

Modellezés és szimuláció Petri hálókkal

dr. Bartha Tamás

Dr. Pataricza András

BME Méréstechnika és Információs Rendszerek Tanszék

Diszkrét rendszermodellezés célja

- Informatikai rendszerek: jól tagoltak
 - rendszerépítés a komponensek integrációjával
 - elemi komponensek kapcsolata
 - explicit logikai kapcsolat: sorrendiség, ok-okozati függőség
 - implicit függőség: pl. osztott erőforrás használata
- Célkitűzés: minőségi vagy/és mennyiségi analízis
 - kvalitatív: logikai helyességbizonyítás
 - kvantitatív: teljesítményelemzés, megbízhatóság és rendelkezésre állás, biztonságosság

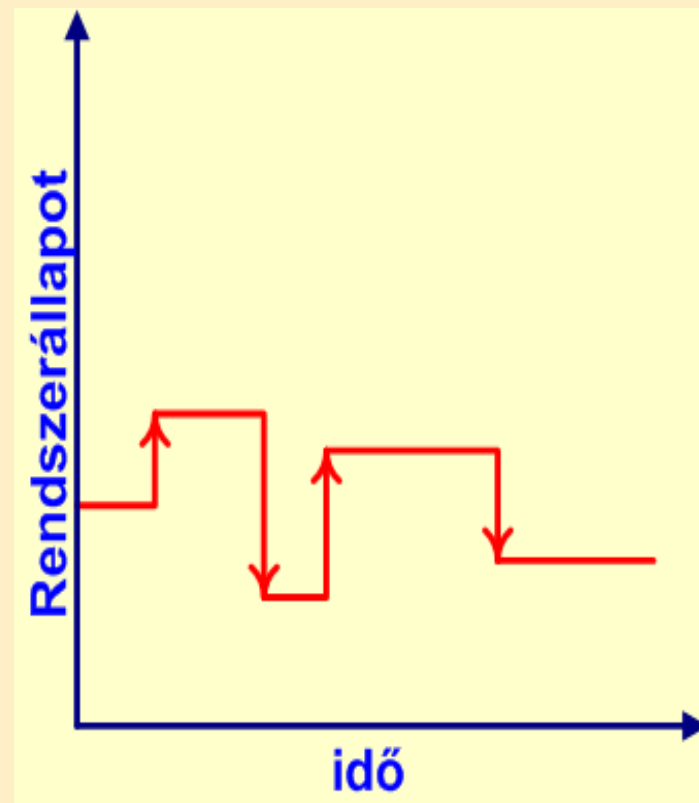
Rendszermodellek felbontása

- Modellezés célja

- Korreláció: modell $\leftrightarrow$ „valóság”
- Érthetőség, áttekinthetőség
 - Nem cél a minimális modell
- A teljes rendszer dinamikája
 - Komponensek modellje egyszerű
 - Bonyolult kölcsönhatások

- Csoportosítás

- Folytonos
 - Diszkrét
- } • értékben
• időben

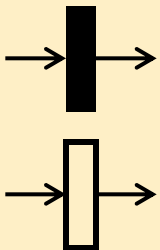


Tevékenységek

- Modellezés hierarchikus és funkcionális
 - (rész)folyamatok
 - tovább nem bontott, elemi lépések → tevékenységek
- Tevékenység
 - elemi lépés
 - felhasznált erőforrások
 - végrehajtási idő
 - determinisztikus
 - sztochasztikus

Tevékenységek modellezése Petri hálóokban

Petri háló alapú modellek esetén a tevékenység:

- elemi lépés $\rightarrow$ tranzíció tüzelése
 - felhasznált erőforrások $\rightarrow$ bemeneti / kimeneti helyek
 - végrehajtási idő
 - determinisztikus
 - sztochasztikus
-  $\rightarrow$ determinisztikus időzítésű tranzíció
 $\rightarrow$ exponenciális időzítésű tranzíció

Az engedélyezettséggel kapcsolatban felmerülő kérdések:

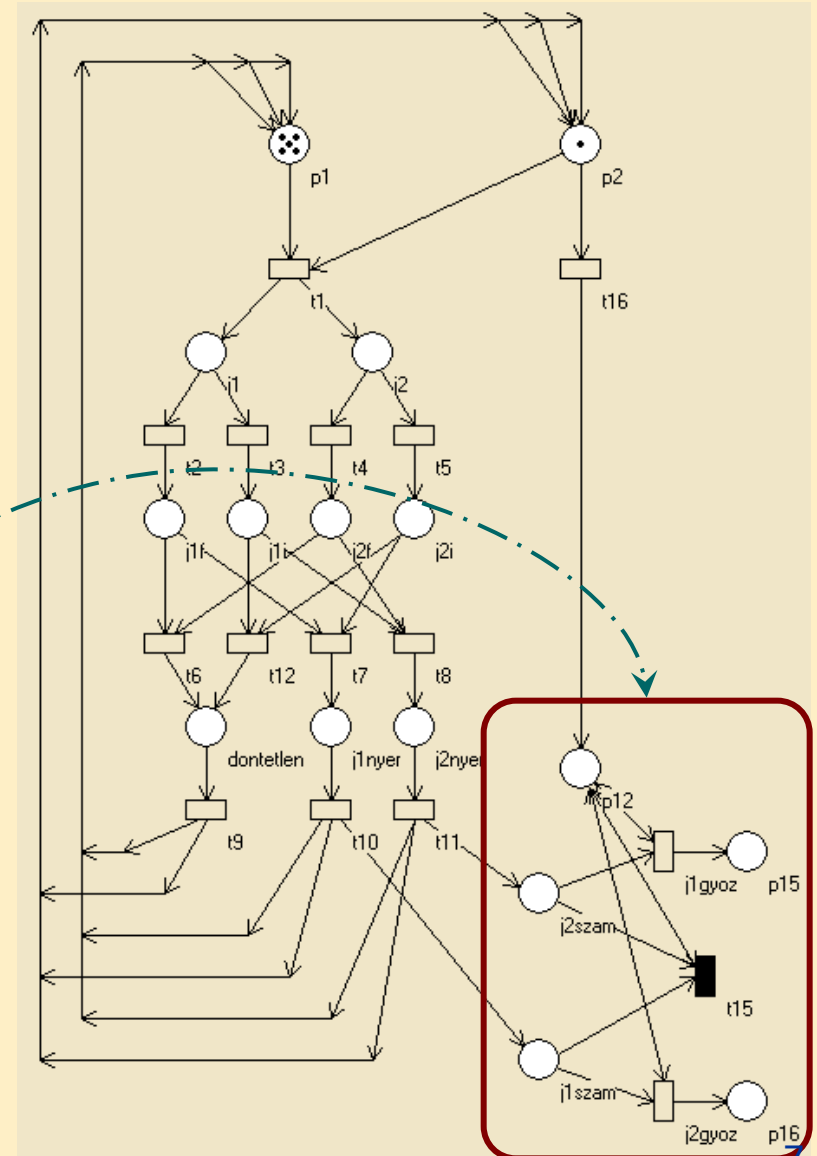
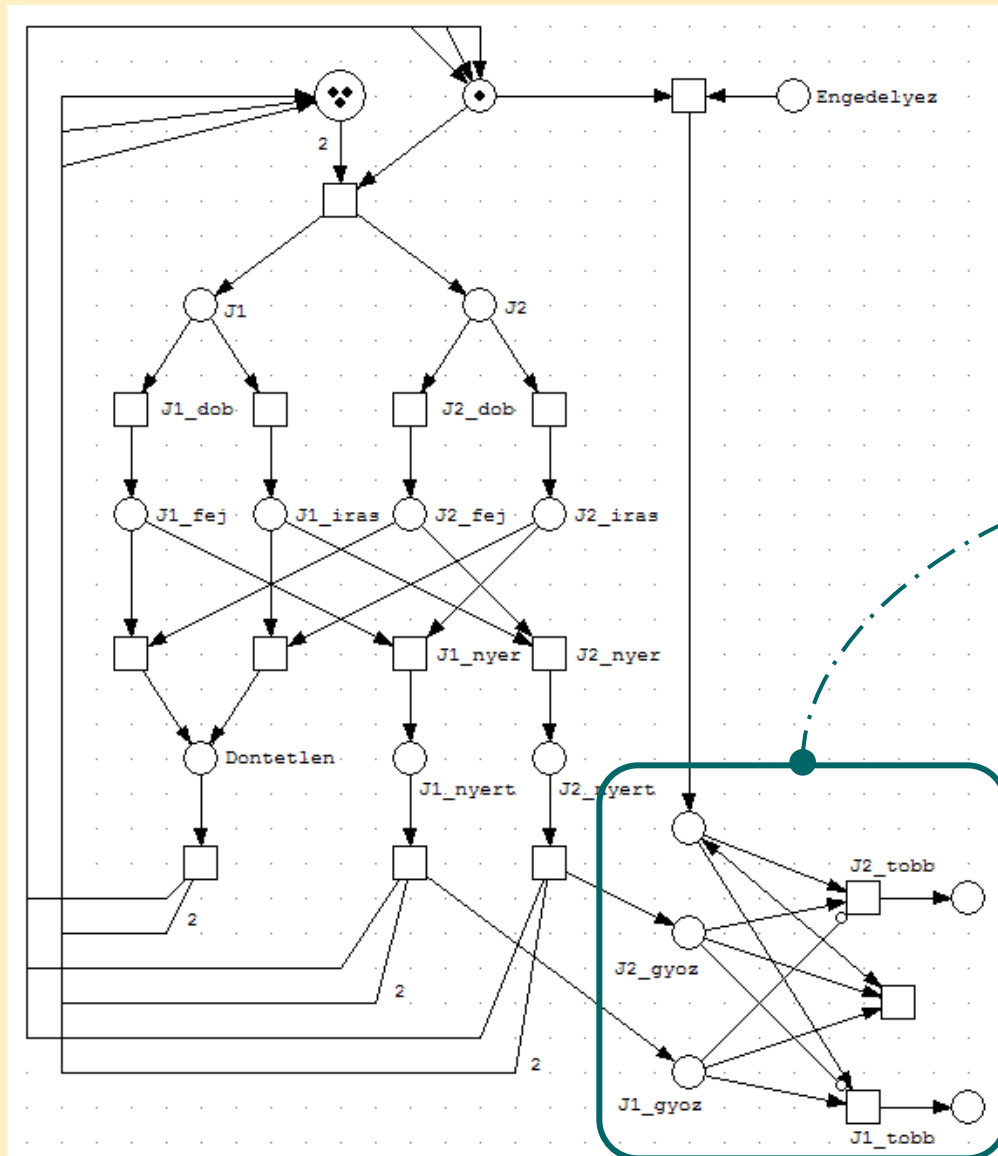
- Időzítetlen tranzíciók „prioritása” nagyobb: előbb tüzelnek
- Engedélyezettség megszűnése: mi lesz a gyújtási idővel?
 - újrainduló (újra sorsol)
 - megőrződő (korábban sorsoltból fennmaradó idő folytatódik)

Sztochasztikus Petri hálók

$$\mathbf{SPN} = (P, T, E, W, M_0, \Lambda)$$

- tüzelési intenzitás $\Lambda : T \rightarrow \mathbb{R}$
- időzítés: τ valószínűségi változó
 - eloszlásfüggvény: $P\{\tau < t\} = F(t) = 1 - e^{-\lambda t}$
 - sűrűségfüggvény: $f(t) = \lambda e^{-\lambda t}$
- tüzelési szabály változik: intenzitás $\sim$ prioritás
- két exponenciális eloszlás minimuma is az: $\lambda_1 + \lambda_2$
- PT és SPN elérhetőségi gráfja azonos
- időzített elérhetőségi gráf: folytonos Markov lánc

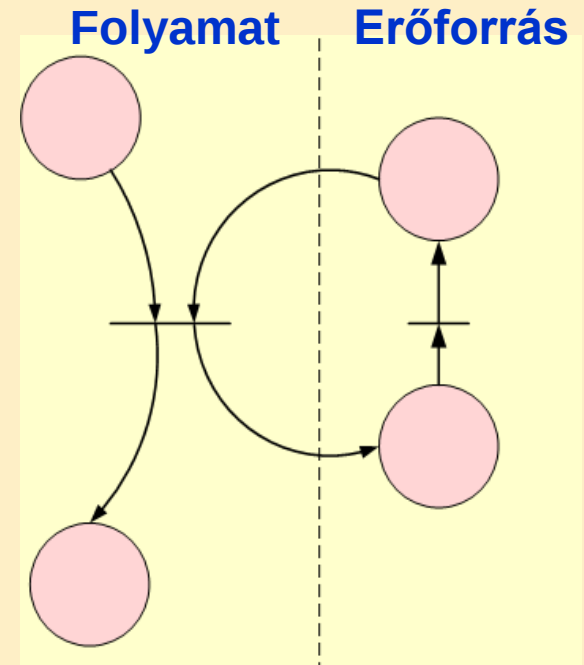
Időzítetlen tranzíciók prioritásának felhasználása



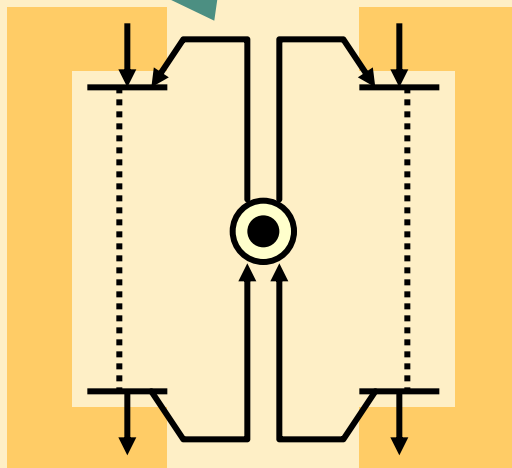
Erőforrás allokáció Petri hálókbán

- Kölcsönös kizárás
- Több darab lefoglalása

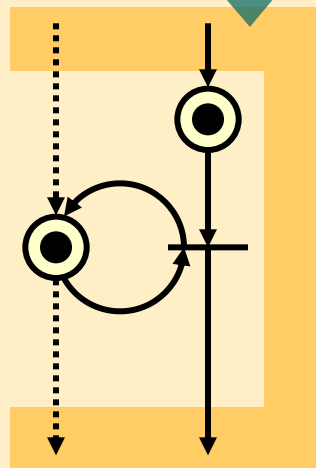
Cél: nem a minimális,
hanem az érthető Petri háló!



