

# Petri-hálóok analízise

dr. Bartha Tamás

dr. Majzik István

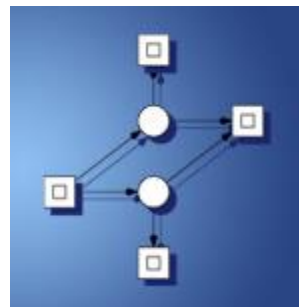
dr. Pataricza András

BME Méréstechnika és Információs Rendszerek Tanszék

# Modellező és analízis eszközök: DNAnet, Snoopy, PetriDotNet



UNIVERSITY OF CAPE TOWN  
Department of Computer Science



b.tu

Brandenburgische  
Technische Universität  
Cottbus

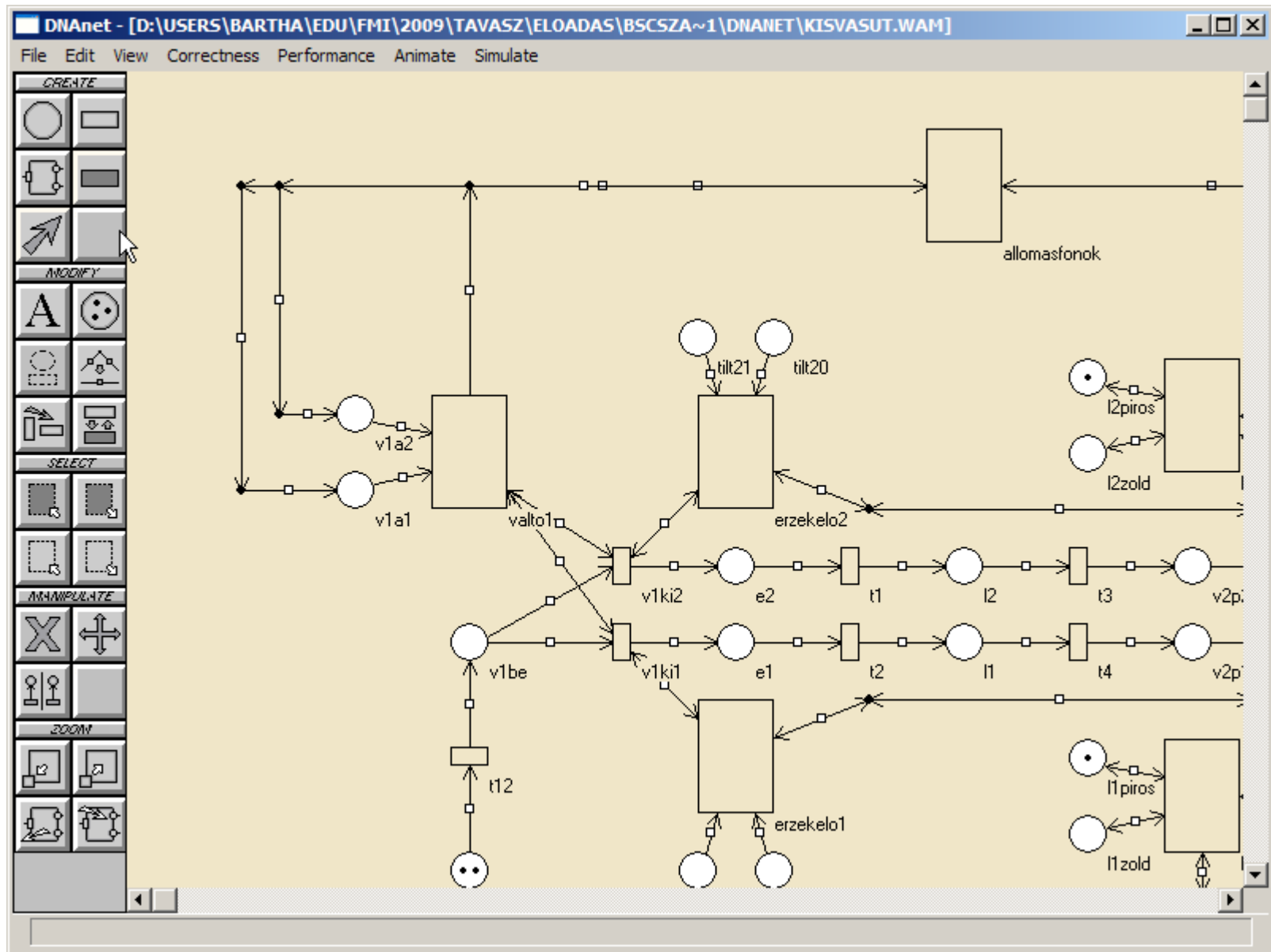
petridotnet  
from BME-MIT



# A DNAnet modellező program

- **Képességei**
  - Grafikus szerkesztő (régi Windowsra)
  - Interaktív animáció (**token game**)
  - Nem interaktív szimuláció (teljesítmény analízishez)
  - Analízisek: egyes **dinamikus és statikus tulajdonságok** ellenőrzése
- **Előnyei**
  - Kicsi, kompakt, gyors, egyszerűen kezelhető
  - Méretéhez képest sokat tud
  - Ingyenes, szabad felhasználású
- **Hátránya**
  - Stabilitás, új Windowson futtatás problémás

# A DNAnet modellező program képe



# A DNAnet analízis képességei

Net is bounded.

Deadlock is not possible.

Net is live.

Net has home states.

Coverability graph generation statistics:

