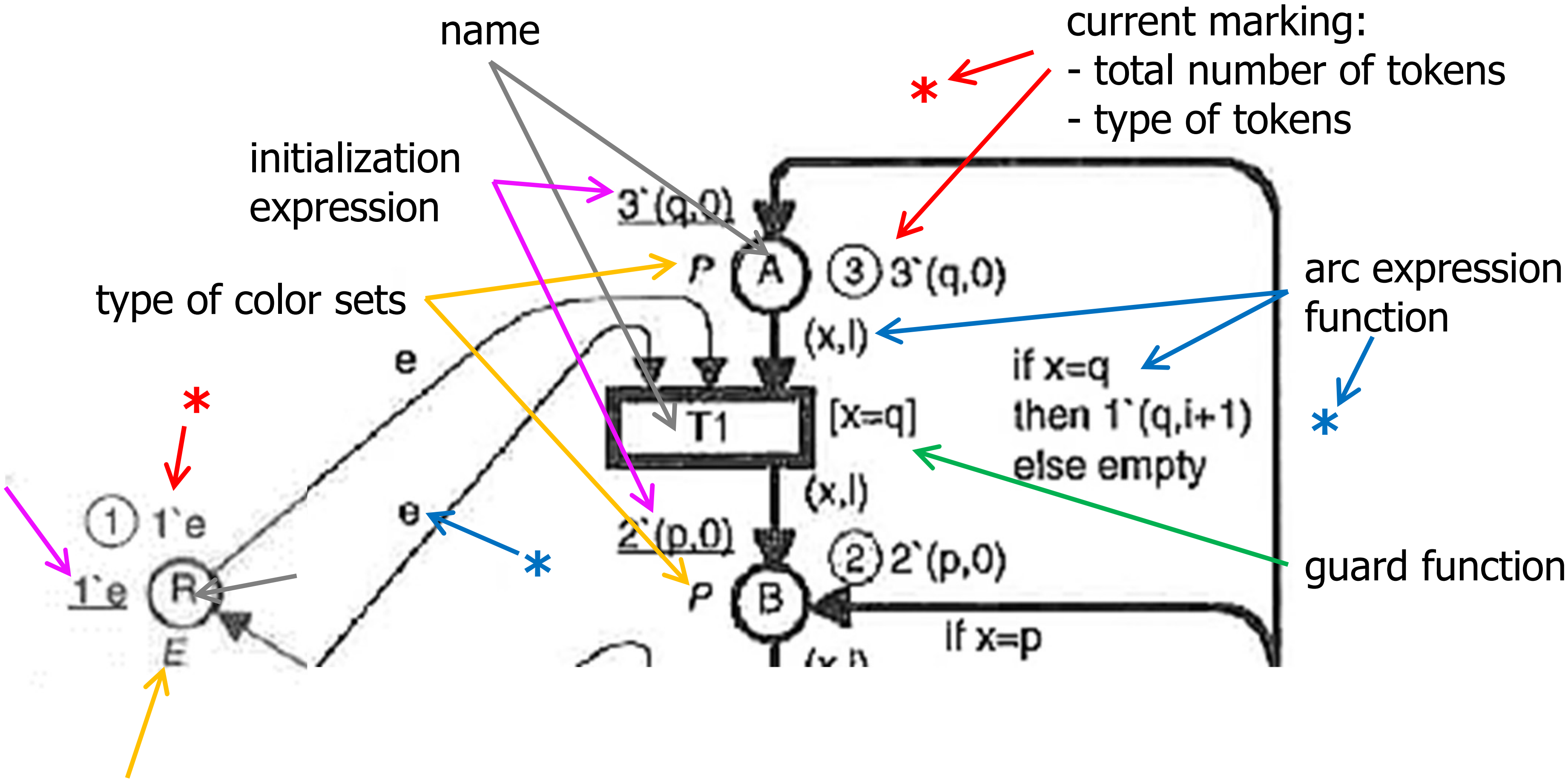
- Declaration
- sets:
  - *U* – type of processes *'p'* or *'q'*
  - *I* – number of cycles *'int'*
  - $P = U \times I$
  - *E* – resources

```
color U = with p | q;
color I = int;
color P = product U * I;
color E = with e;
var x : U;
var i : I;
```