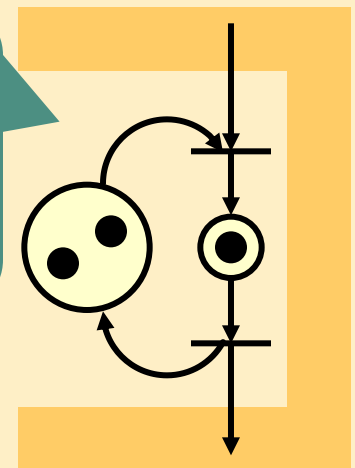
Kölcsönös kizárás
megvalósítása



Állapotváltozó
leolvasása

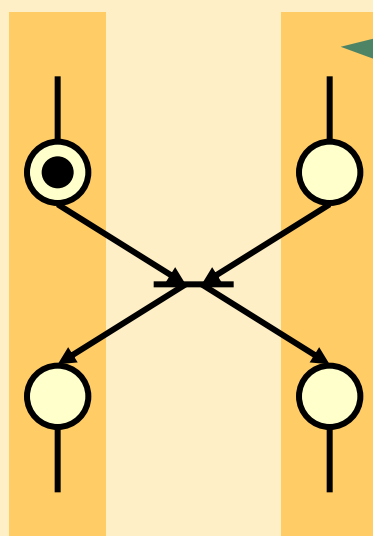


Korlátos
erőforrás
kapacitás
modellezése



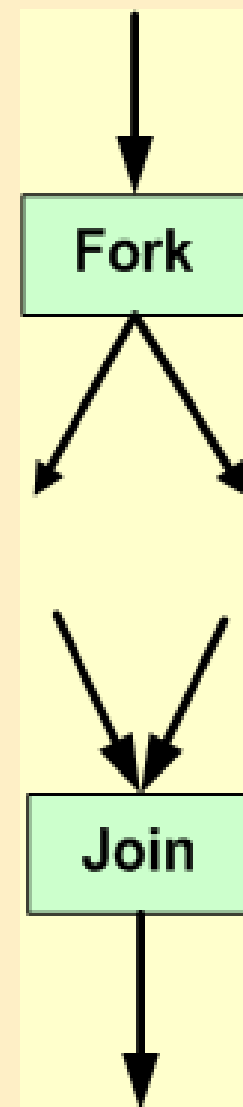
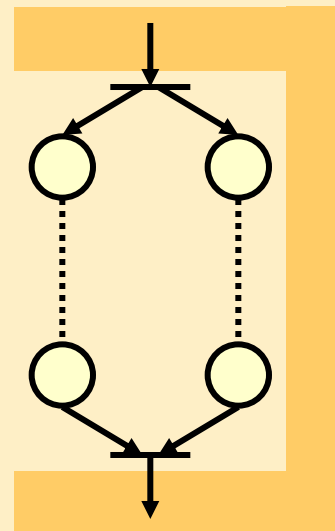
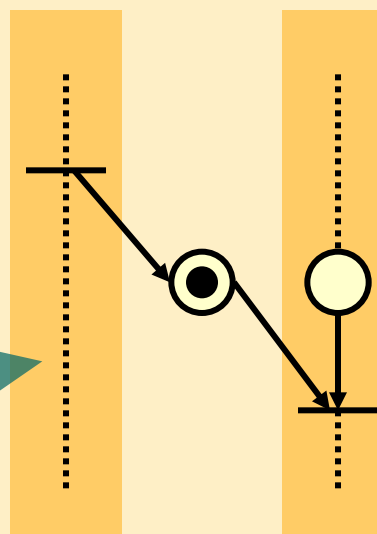
Üzenetek

- Szoftverben párhuzamosság
 - FORK - elágazás
 - JOIN - visszatérés
- Kommunikáció biztosítása
 - üzenetekkel
- Wait – egymásra várás



Szinkron
kommunikáció

Aszinkron
(levelesláda)
kommunikáció



Rendszermodellezés

Modellépítés folyamata:

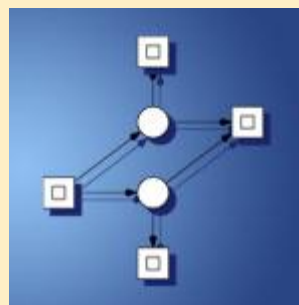
- a három fő modellelem-fajta:
 - folyamatok, illetve tevékenységek
 - erőforrások
 - a logikailag csatolt folyamatok közti esetleges üzenetek
- felépítése először egyedileg
- majd ezekből a modell összeintegrálása

Rendszermodellezés

Modellépítés elemei:

1. a folyamat modellje (az erőforrás használat, illetve üzenetváltás részletes feltüntetése nélkül)
2. minden egyes erőforráshoz fel építeni
 - a foglalt/szabad állapotot jellemző kétállapotú véges automata modellrészt,
 - valamint az üzenetek pufferjének modelljét
3. a folyamat és erőforrás (illetve az üzenetek pufferje) modelljében a megfelelő állapotátmenetek összevonása

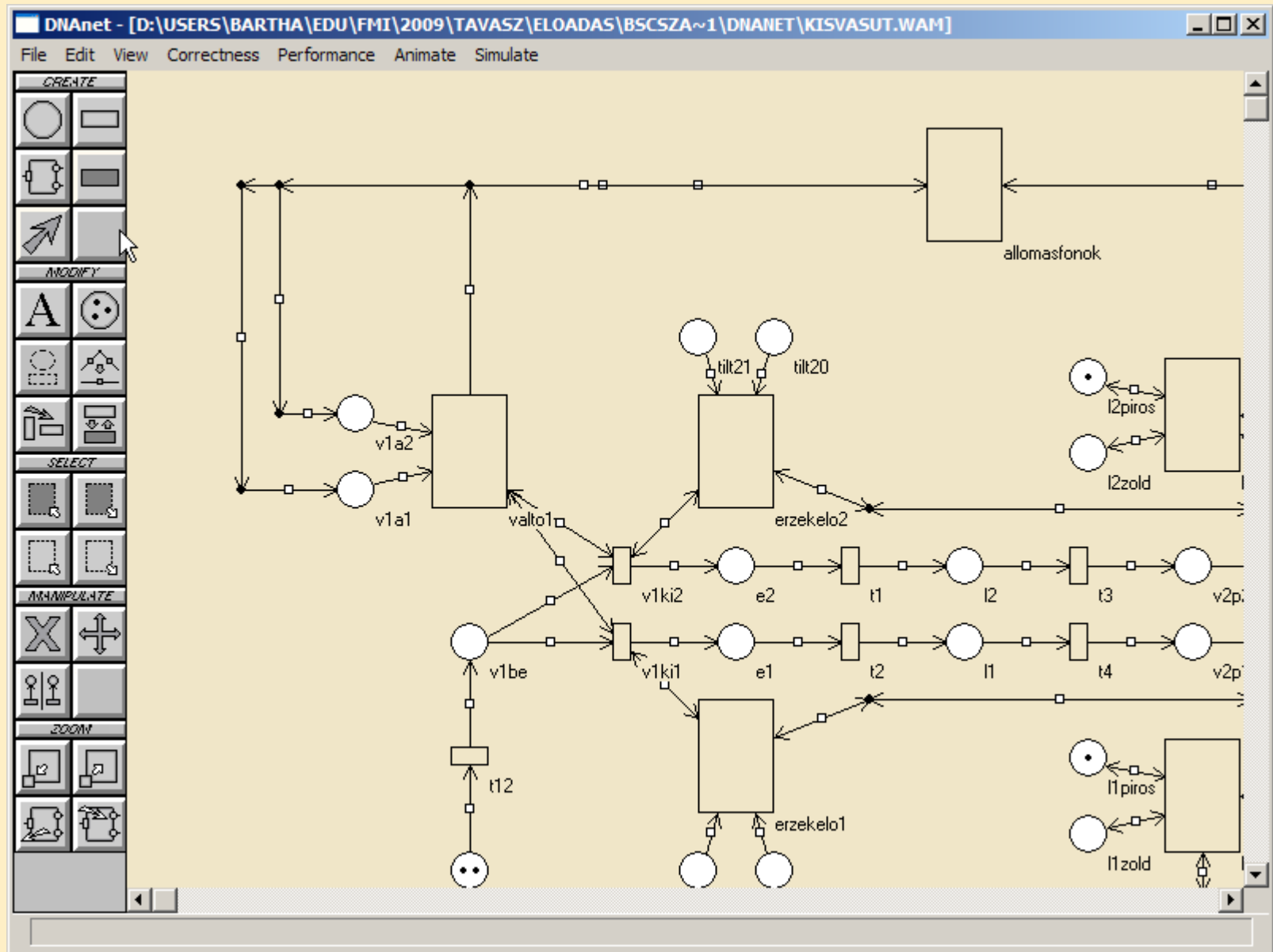
Modellező eszközök: DNAnet, Snoopy, PetriDotNet



A DNAnet modellező program

- Képességei
 - grafikus szerkesztő
 - interaktív animáció (token game)
 - egyszerű analízis: dinamikus tulajdonságok ellenőrzése
 - nem interaktív szimuláció (teljesítmény analízishez)
- Előnyei
 - kicsi, kompakt, gyors, egyszerűen kezelhető
 - méretéhez képest sokat tud
 - ingyenes, szabad felhasználású
- Hátránya
 - nem minden környezetben futtatható
 - nem túl stabil ☹