108 unique markings

1 strongly connected components

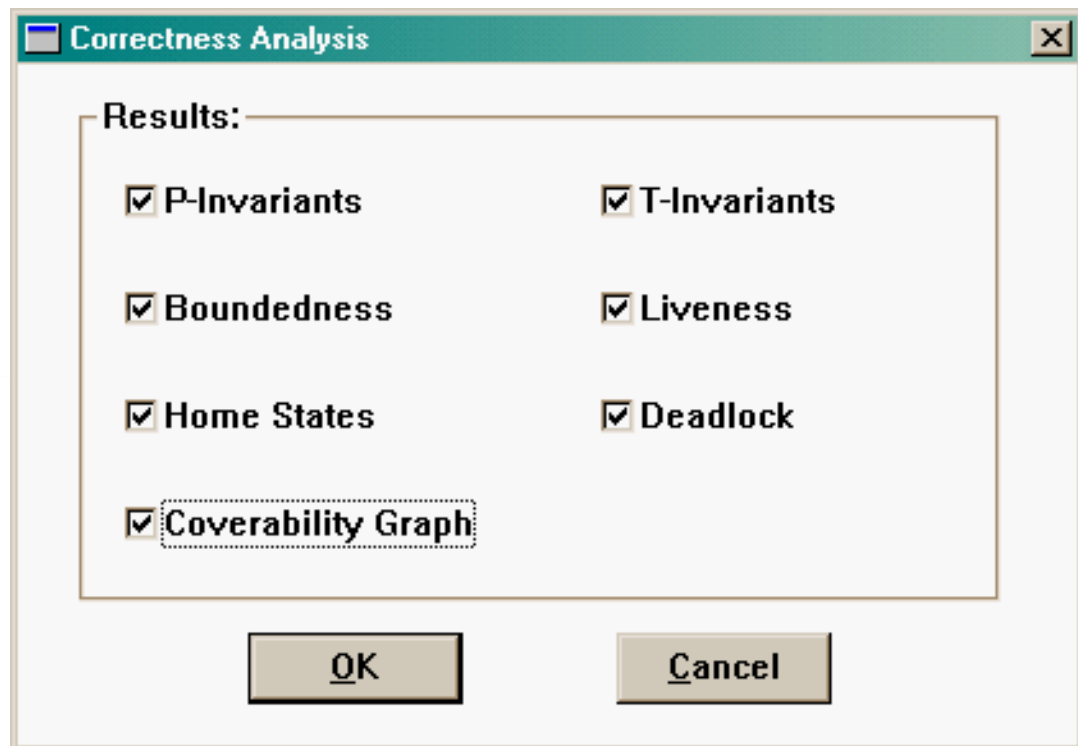
108 hash table entries used

1 was the longest hash list length

1 was the average hash list length

27 was the maximum stack height

3 was the maximum component stack height



# A DNAnet invariáns keresése

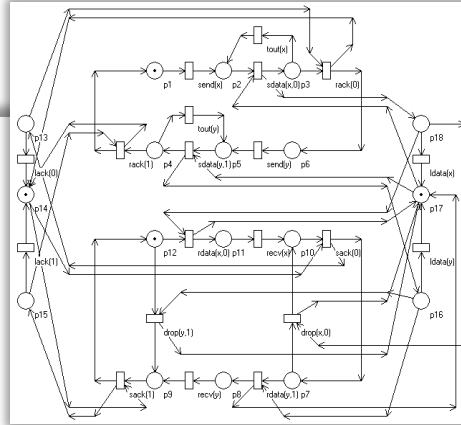
P-invariants:

1	0	0	0		(main.p1)
1	0	0	0		(main.p2)
1	0	0	0		(main.p3)
1	0	0	0		(main.p4)
1	0	0	0		(main.p5)
1	0	0	0		(main.p6)
0	1	0	0		(main.p7)
0	1	0	0		(main.p8)
0	1	0	0		(main.p9)
0	1	0	0		(main.p10)
0	1	0	0		(main.p11)
0	1	0	0		(main.p12)
0	0	1	0		(main.p13)
0	0	1	0		(main.p14)
0	0	1	0		(main.p15)
0	0	0	1		(main.p16)
0	0	0	1		(main.p17)
0	0	0	1		(main.p18)

ie.

$M(\text{main.p1}) + M(\text{main.p2}) + M(\text{main.p3}) + M(\text{main.p4}) + M(\text{main.p5}) + M(\text{main.p6})$   
 $M(\text{main.p7}) + M(\text{main.p8}) + M(\text{main.p9}) + M(\text{main.p10}) + M(\text{main.p11}) + M(\text{main.p12})$   
 $M(\text{main.p13}) + M(\text{main.p14}) + M(\text{main.p15})$   
 $M(\text{main.p16}) + M(\text{main.p17}) + M(\text{main.p18})$

All places are covered by P-invariants.



T invariants:

1	0	0	0	0	0	1		(main.send(x))
1	0	1	1	1	0	1		(main.sdata(x,0))
1	0	0	0	0	0	1		(main.rack(0))