- variables:
  - *x* – an element of set *U*
  - *i* – an element of set *I*

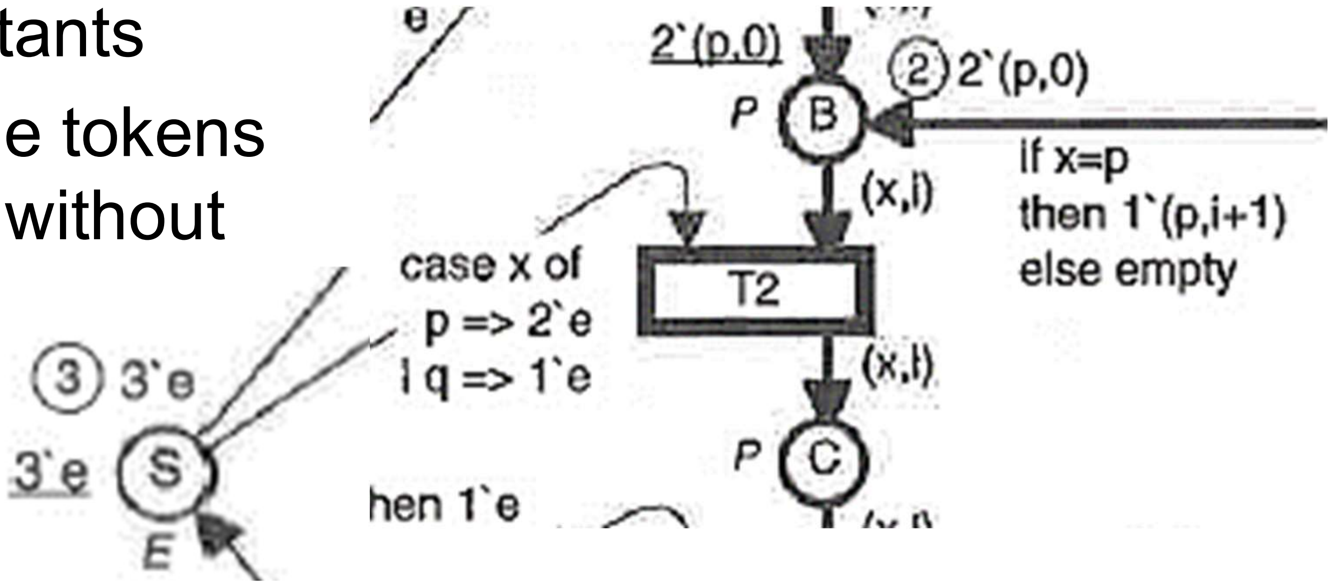
Introductory example

- Notation on CP-nets

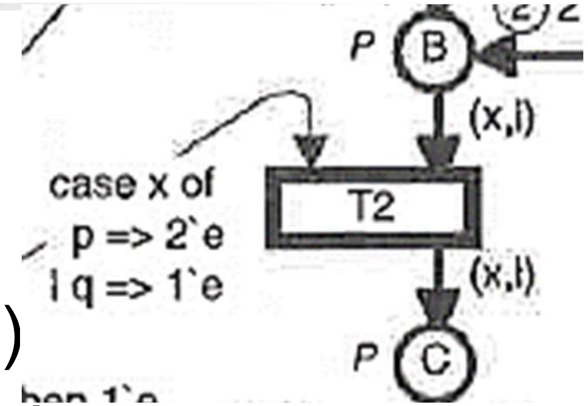


Introductory example

- occurrence of T_2:
  - x, i variables
  - p, q, e constants
  - T_2 moves the tokens from B to C without changing their color, and removes a multi-set of tokens from S