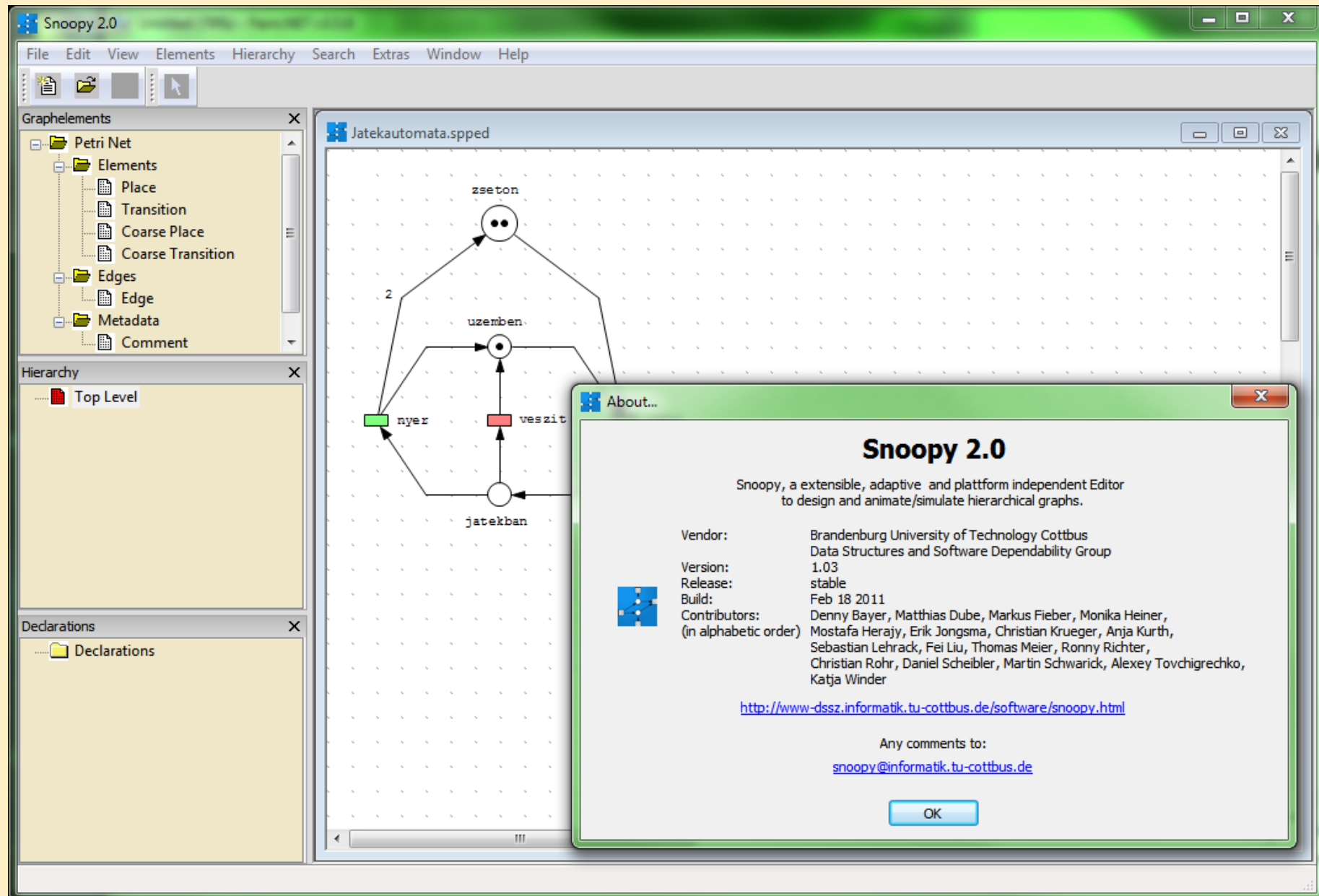
A DNAnet modellező program képe



A Snoopy modellező program

- Snoopy (Windows, Linux)
 - (kizárólag) grafikus szerkesztő
 - egyszerűen kezelhető
 - kényelmi funkciók: copy / paste, undo / redo
 - Token Game (animált)
 - kiterjesztések: tiltó él, olvasó él, reset él, egyenlőség él
 - számos háló típus, többek között színezett háló is
 - támogatja hierarchikus Petri hálók készítését
 - elemek színezése, méretezése, élsúlyok kijelzése
 - on-line help (hiányos)

A Snoopy modellező program képe



Analízis eszközök Snoopy-hoz

- Charlie (Java)
 - dinamikus tulajdonságok, elérhetőség
 - strukturális tulajdonságok, invariánsok
 - explicit CTL és LTL modellellenőrző
- INA (Windows, Linux)
 - szöveges felületű **parancssori** program
 - Token Game (szöveges)
 - invariáns analízis, elérhetőségi gráf generálás
 - strukturális tulajdonságok ellenőrzése
 - szimulációs képességekkel nem rendelkezik

A PetriDotNet modellező program

- Képességei
 - grafikus szerkesztő + Token Game + szimuláció
 - egyszerűen kezelhető, sok kényelmi funkció
 - kiterjesztések: tiltó él, időzítés, **színezett háló**
 - támogatja hierarchikus Petri hálók készítését
 - kiegészítő modulokkal bővíthető, pl. analízis modulok
 - dinamikus tulajdonságok, CTL modellellenőrző
 - elemek színezése, elforgatása, élsúlyok kijelzése
 - szabványos PNML fájlformátum, van hozzá INA kimenet
- Hazai ☺ fejlesztés: Darvas Dániel fejleszti

A Petri.NET modellező program képe

The screenshot displays the PetriDotNet software interface. The main window, titled "PetriDotNet - [Jatekautomata]", features a menu bar (File, Edit, View, Insert, Mode, Tools, Add-in, Window, Help) and a toolbar with various editing tools. On the left, there is a "Toolbox" with buttons for "Select", "Place", "Transition", "Edge", and "Token", along with an "Other elements" button. Below the toolbox is a "Properties" panel showing the "Identity" of the selected element, with fields for "Id" (n1) and "Name" (Net 1). At the bottom left, there is a "Hierarchy" section with a dropdown menu showing "Net1".

The central workspace contains a Petri net diagram with the following components:

- Places:** "zseton" (top), "nyer" (bottom left, green), "jatekban" (bottom center), "uzemben" (top center), "veszit" (bottom center, red), and "indul" (bottom right, grey).
- Transitions:** "nyer" (bottom left, green), "veszit" (bottom center, red), and "indul" (bottom right, grey).
- Edges:** Directed edges connect the places and transitions. A yellow box with the number "2" is placed on the edge from "zseton" to "nyer".

An "About PetriDotNet" dialog box is open in the bottom right corner. It features the "petridotnet" logo and the text "from BME-MIT". The dialog provides the following information:

- Version 1.3.4098.34586
- Petri Net Editor by Dániel Darvas, 2009-2011 at BME-MIT (BUTE DMIS)
- based on Petri.NET 1.0 by Bertalan Szilvási (advisor: Gábor Huszerl), 2008
- Advisors: András Vörös (BME-MIT), dr. Tamás Bartha (BME-MIT)
- Contact us at <http://petridotnet.inf.mit.bme.hu/> or petridotnet@inf.mit.bme.hu.

At the bottom of the dialog box, there is a "Send error feedback" button and an "OK" button. A small logo for "MŰEGYETEM 1782" is visible at the bottom left of the dialog box.

Petri háló alapú modellek készítése (Példa a modellezési folyamatra)

A modellezési feladat

Alternating Bit Protocol

- Átviteli protokoll veszteséges csatornához
 - üzenet elveszhet (véges számú alkalommal)
 - üzenet tartalma nem változhat
- Cél: a protokoll biztosítsa, hogy minden üzenet véges időn belül eljusson a vevőhöz

Küldő folyamat

- Üzenetekhez egy ellenőrző bitet kapcsol
- Az üzenetek megérkezését nyugta jelzi
- A nyugta tartalmazza az ellenőrző bitet
- Első üzenethez csatolt bit: b^0
 - ha az üzenet elvész, a folyamat időtúllepéssel észleli a nyugta hiányát $\rightarrow$ újra küldi
 - ha a folyamat b^0 bittel ellátott nyugtát kap (ilyet várt), akkor a következő üzenethez $b^1 = \neg b^0$ bitet köt
 - ha a folyamat b^x bittel ellátott nyugtát vár és b^y bittel ellátott nyugtát kap $\rightarrow$ egyszerűen eldobja

Fogadó folyamat

- Első vétel: b^0 ellenőrző bittel jelölt üzenetet kap
- Az üzenetet feldolgozza, a vételt a kapott bit visszaküldésével nyugtázza
 - ha a következő üzenetben az ellenőrző bit értéke b^1 (helyesen), akkor az új üzenetet is feldolgozza és a b^1 bit visszaküldésével nyugtázza
 - ha a következő üzenetben az ellenőrző bit értéke b^0 (nem megfelelő), akkor az üzenetet eldobja (korábban már feldolgozta), de nyugtát küld

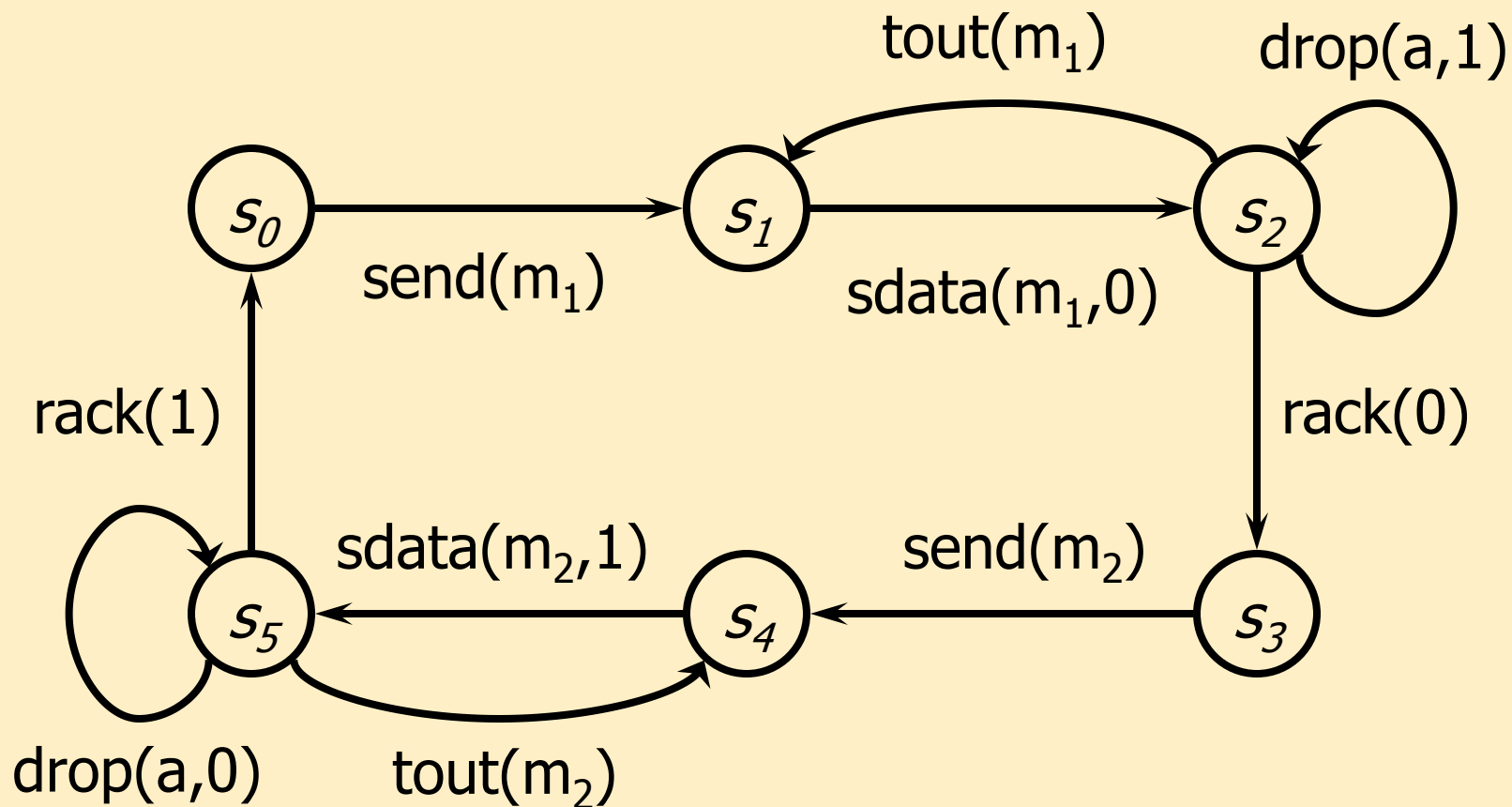
A modellalkotás lépései

1. A feladat felbontása tevékenységekre és erőforrásokra
2. Tevékenységek állapotgráfjának kidolgozása
3. Erőforrások modellje a pufferek modelljeivel
4. Állapotgráf modellekből Petri háló modell készítése
5. Tevékenységek és erőforrások kapcsolatának leírása
6. Tevékenység és erőforrás modellek integrálása
7. Integrált modell helyességének ellenőrzése
8. Modell felhasználása a (kvantitatív) feladat megoldására