1	0	0	0	0	0	1		(main.send(y))
1	1	0	0	1	1	1		(main.sdata(y,1))
1	0	0	0	0	0	1		(main.rack(1))
0	0	1	1	1	0	0		(main.tout(x))
0	1	0	0	1	1	0		(main.tout(y))
1	0	0	0	1	1	1		(main.sack(1))
1	0	0	0	1	0	0		(main.recv(y))
1	0	0	0	1	0	0		(main.rdata(y,1))
1	0	0	1	1	0	1		(main.sack(0))
1	0	0	0	1	0	0		(main.recv(x))
1	0	0	0	1	0	0		(main.rdata(x,0))
0	0	0	1	1	0	0		(main.lack(0))
0	0	0	0	1	1	0		(main.lack(1))
0	1	0	0	0	0	0		(main.ldata(y))
0	0	1	0	0	0	0		(main.ldata(x))
0	0	0	1	0	0	1		(main.drop(x,0))
0	0	0	0	0	1	1		(main.drop(y,1))

# A Snoopy modellező program

- Snoopy (Windows, Linux)
  - Grafikus szerkesztő + token game (animált)
  - Egyszerűen kezelhető
  - Kényelmi funkciók: copy / paste, undo / redo
  - **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
  - Külső analízis eszköz: Charlie (Java)

# A Snoopy modellező program képe

The image shows the Snoopy 2.0 software interface. The main window displays a Petri net diagram titled "Jatekautomata.spped". The diagram consists of several places and transitions:

- nyer**: A green place containing 2 tokens.
- veszi t**: A red place containing 1 token.
- uzemben**: A white place containing 1 token.
- zseton**: A white place containing 2 tokens.
- jatekban**: A white place containing 1 token.