Introductory example



- binding of variables:
  - input arc expressions of T_2 :
    - (x, i) (variables) and e (constant)
    - variables have to be bound to colors
    - b_1=<x=p,i=0> or b_2=<x=q,i=10>
- enabling of a transition with a given binding:
  - b_1: the two input arc expressions evaluate to (p,0) and $e \rightarrow$ b_1 enabled
  - b_2: evaluate to (q,10) and 2\`$e \rightarrow$ b_1 enabled not enabled

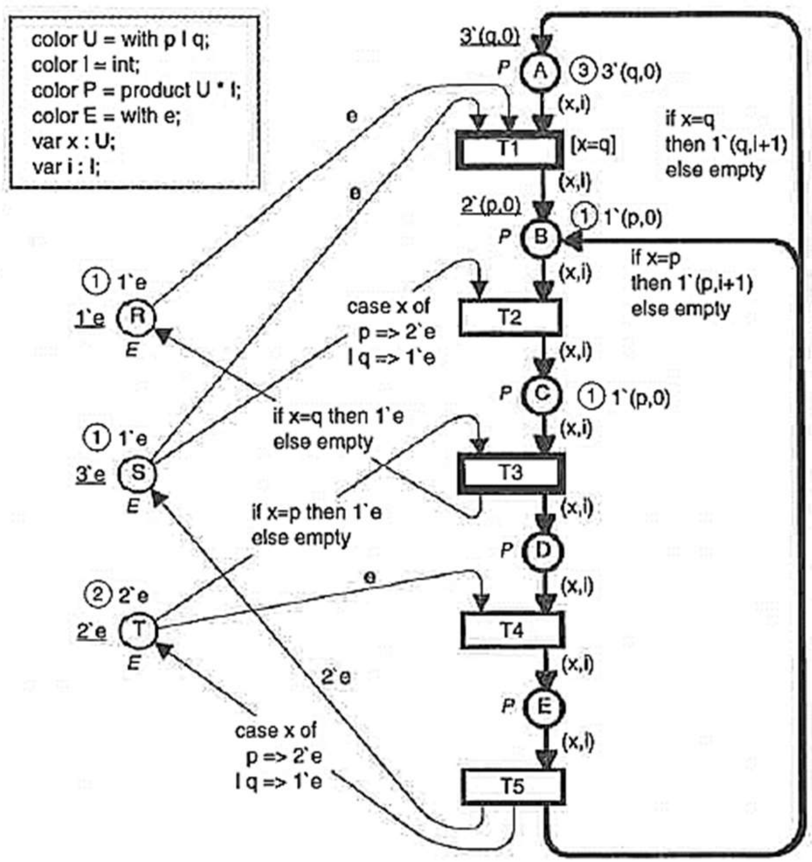
Introductory example

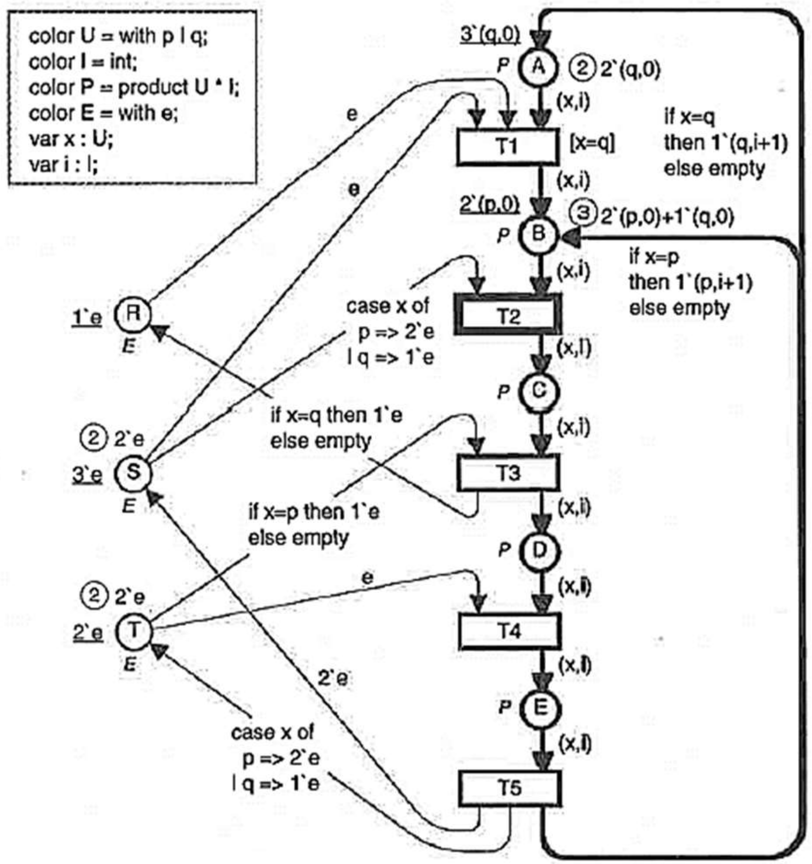
- enabled transition may occur
- binding element: (t, b)
  - (T_2, b_1)
- the binding element (T_2, b_1) is enabled in the initial marking M_0 and that it transforms M_0 into the marking M_1
- the binding element (T_1, b_2) where $(b_2 = \langle x=q, i=0 \rangle)$ is also enabled in the initial marking M_0 and that it transforms M_0 into the marking M_2