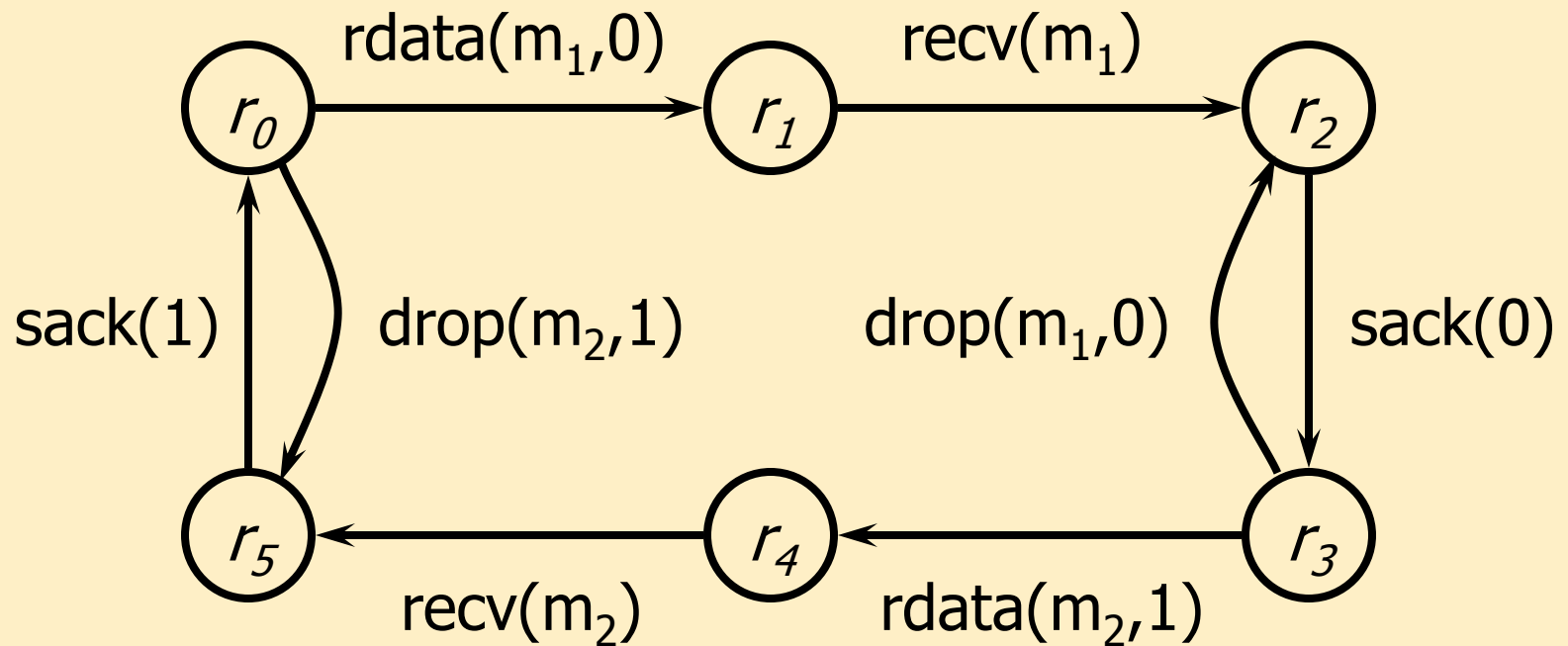
Komponensek és állapotgráfjaik

- Alrendszerek
 - Tevékenységek: küldő folyamat, fogadó folyamat
 - Erőforrások: adat csatorna, nyugtázó csatorna
- Minden alrendszer saját állapottal rendelkezik
 - körök: állapotok
 - nyilak: események
- Azonos események egy időben mennek végbe: szinkronizáció

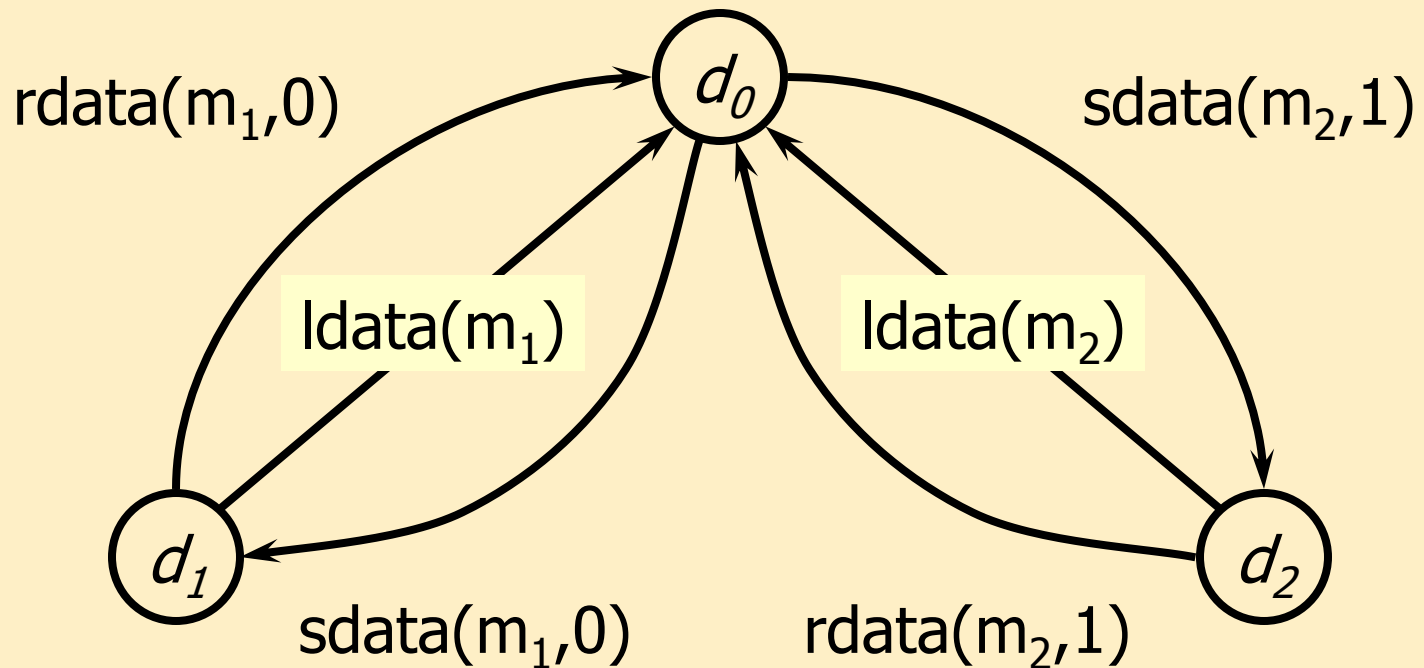
Küldő folyamat állapotgráfja



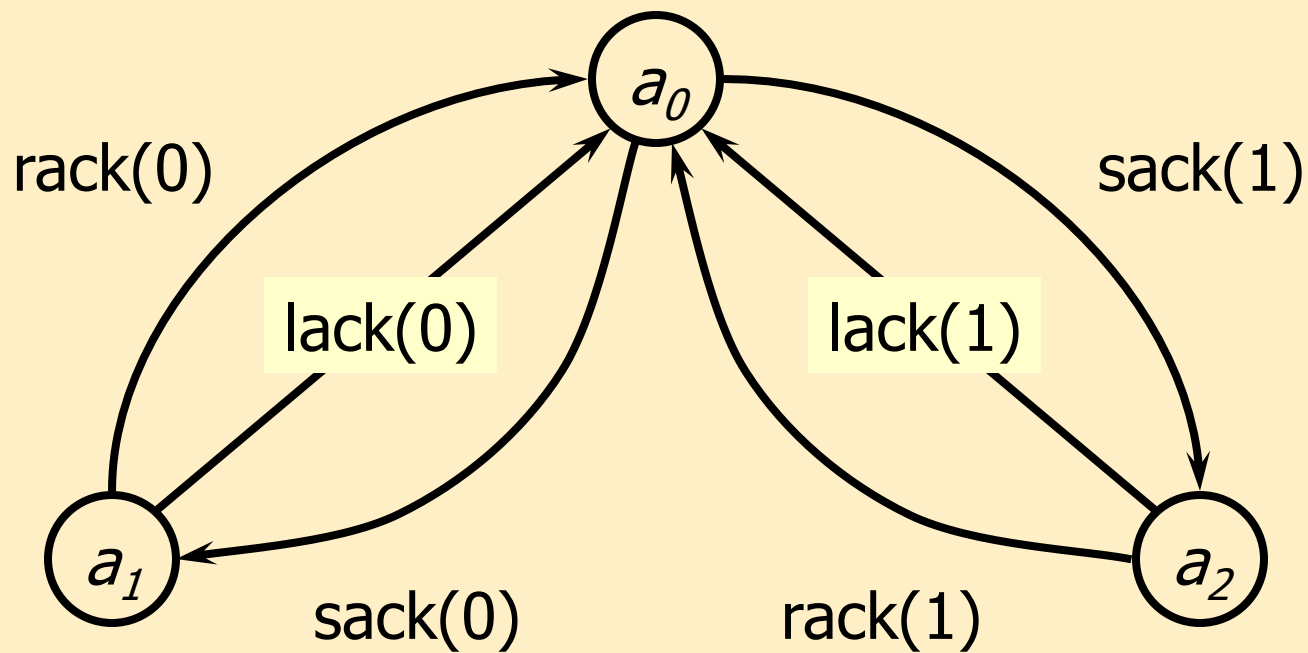
Fogadó folyamat állapotgráfja



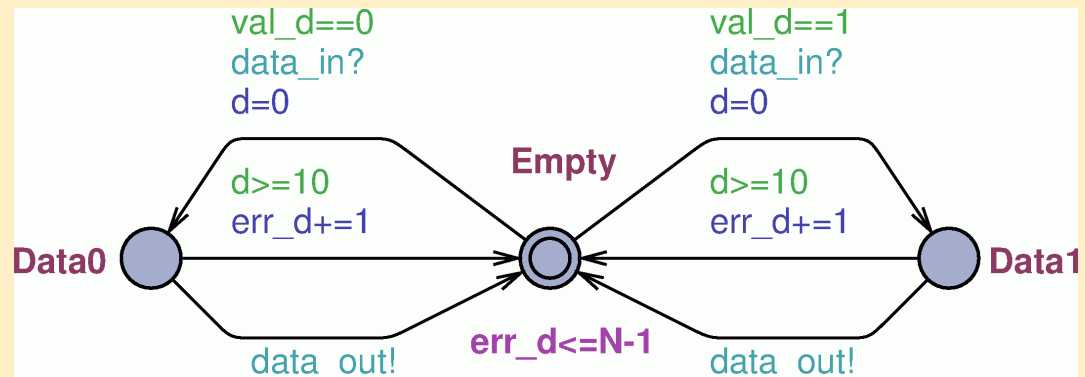
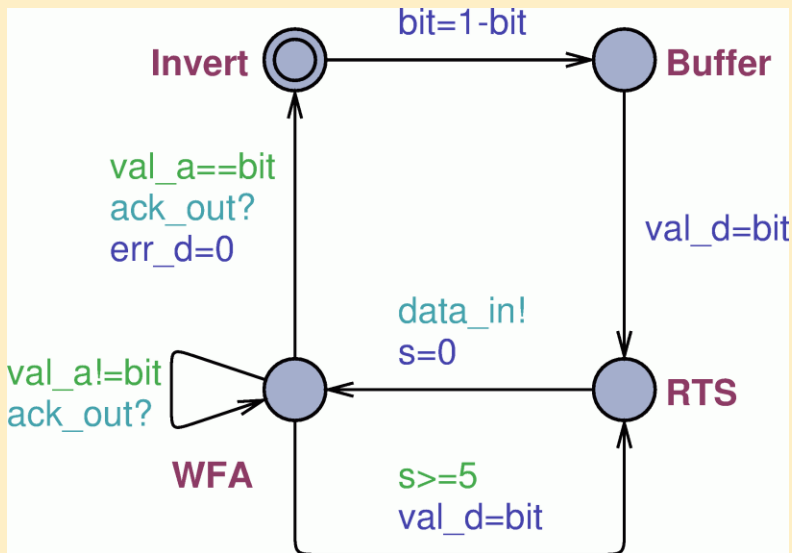
Adat csatorna állapotgráfja



Nyugtázó csatorna állapotgráfja



UPPAAL modell

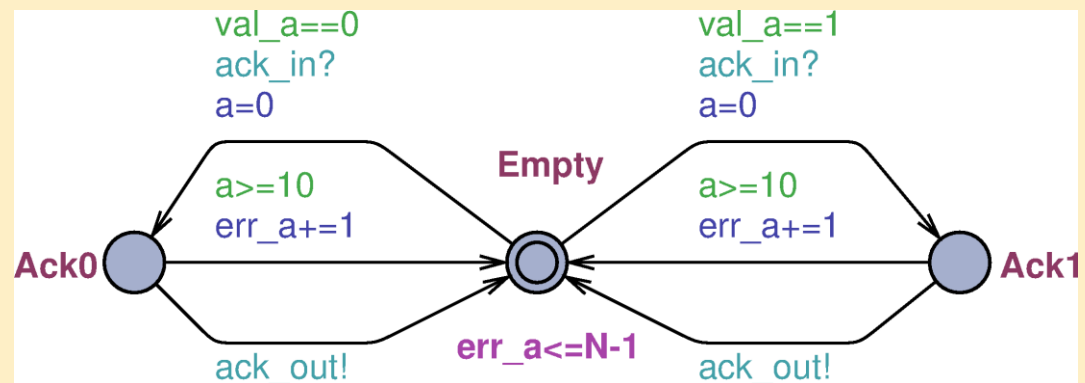
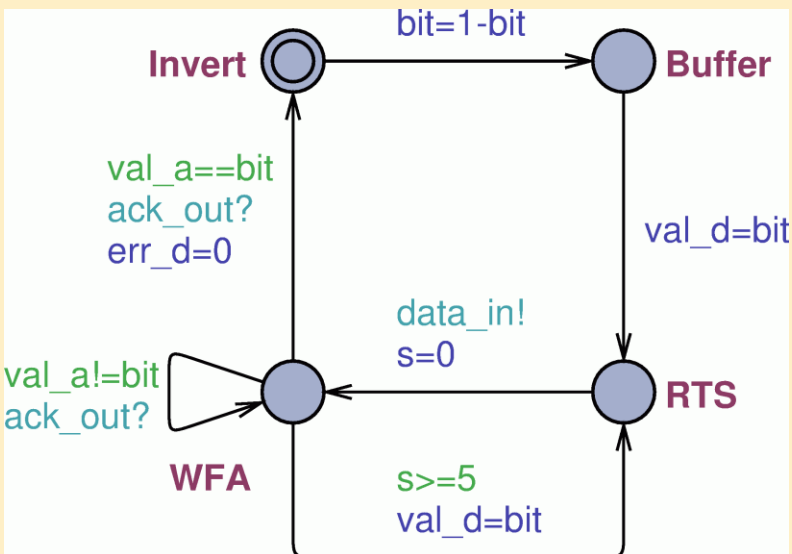


```
const int N=10;
```