Transitions are represented by small circles with arrows indicating the flow of tokens between places. A transition between "nyer" and "zseton" is labeled with the number "2".

An "About..." dialog box is open in the foreground, displaying the following information:

**Snoopy 2.0**  
Snoopy, a extensible, adaptive and plattform independent Editor to design and animate/simulate hierarchical graphs.

Vendor: Brandenburg University of Technology Cottbus  
Data Structures and Software Dependability Group

Version: 1.03  
Release: stable  
Build: Feb 18 2011

Contributors: Denny Bayer, Matthias Dube, Markus Fieber, Monika Heiner, Mostafa Herajy, Erik Jongsma, Christian Krueger, Anja Kurth, Sebastian Lehrack, Fei Liu, Thomas Meier, Ronny Richter, Christian Rohr, Daniel Scheibler, Martin Schwarick, Alexey Tovchigrechko, Katja Winder

<http://www-dssz.informatik.tu-cottbus.de/software/snoopy.html>

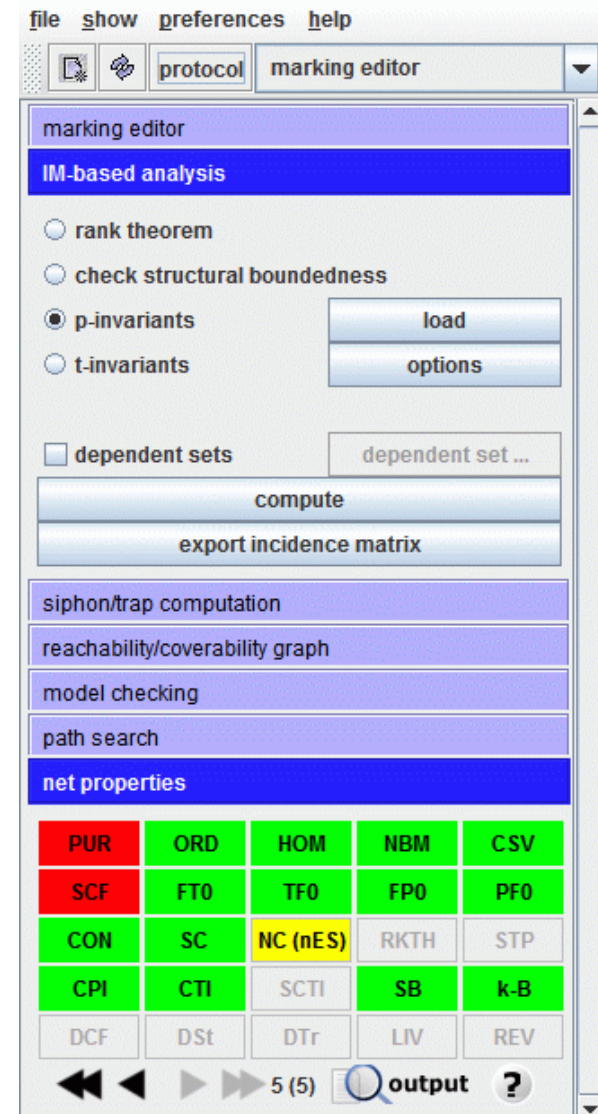
Any comments to:  
[snoopy@informatik.tu-cottbus.de](mailto:snoopy@informatik.tu-cottbus.de)

OK



# Analízis eszközök Snoopy-hoz

- Charlie (Java)
  - Dinamikus tulajdonságok, elérhetőség
  - Strukturális tulajdonságok, invariánsok
  - CTL és LTL modellellenőrző
- INA (Windows, Linux)
  - Szöveges felületű **parancssori** program
  - Invariáns analízis, elérhetőségi gráf generálás
  - Strukturális tulajdonságok ellenőrzése
  - Szimulációs képességei nincsenek