Introductory example

- Markings M_1 and M_2 are reachable from M_0

  by T_2,<x=p,i=0> by T_1,<x=q,i=0>



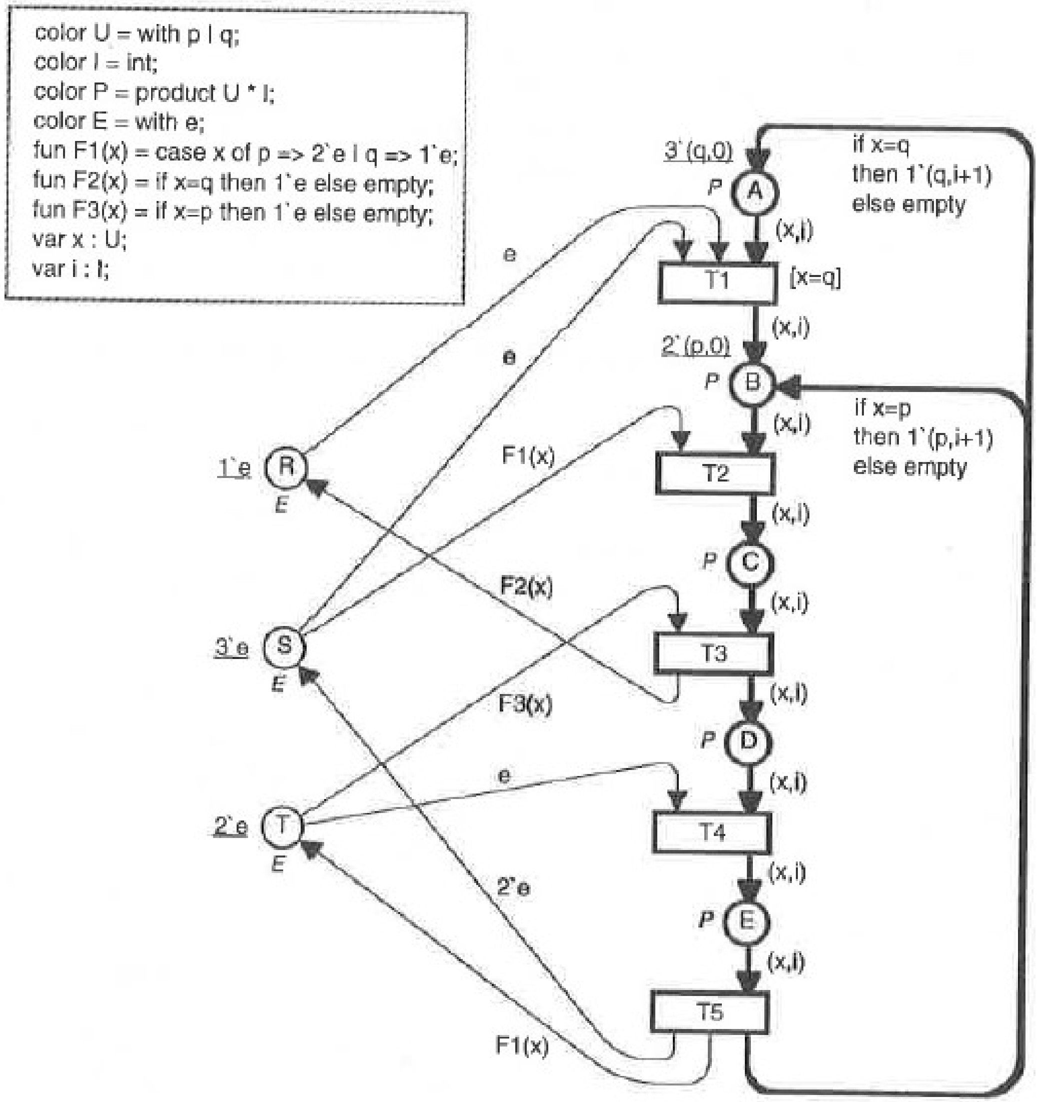


Introductory example

- CP-net consists of three different part:
  - the net structure:
    - places, transitions, arcs
  - declarations:
    - color *U*: with *p* | *q*;
    - var *x*: *U*;
  - net inscriptions
    - arc expressions
    - initialization expressions
    - guard functions
    - current marking,…