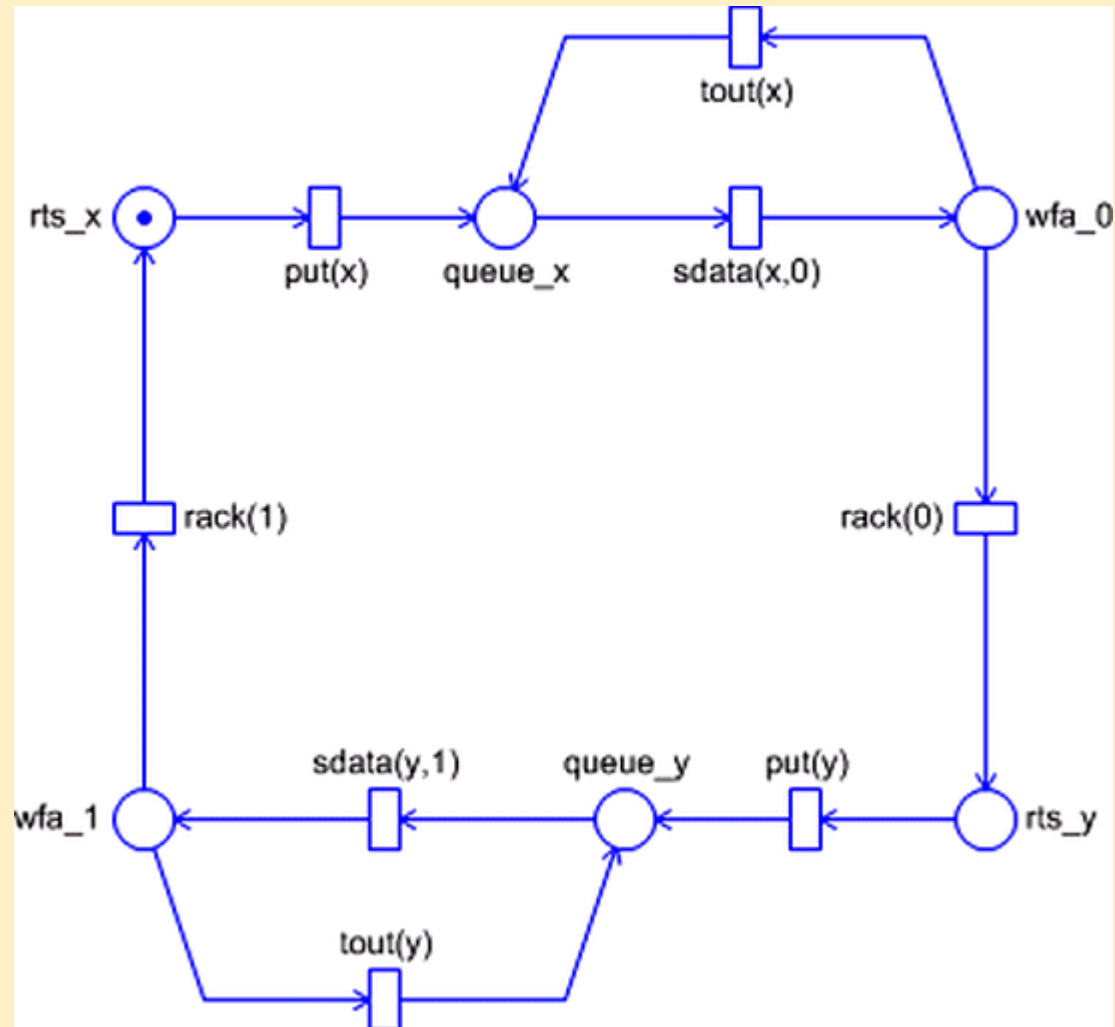
```
value_t val_d, val_a;  
error_t err_d, err_a;
```

```
typedef int[0,1] bit_t;  
typedef int[0,2] value_t;  
typedef int[0,N] error_t;
```

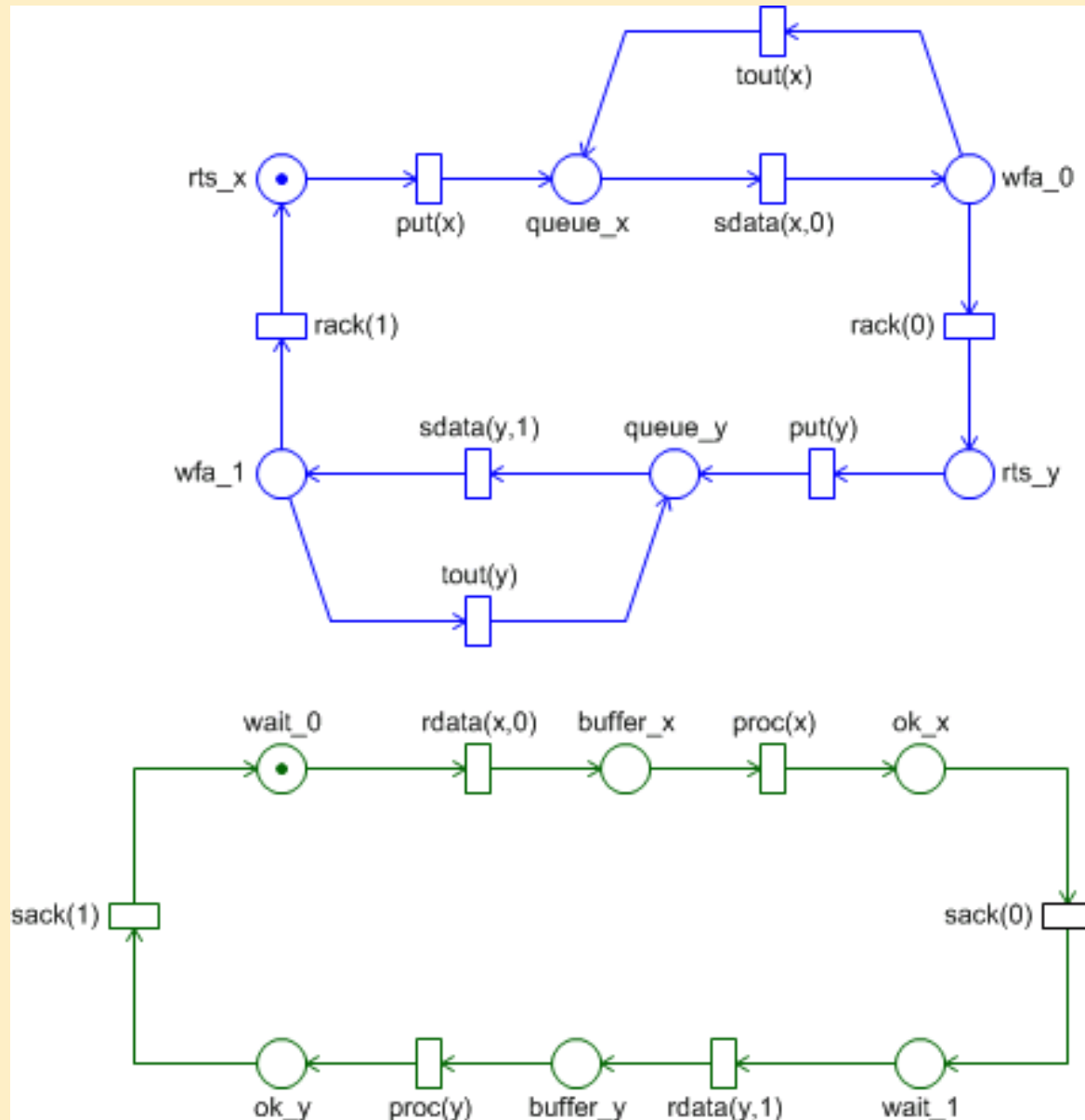
```
chan data_in, ack_in;  
urgent chan data_out, ack_out;
```