# A PetriDotNet modellező program

- Alapképességei
  - Grafikus szerkesztő + token game + szimuláció
  - **Kiterjesztések:** tiltó él, időzítés, színezett háló
  - Szabványos PNML fájlformátum, van hozzá INA kimenet
  - Támogatja **hierarchikus** Petri-hálók készítését
- Analízis képességei (modulárisan bővíthető)
  - Dinamikus és strukturális tulajdonságok
  - **CTL modellellenőrző**
  - Elérhetőségi és fedési gráf
- BME MIT fejlesztés: [petridotnet.inf.mit.bme.hu](http://petridotnet.inf.mit.bme.hu)

# A Petri.NET modellező program képe

The image shows the PetriDotNet software interface. The main window displays a Petri net diagram with the following components:

- Places:** zseton (top, 2 tokens), nyer (left, 1 token), veszit (middle, 1 token), jatekban (bottom, 1 token), indul (right, 1 token).
- Transitions:** uzemben (center).
- Edges:** Directed edges connect the places and transitions. A weight of 2 is shown on the edge from nyer to uzemben.

The interface includes a menu bar (File, Edit, View, Insert, Mode, Tools, Add-in, Window, Help), a toolbar, and a left sidebar with a **Design** tab. The sidebar contains a **Toolbox** with buttons for Select, Place, Transition, Edge, and Token, and a **Properties** section for the selected element (Net 1).

An **About PetriDotNet** dialog box is open in the foreground, displaying the following information:

**petridotnet**  
from BME-MIT

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](mailto:petridotnet@inf.mit.bme.hu).

Buttons: Send error feedback, OK

Footer: MŰEGYETEM 1782 logo and F T S RG icons.

# A PetriDotNet analízis képességei

## A(z) AlterBit háló tulajdonságai

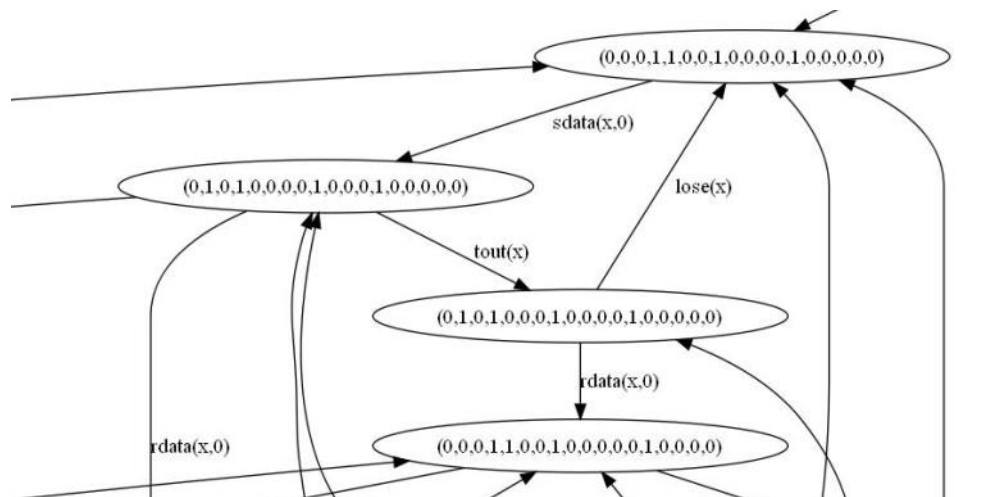
### Dinamikus tulajdonságok

Állapotok száma:	108
Korlátosság:	<b>korlátos</b>
	1-korlátos (biztos háló)
Holtpontmentesség:	<b>holtpontmentes</b>
Megfordíthatóság:	<b>megfordítható</b>
Perszisztencia:	<b>nem perzisztens</b>