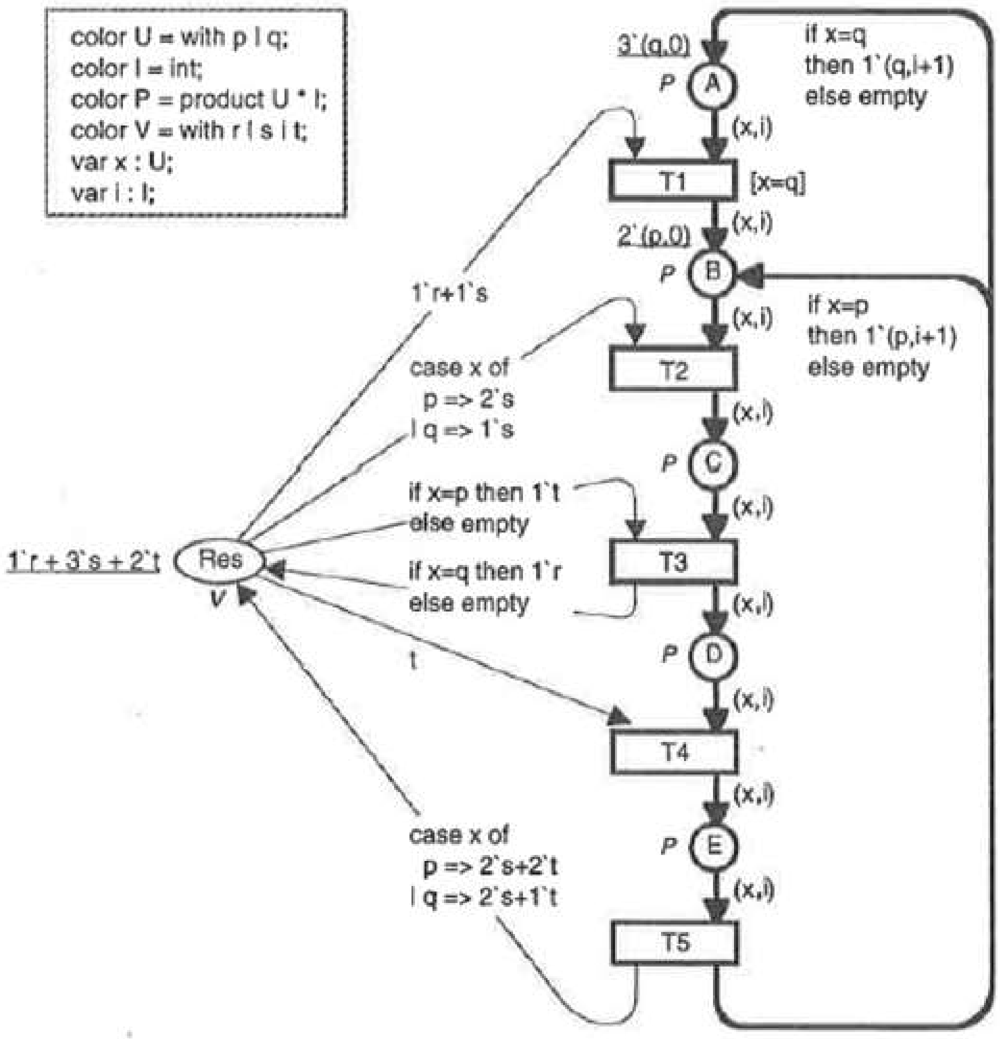
Introductory example

- introducing of functions instead of complex arc expressions



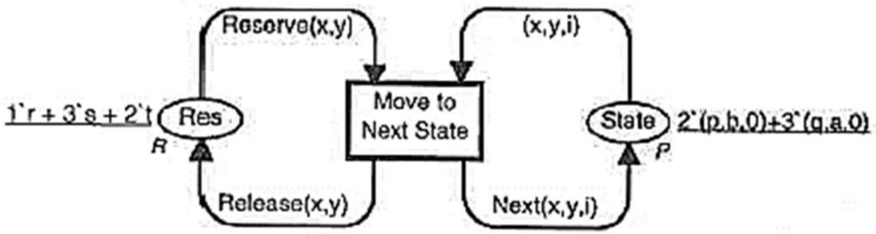
Introductory example

- introducing of one resource place having tokens with three different colors



Introductory example

```
color U = with p | q;
color S = with a | b | c | d | e;
color I = int;
color P = product U *S * I;
color R = with r | s | t;
fun Succ(y) = case y of a=>b |  b=>c |  c=>d |  d=>e |  e=>a;
fun Next(x,y,i) = (x, if (x,y) = (p,e) then b else Succ(y), if y=e then i+1 else i);
fun Reserve(x,y) = case (x,y) of (p,b)=>2`s | (p,c)=>1`t | (p,d)=>1`t
                                 | (q,a)=>1`r+1`s | (q,b)=>1`s | (q,d)=>1`t | _=>empty;
fun Release(x,y) = case (x,y) of (p,e)=>2`s+2`t | (q,c)=>1`r | (q,e)=>2`s+1`t | _=>empty;
var x : U;
var y : S;
var i : I;
```



color sets:

color U = with p | q; – processes

color S = with a | b | c | d | e; – states, steps of processes

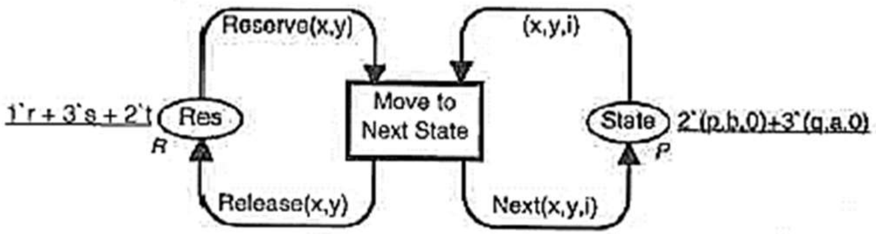
color I = int; number of cycles

color P = product U *S *I – $U \times S \times I$

color R = with r | s | t; – resources

Introductory example

```
color U = with p | q;
color S = with a | b | c | d | e;
color I = int;
color P = product U *S * I;
color R = with r | s | t;
fun Succ(y) = case y of a=>b |  b=>c |  c=>d |  d=>e |  e=>a;
fun Next(x,y,i) = (x, if (x,y) = (p,e) then b else Succ(y), if y=e then i+1 else i);
fun Reserve(x,y) = case (x,y) of (p,b)=>2`s | (p,c)=>1`t | (p,d)=>1`t
                             | (q,a)=>1`r+1`s | (q,b)=>1`s | (q,d)=>1`t | _=>empty;
fun Release(x,y) = case (x,y) of (p,e)=>2`s+2`t | (q,c)=>1`r | (q,e)=>2`s+1`t | _=>empty;
var x : U;
var y : S;
var i : I;
```



variables:

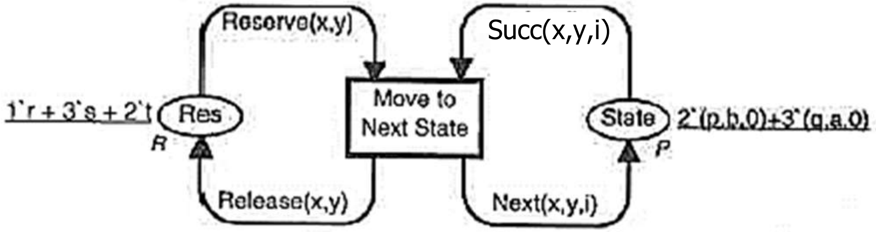
var *x* = *U* (processes)

var *y* = *S* (states, steps of processes)

var *i* = *I* (number of cycles)

Introductory example

```
color U = with p | q;
color S = with a | b | c | d | e;
color I = int;
color P = product U *S * I;
color R = with r | s | t;
fun Succ(y) = case y of a=>b | b=>c | c=>d | d=>e | e=>a;
fun Next(x,y,i) = (x, if (x,y) = (p,e) then b else Succ(y), if y=e then i+1 else i);
fun Reserve(x,y) = case (x,y) of (p,b)=>2`s | (p,c)=>1`t | (p,d)=>1`t
                              | (q,a)=>1`r+1`s | (q,b)=>1`s | (q,d)=>1`t | _=>empty;
fun Release(x,y) = case (x,y) of (p,e)=>2`s+2`t | (q,c)=>1`r | (q,e)=>2`s+1`t | _=>empty;
var x : U;
var y : S;
var i : I;
```



arc expression functions:

fun Succ(*y*) = case *y* of *a*⇒*b* | *b*⇒*c* | *c*⇒*d* | *d*⇒*e* | *e*⇒*a*;

fun Next(*x*,*y*,*i*) = (*x*, if (*x*,*y*) = (*p*,*e*) then *b* else Succ(*y*),
if *y*=*e* then *i*+1 else *i*);

fun Reserve(*x*,*y*) = case (*x*,*y*) of (*p*,*b*)⇒2`*s* | (*p*,*c*)⇒1`*t* |
(*p*,*d*)⇒1`*t* | (*q*,*a*)⇒1`*r*+1`*s* | (*q*,*b*)⇒1`*s* | (*q*,*d*)⇒1`*t* | _⇒empty;

fun Release(*x*,*y*) = case (*x*,*y*) of (*p*,*e*)⇒2`*s*+2`*t* | (*q*,*c*)⇒1`*r* |
(*q*,*e*)⇒2`*s*+1`*t* | _⇒empty;

Homework

- Convert this PT-net into CP-net form!