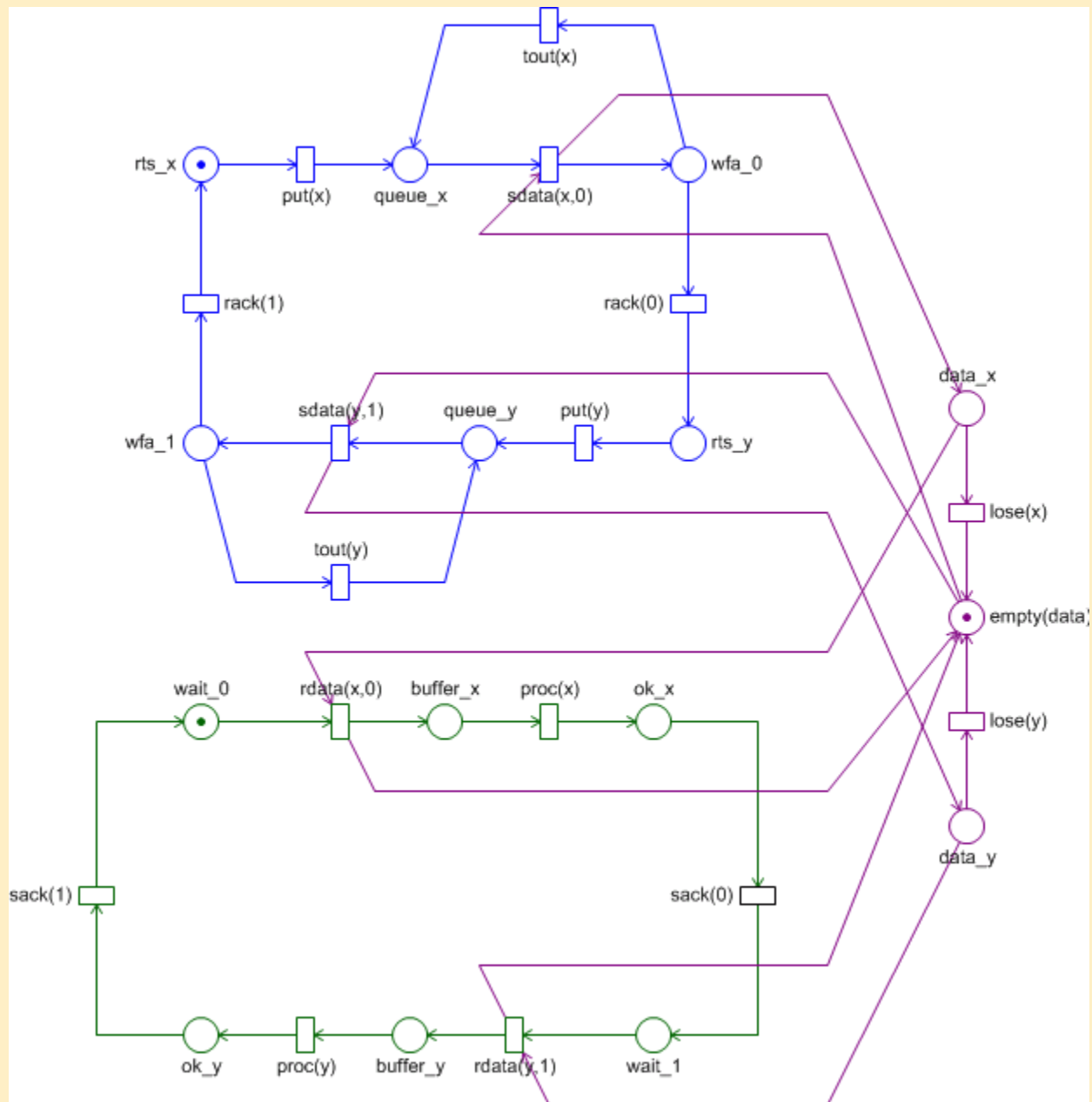
Küldő folyamat Petri háló modellje



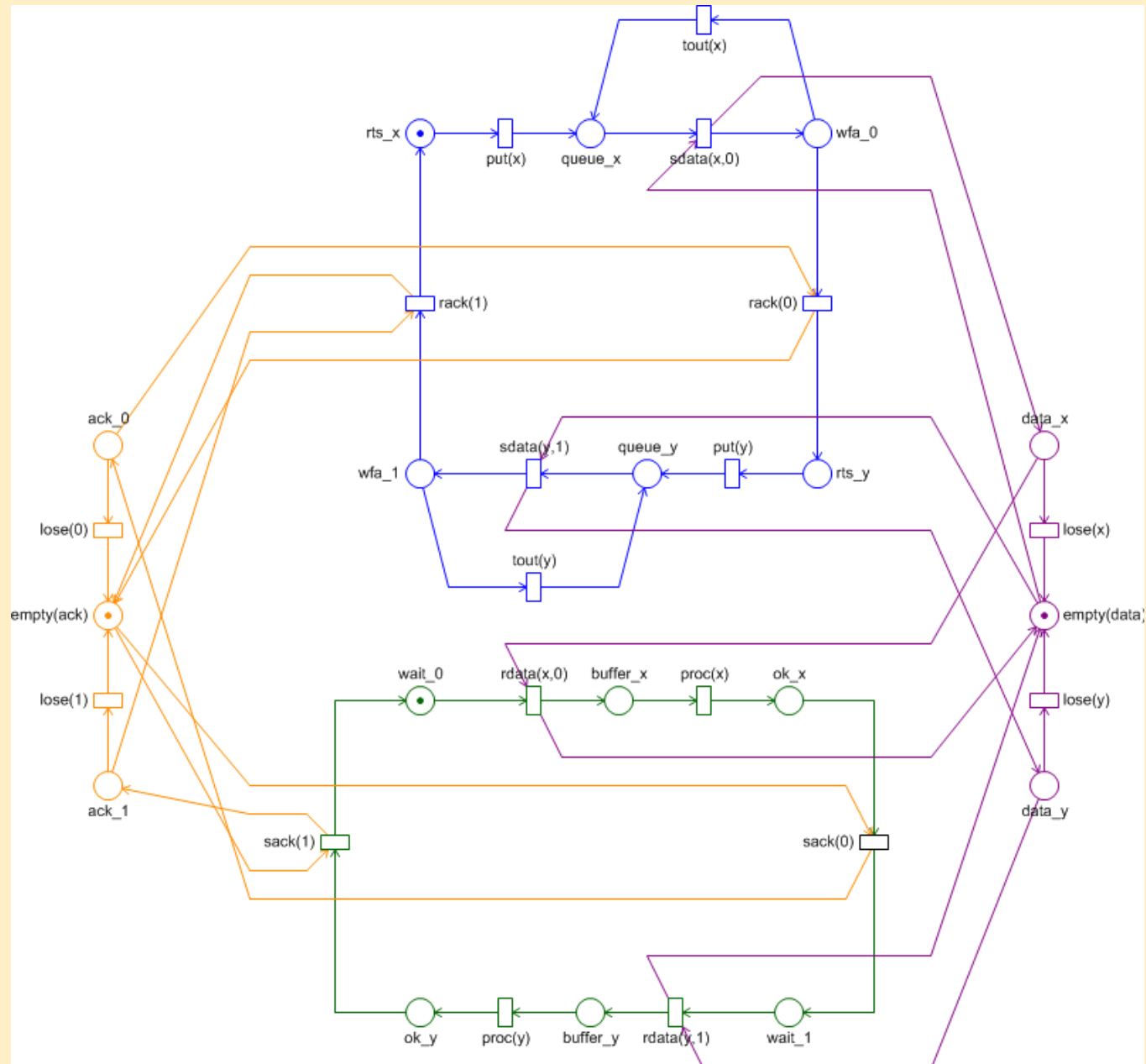
Fogadó folyamat Petri háló modellje



Adat csatorna és az átvitel

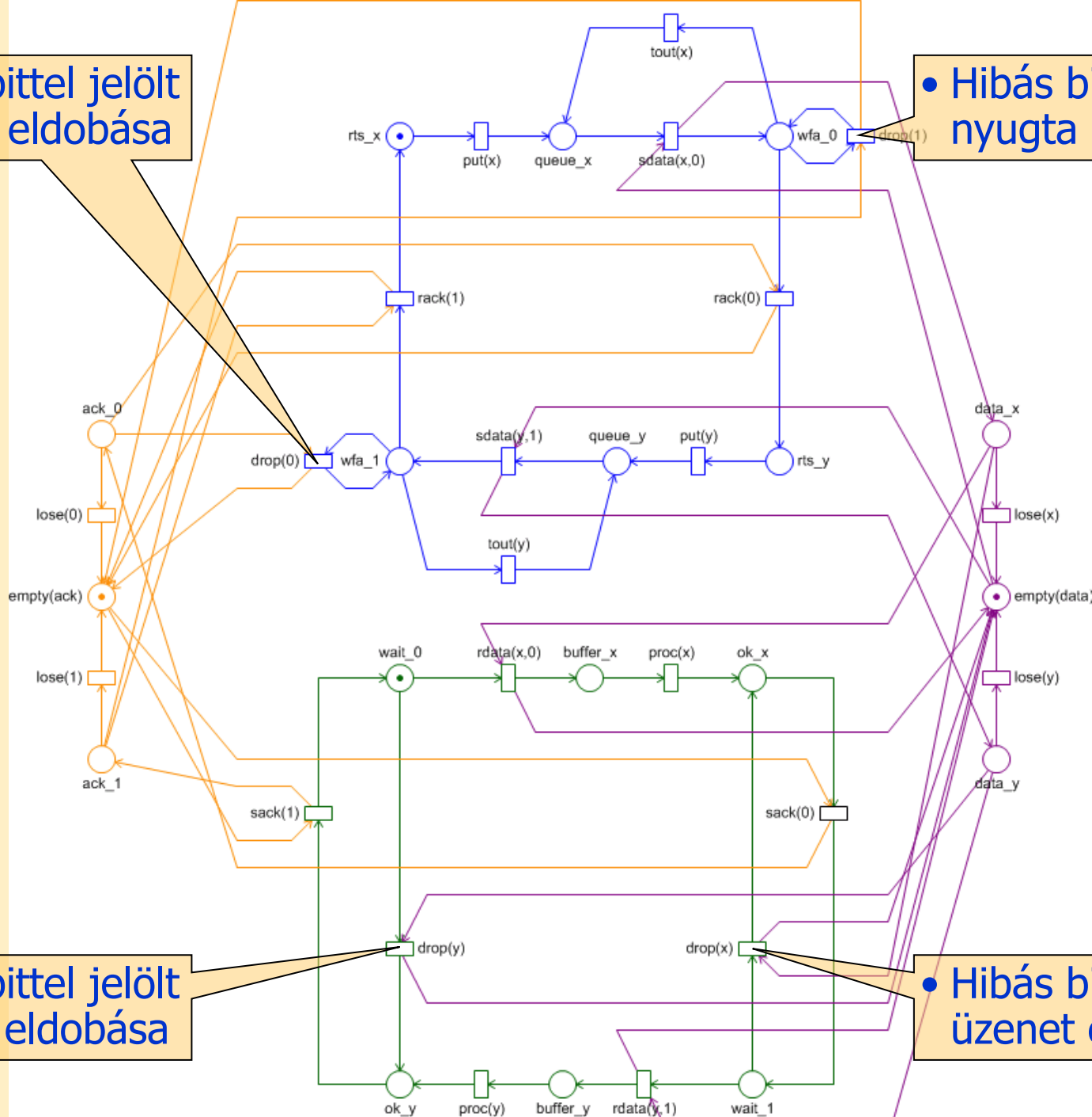


Nyugtázó csatorna



- Hibás bittel jelölt nyugta eldobása

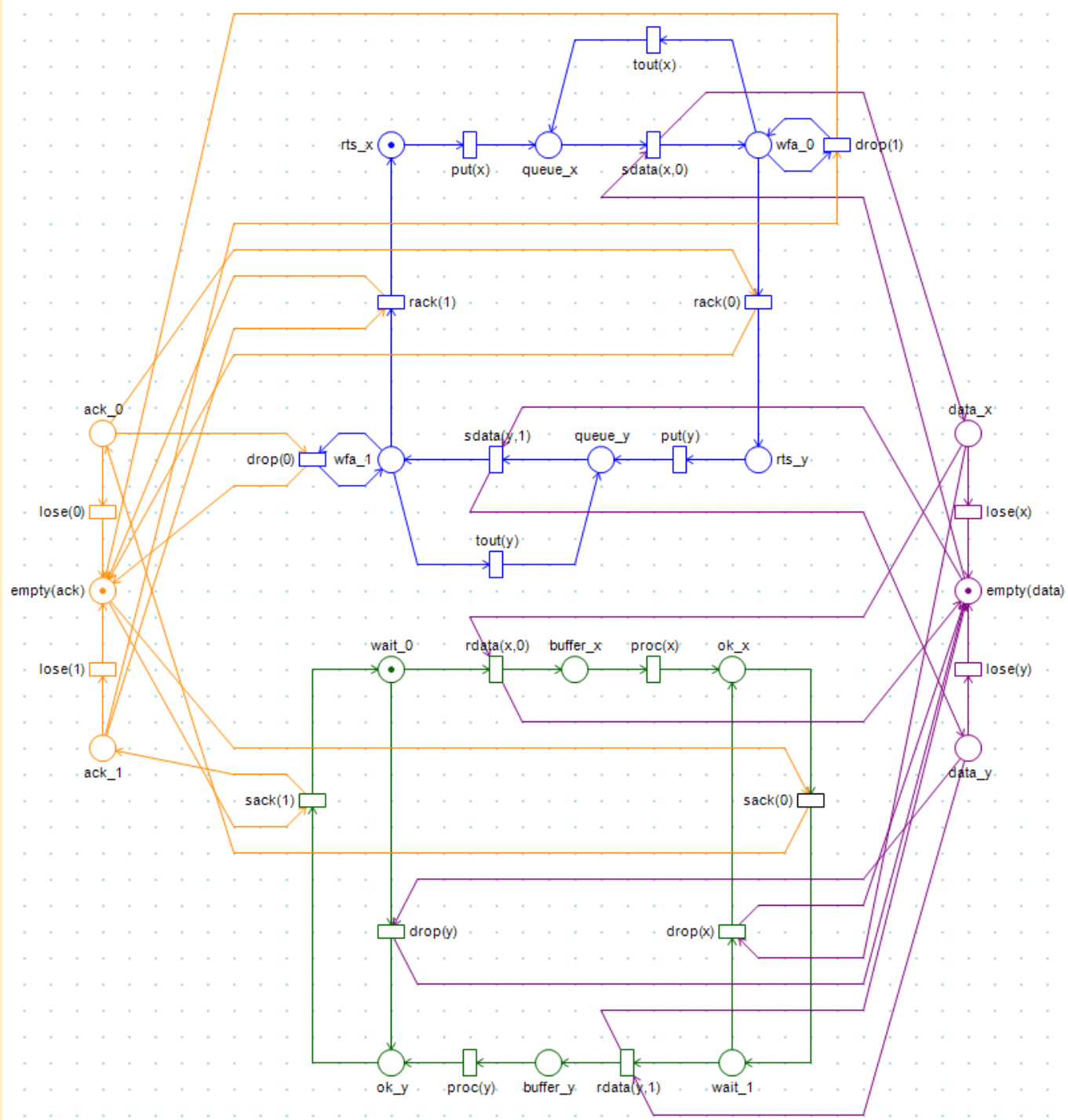
- Hibás bittel jelölt nyugta eldobása



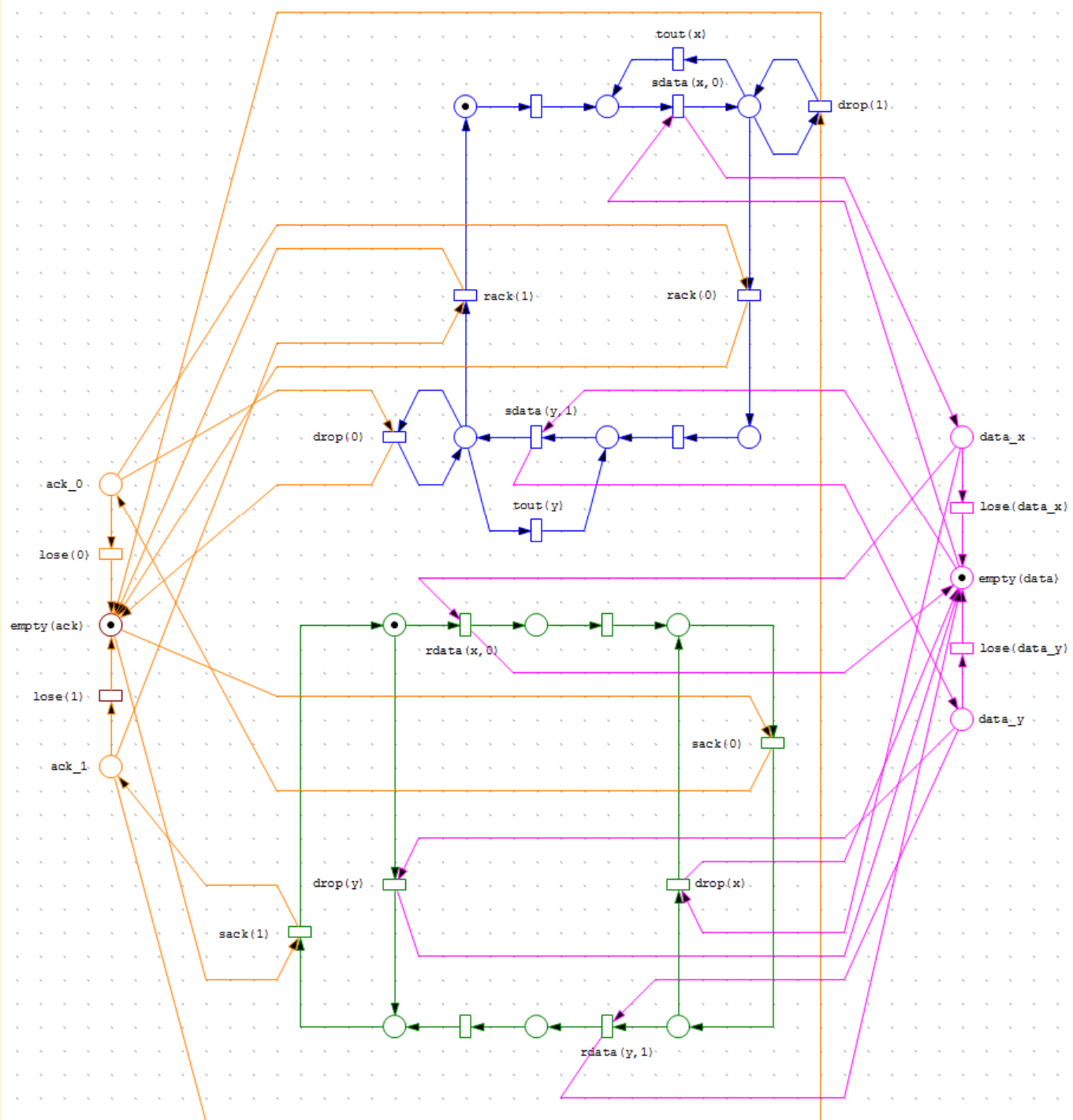
- Hibás bittel jelölt üzenet eldobása

- Hibás bittel jelölt üzenet eldobása

PetriDotNet: A teljes modell



Snoopy: A teljes modell



Petri háló modellek „tesztelése”

Szimuláció, token játék algoritmusai

Vizsgálati lehetőségek

Az elemzés mélysége szerint:

- Szimuláció
 - Állapottér bejárása
 - elérhetőségi gráf analízis
 - dinamikus (viselkedési) tulajdonságok
 - Strukturális tulajdonságok
 - invariáns analízis
- ha mindez nem vezet eredményre
- ↓
- Algebrai közelítés, részleges döntés
-

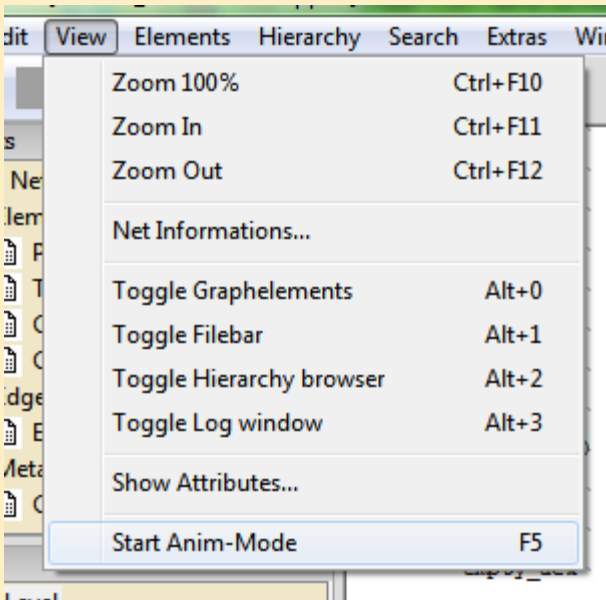
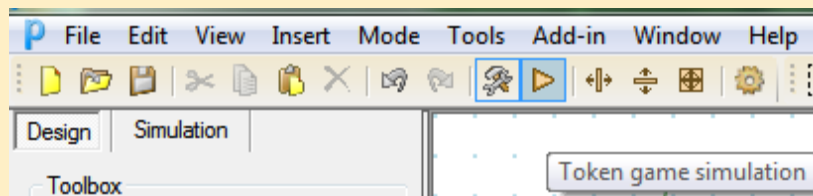
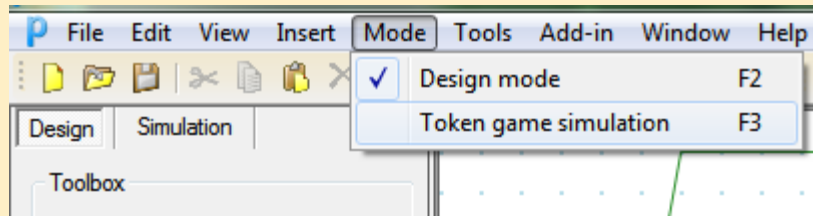
Diszkrét rendszerek szimulációja

- Cél: a vizsgált rendszer „valósághű” modellezése
 - valósághű $\sim$ validáció (modell megfelelőség)
- Szimuláció folyamatmodellek esetén
 - tevékenységorientált (eseményorientált)
 - csak az események időpontjait tartjuk nyilván
 - tevékenységek kezdete és vége (vagy időtartama)
 - erőforrások lefoglalása és elengedése

Petri hálók szimulációja

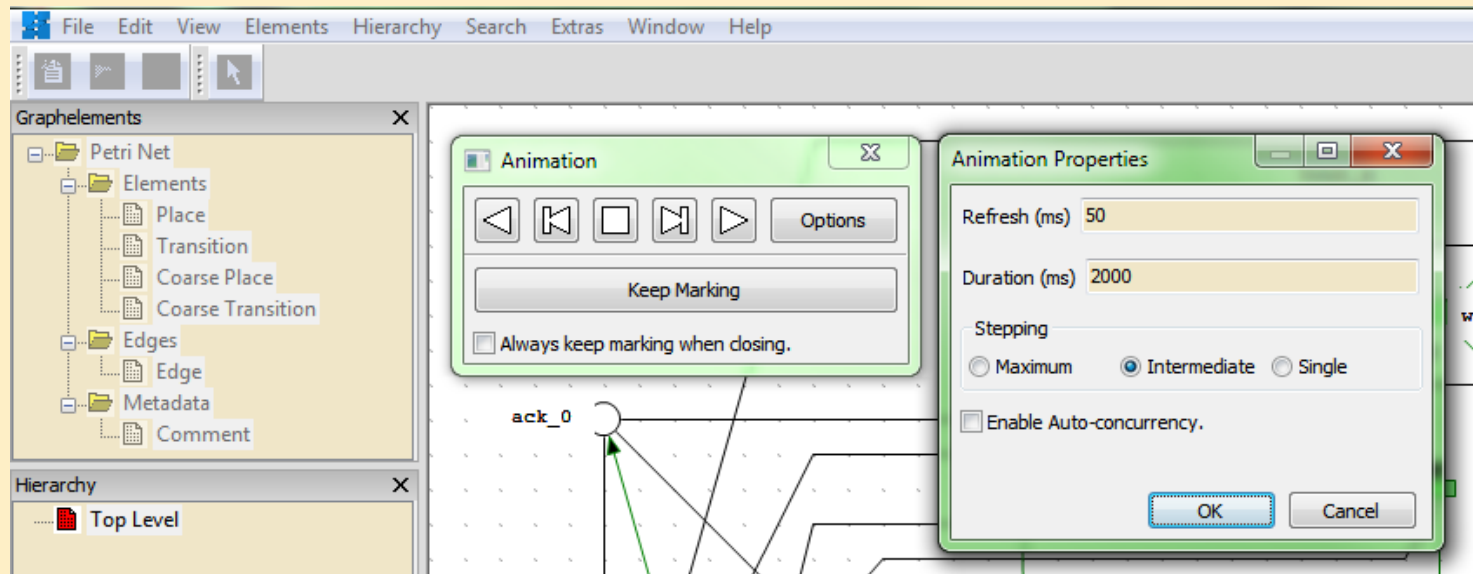
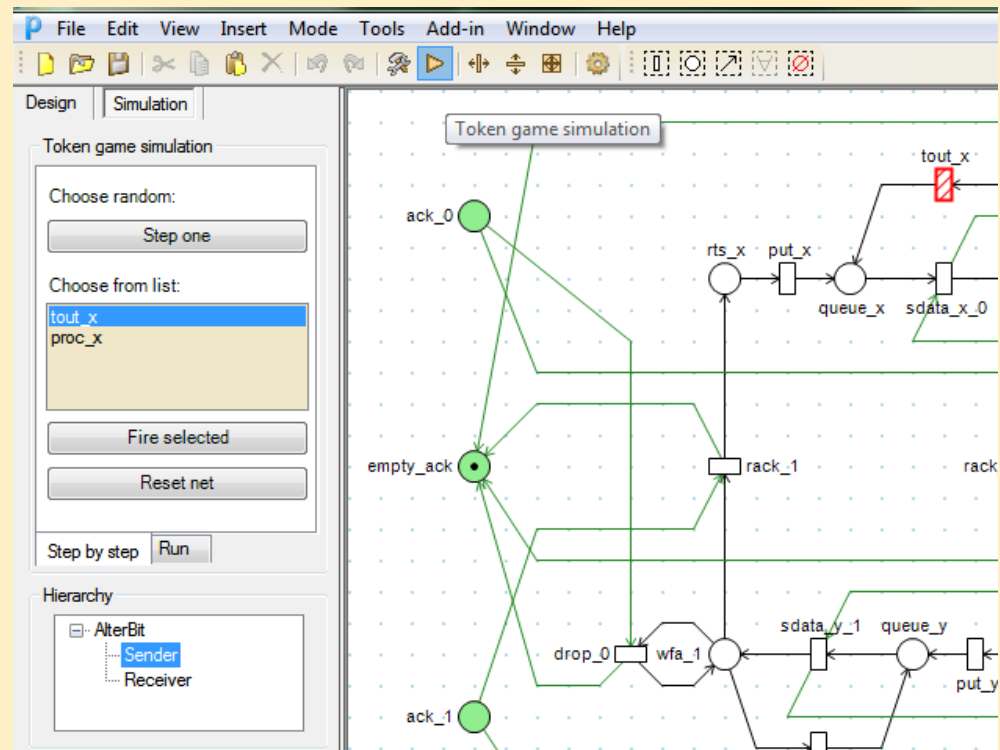
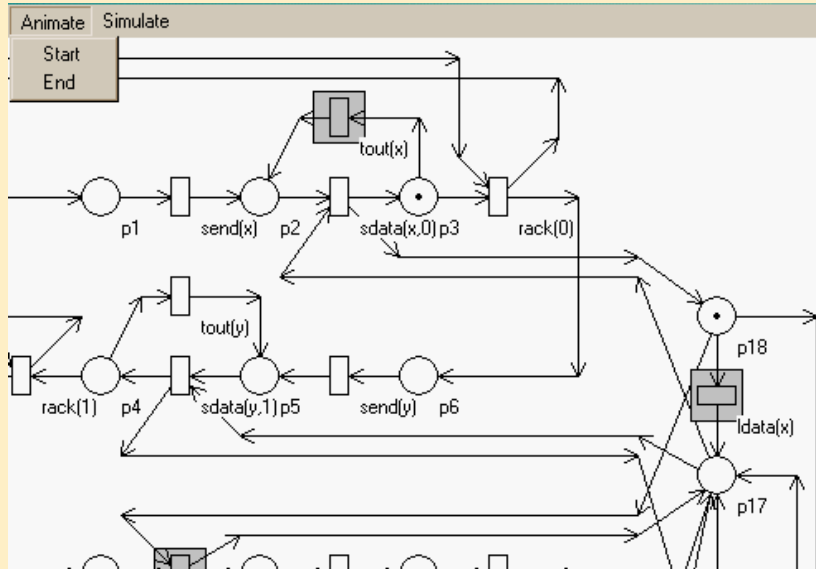
- A rendszer lehetséges trajektóriáinak vizsgálata
- Petri háló állapota: token eloszlás (jelölés)
 - állapotváltás = tüzelés
 - trajektóriák az állapottérben = tüzelési szekvenciák
- Petri háló nemdeterminisztikus
 - a nem-determinizmust is modellezni kell
 - valódi (ál-)véletlen generálásra van szükség
 - állapottér bejárása: interaktív szimuláció (választás)

Animáció (token game)



- A modell interaktív ellenőrzése
- Engedélyezett átmenetek jelölve (DNAet)
- Jelölt átmenetre kattintva tüzel
- Előállítja az új tokeneloszlást
- Konkurens átmenetek:
 - manuálisan
 - PetriDotNet, Snoopy: véletlen választással is
- Befejezéskor visszaállítja a kezdeti tokeneloszlást

Animációs képernyő



Szimuláció

Large scale statistics

Settings

Number of firings: 10 000

☒ Run from current state ☐ Keep ending state

☐ Run from initial state ☐ Show hierarchical names

Transitions

Transition	Firings	Percentage
put_x	137	1,37 %
drop_1	6	0,06 %
lose_0	413	4,13 %
put_y	137	1,37 %
rack_0	137	1,37 %

Places

Place	Avg token	Avg token in time
ack_0	0,337026793348499	---
empty_ack	0,585617646523382	---
ack_1	0,236643479018611	---
data_x	0,450335110909001	---
empty_data	0,506631204575518	---

Progress: 0,432 s

Run

OK

Simulation

Run Length:

Number of Firings 5000

Results:

☒ Tokens per Place ☒ Transition Throughput

☐ Produce Trace

OK Cancel

Egyszerű szimulációs algoritmus

```
while (true) do
```

```
    Engedélyezett tranzíciók felmérése
```

```
    if Van tüzelhető tranzíció?
```

```
        then Végrehajtandó kiválasztása (nemdeterminisztikus)
```

```
        else Szimuláció vége.
```

```
    Tüzelés
```

```
end while
```

Tüzelhető tranzíciók listájának összeállítása

```
function collect_fireable_transitions(M)  
  // Tüzelhető tranzíciók halmaza  
   $L_{\text{fireable}} \leftarrow \emptyset$   
  for all  $t \in T$  do  
    if enabled( $t, M$ ) then  $L_{\text{fireable}} \leftarrow L_{\text{fireable}} \cup \{t\}$   
  return  $L_{\text{fireable}}$   
end function
```

Az állapotváltozás nagysága

Ha t tüzel M állapotban

- Új állapot: $M' = M + \mathbf{W}^T \cdot e_t$
 - ahol e_t a t tranzíciónak megfelelő egységvektor
- Ahol $\mathbf{W}$ a súlyozott szomszédossági mátrix
 - $\mathbf{W} = [w(t, p)]$
 - Dimenziója: $\tau \times \pi = |T| \times |P|$
 - Ha t tüzel, mennyit változik a p -beli tokenszám:

$$w(t, p) = \begin{cases} w^+(t, p) - w^-(p, t) & \text{ha } (t, p) \in E \text{ vagy } (p, t) \in E \\ 0 & \text{ha } (t, p) \notin E \text{ és } (p, t) \notin E \end{cases}$$

Tüzelés

// Inicializálás

$M \leftarrow M_0$

$L_{\text{fireable}} \leftarrow \text{collect_fireable_transitions}(M)$

while $L_{\text{fireable}} \neq \emptyset$ **do**

 // Tüzelés

$t \leftarrow \text{rnd}(L_{\text{fireable}})$

$M' \leftarrow M + \mathbf{W}^T \cdot \underline{e}_t$

$L_{\text{fireable}} \leftarrow \text{collect_fireable_transitions}(M')$

$M \leftarrow M'$

end while

Redundancia

- Miért kellene mindig megnézni az összes tranzíció tüzelhetőségét ($|T|$ db vizsgálat), ha csak az éppen tüzelt tranzíció környezete ($\bullet t \cup t \bullet$) változik?
 - Letiltódó tranzíciók
 - Engedélyeződő tranzíciók

Letiltódó tüzelések

- Egy t tranzíció tüzelése során letiltódhat
 - olyan t' tranzíció, melynek bemenete kapcsolódik $\bullet t$ -hez
 - konfliktusban van t -vel: $\bullet t' \cap \bullet t \neq \emptyset$
- Numerikus meghatározás:
 - Az elvett tokenek száma: $M^- = \mathbf{W}^{-\mathbf{T}} \cdot \underline{e}_t$
 - t ősei (bemeneti helyei): $\bullet t \Rightarrow \{p \in P: M^-(p) > 0\}$
 - A t -vel konfliktusban levők: $T' = \{(\bullet t)\bullet\}$

Engedélyeződő tüzelések

- Egy t tranzíció tüzelése során engedélyeződhet
 - olyan t'' tranzíció, melynek bemenete kapcsolódik $t\bullet$ -hez
 - t engedélyezi t'' -t: $\bullet t'' \cap t\bullet \neq \emptyset$
- Numerikus meghatározás
 - A hozzáadott tokenek száma: $M^+ = \mathbf{W}^{+T} \cdot \underline{e}_t$
 - t utódai (kimeneti helyei): $t\bullet \Rightarrow \{p \in P: M^+(p) > 0\}$
 - A t által engedélyezettek: $T'' = \{(t\bullet)\bullet\}$
- Elég ezeket (letiltódó + engedélyeződő) tranzíciókat újraértékelni!

Hatékony algoritmus: inicializálás

- Inicializálási fázis ugyanaz

// Inicializálás

$M \leftarrow M_0$

$L_{fireable} \leftarrow \emptyset$

// Tüzelhető tranzíciók kezdeti halmaza

for all $t \in T$ **do**

if $enabled(t, M_0)$ **then** $L_{fireable} \leftarrow L_{fireable} \cup \{t\}$

Hatékony algoritmus: tüzelési ciklus

```
while  $L_{\text{fireable}} \neq \emptyset$  do  
  // Tüzelés  
   $t \leftarrow \text{rnd}(L_{\text{fireable}})$   
   $M' \leftarrow M + \mathbf{W}^T \cdot \underline{\mathbf{e}}_t$   
  // Letiltottak eltávolítása  
  for all  $t' \in \{(\bullet t)\bullet\}$  do  
    if  $\text{not}(\text{enabled}(t', M'))$  then  $L_{\text{fireable}} \leftarrow L_{\text{fireable}} \setminus \{t'\}$   
  // Engedélyezettek bevonása  
  for all  $t'' \in \{(t\bullet)\bullet\}$  do  
    if  $\text{enabled}(t'', M')$  then  $L_{\text{fireable}} \leftarrow L_{\text{fireable}} \cup \{t''\}$   
   $M \leftarrow M'$   
end while
```

Prioritás

- Tüzelési szabály kiegészül: t a.cs.a. tüzelhet, ha
 - engedélyezett és
 - nincs az ő $\pi(t)$ prioritásánál nagyobb prioritású engedélyezett
- Következmény:
 - $L_{fireable}$ nem halmaz, hanem halmazok $\pi \in \Pi$ prioritás szerint rendezett $L_{fireable}[\pi]$ vektora
 - tüzeléskor a legmagasabb prioritású nem üres $L_{fireable}[\pi]$ halmazból választunk nemdeterminisztikusan

Prioritásos algoritmus: inicializálás

// Inicializálás

$M \leftarrow M_0$

for all $\pi \in \Pi$ **do**

$L_{fireable}[\pi] \leftarrow \emptyset$

// Tüzelhető tranzíciók kezdeti halmaza

for all $t \in T$ **do**

if $enabled(t, M_0)$ **then** $L_{fireable}[\pi(t)] \leftarrow L_{fireable}[\pi(t)] \cup \{t\}$

Prioritásos algoritmus: tüzelési ciklus

```
while  $\bigcup_{\pi \in \Pi} L_{\text{fireable}}[\pi] \neq \emptyset$  do
  for  $\pi = \pi_{\max}$  to  $\pi_{\min}$  step -1 do // Tüzelés
    if  $L_{\text{fireable}}[\pi] \neq \emptyset$  then
       $t \leftarrow \text{rnd}(L_{\text{fireable}}[\pi])$ 
       $M' \leftarrow M + \mathbf{W}^T \cdot \underline{e}_t$ 
      exit for
    end if

  for all  $\pi \in \Pi$  do // Engedélyezett tranzíciók
    for all  $t' \in \{(\bullet t) \bullet\}$  do
      if not(enabled( $t'$ ,  $M'$ )) then  $L_{\text{fireable}}[\pi(t')]$   $\leftarrow L_{\text{fireable}}[\pi(t')] \setminus \{t'\}$ 
    for all  $t'' \in \{(t \bullet) \bullet\}$  do
      if enabled( $t''$ ,  $M'$ ) then  $L_{\text{fireable}}[\pi(t'')]$   $\leftarrow L_{\text{fireable}}[\pi(t'')] \cup \{t''\}$ 
    end for
  end for

   $M \leftarrow M'$ 
end while
```

Időzítés

- „Fizikai” időfogalom bevezetése a modellezésbe
 - Események (tüzelések) adott idő alatt mennek végbe
- Lehetőségek:
 - nem időzített tranzíciók
 - időzített tranzíciók
- Tranzíciók időzítése
 - determinisztikus időzítés: konstans gyújtási idő
 - véletlen időzítés: a gyújtási idő valószínűségi változó
 - pl. exponenciális eloszlású: SPN (Stochastic Petri Net)
 - Markov folyamat, analízis Markov láncok elmélete alapján

Időzített Petri hálók működése

- Tüzelés fázisai:

1. Tüzelhetőség kiértékelése

2. Időzítés

- Véletlen időzítésű engedélyezett tranzíciók sorsolnak
- Legkisebb gyújtási idejű tranzíció(k egyike) tüzel

3. Tüzelés végrehajtása

- Logikai idő léptetése

Megoldandó problémák

- Időzített vs. nem időzített?
 - Megoldás: időzítetlen tranzíciók magas prioritásúak
 - Azonnali (immediate) tranzíció
- Mi van, ha több, konfliktusban levő időzített tranzíció egyike tüzel és letiltja mások tüzelését?
 - Alternatív definíciók:
 - időzítés újraindul
 - letelt idő megőrződik (reward saving)
 - SPN-nél statisztikai értelemben mindegy
 - exponenciális eloszlás örökifjú tulajdonsága miatt
 - $P(\xi \geq x+y \mid \xi \geq x) = P(\xi \geq y)$

Algoritmus: újrainduló időzítés

procedure *maintenance*(t, M)

// Azonnali tranzíciók karbantartása

for all $\pi \in \Pi$ **do**

for all $t' \in \{(\bullet t) \bullet\} \cap T_{\text{immediate}}$ **do**

if *not*(*enabled*(t', M)) **then** $L_{\text{immediate}}[\pi(t')] \leftarrow L_{\text{immediate}}[\pi(t')] \setminus \{t'\}$

for all $t'' \in \{(t \bullet) \bullet\} \cap T_{\text{immediate}}$ **do**

if *enabled*(t'', M) **then** $L_{\text{immediate}}[\pi(t'')] \leftarrow L_{\text{immediate}}[\pi(t'')] \cup \{t''\}$

end for

...

Időzített tranzíciók kezelése

...

// futó időzítésű letiltott tranzíciók kivétele

for all $\tau > \tau_{simulation}$ **do**

for all $t' \in \{(\bullet t)\bullet\} \cap T_{timed} \cap L_{timed}[\tau]$ **do**

if $\text{not}(\text{enabled}(t', M))$ **then** $L_{timed}[\tau] \leftarrow L_{timed}[\tau] \setminus \{t'\}$

// új engedélyezett időzített tranzíciók felvétele

for all $t'' \in \{(t\bullet)\bullet\} \cap T_{timed}$ **do**

$\tau'' \leftarrow \text{time}(t'')$

if $\text{enabled}(t'', M)$ **then** $L_{timed}[\tau''] \leftarrow L_{timed}[\tau''] \cup \{t''\}$

end for

end procedure

Főprogram: inicializálás

// Inicializálás

$M \leftarrow M_0$

$\tau_{simulation} \leftarrow 0$

for all $\pi \in \Pi$ **do**

$L_{immediate}[\pi] \leftarrow \emptyset$

// Tüzelhető azonnali tranzíciók kezdeti halmaza

for all $t \in T_{immediate}$ **do**

if $enabled(t, M_0)$ **then** $L_{immediate}[\pi(t)] \leftarrow L_{immediate}[\pi(t)] \cup \{t\}$

...

Főprogram: időzítetlen tranzíciók tüzelése

```
MAIN: while  $\bigcup_{\pi \in \Pi} L_{\text{immediate}}[\pi] \neq \emptyset$  do
  for  $\pi = \pi_{\text{max}}$  to  $\pi_{\text{min}}$  step -1 do
    if  $L_{\text{immediate}}[\pi] \neq \emptyset$  then
      // Azonnali tranzíció tüzelése
       $t \leftarrow \text{rnd}(L_{\text{immediate}}[\pi])$ 
       $M' \leftarrow M + \mathbf{W}^T \cdot \underline{e}_t$ 
       $\text{maintenance}(t, M')$ 
       $M \leftarrow M'$ 
    exit for
  end if
end for
end while
...
```

Főprogram: időzített tranzíciók tüzelése

...

if $\exists \tau \geq \tau_{simulation} : L_{timed}[\tau] \neq \emptyset$ **then**

// Idő léptetése

$\tau_{simulation} \leftarrow \min(\{\tau : L_{timed}[\tau] \neq \emptyset\})$

// Időzített tranzíció tüzelése

$t \leftarrow rnd(L_{timed}[\tau_{simulation}])$

$M' \leftarrow M + \mathbf{W}^T \cdot \underline{e}_t$

$\text{maintenance}(t, M')$

$M \leftarrow M'$

goto MAIN

end if

end.