### Strukturális tulajdonságok

Legszűkebb alosztály:	Petri-háló
Tisztaság:	<b>nem tiszta (van hurokél)</b>

[Elérhetőség vizsgálata:](#) [CTL-kifejezés vizsgálata:](#)  
[Elérhetőségi gráf mentése:](#) [Szomszédossági mátrix mentése:](#)  
[T-invariánsok keresése:](#) [P-invariánsok keresése:](#)  
[Helyek tokenkorlátjainak kiírása:](#)



The screenshot shows the CTL Expression Editor window. The expression being entered is  $AF(AlterBit.wfa\_0 > 0 \& EX(AlterBit.buffer\_x > 0))$ . Below the editor, a small window displays the verification results:

```
CTL MODEL CHECKING
Expression: AF(AlterBit.wfa_0>0&EX(AlterBit.buffer_x>0))
Model: AlterBit
Result: True
Runtime: 0,01 s
```

# A PetriDotNet invariáns analízis

The screenshot displays the PetriDotNet application interface with three windows open:

- Háló tulajdonságai**: Contains two **ShowInvariants** windows. The first shows the expression `{lose(x), sdata(x,0), tout(x)}` and the second shows `{ack_0, ack_1, empty(ack)}`.
- P-Invariants**: A dialog box showing the results of P-invariant calculation. The text reads: "List of P-Invariants calculated by Martinez-Silva algorithm", "Calculation finished in 0,00 ms. (places=18, transitions=22)", and lists the invariants: `{ack_0, ack_1, empty(ack)}`, `{data_x, empty(data), data_y}`, `{rts_x, queue_x, wfa_0, rts_y, wfa_1, queue_y}`, and `{wait_0, buffer_x, ok_x, ok_y, buffer_y, wait_1}`. An **OK** button is at the bottom.
- T-Invariants**: A larger dialog box showing the results of T-invariant calculation. The text reads: "List of T-Invariants calculated by Martinez-Silva algorithm", "Calculation finished in 15,60 ms. (places=18, transitions=22)", and lists 22 invariants, each enclosed in curly braces and separated by line breaks. An **OK** button is at the bottom.

At the bottom left of the main window, there are several links: [Elérhetőségek](#), [Elérhetőségi oldal mentése](#), [Szomszédossági matrix mentése](#), [T-invariánsok keresése](#), [P-invariánsok keresése](#), and [Helyek tokenkorlátjainak kiírása](#).

Példa modell:  
Az alternáló bit protokoll

# A modellezési feladat

## Alternating Bit Protocol

- Átviteli protokoll veszteséges csatornához
  - Üzenet **elveszhet** (véges számú alkalommal)
  - Üzenet tartalma nem változhat (ha változik, az detektálható és az üzenet elveszettnek tekinthető)
- Cél: a protokoll biztosítsa, hogy minden üzenet (véges számú próbálkozással) eljusson a vevőhöz

# Küldő folyamat

- A küldő az üzenetekhez egy ellenőrző bitet kapcsol
- Az üzenetek megérkezését a vevőtől nyugta jelzi, ugyanazzal az ellenőrző bittel
- Hibakezelés a küldőnél
  - Ha a küldő időtúllepéssel észleli a nyugta hiányát → újra küldi az üzenetet
  - Ha a küldő egy  $b$  bittel ellátott nyugtát vár és ilyet kap, akkor a következő üzenethez  $\neg b$  bitet csatol
  - Ha a küldő egy  $b$  bittel ellátott nyugtát vár de nem ilyet kap, akkor egyszerűen eldobja a nyugtát (majd időtúllépés lesz a nyugta hiánya miatt)



# Fogadó folyamat

- Az üzenet vételét nyugtázza a kapott ellenőrző bitet visszaküldve a nyugtával
- Hibakezelés a fogadónál:
  - Ha egy üzenetben az előző üzenethez képest nem negált ellenőrző bit érkezik, akkor
    - az üzenetet eldobja (nem dolgozza fel),
    - nyugtát küld a bejött ellenőrző bittel (arra számítva, hogy bizonyára az üzenet újraküldése történt az előző nyugta elvesztése miatt)

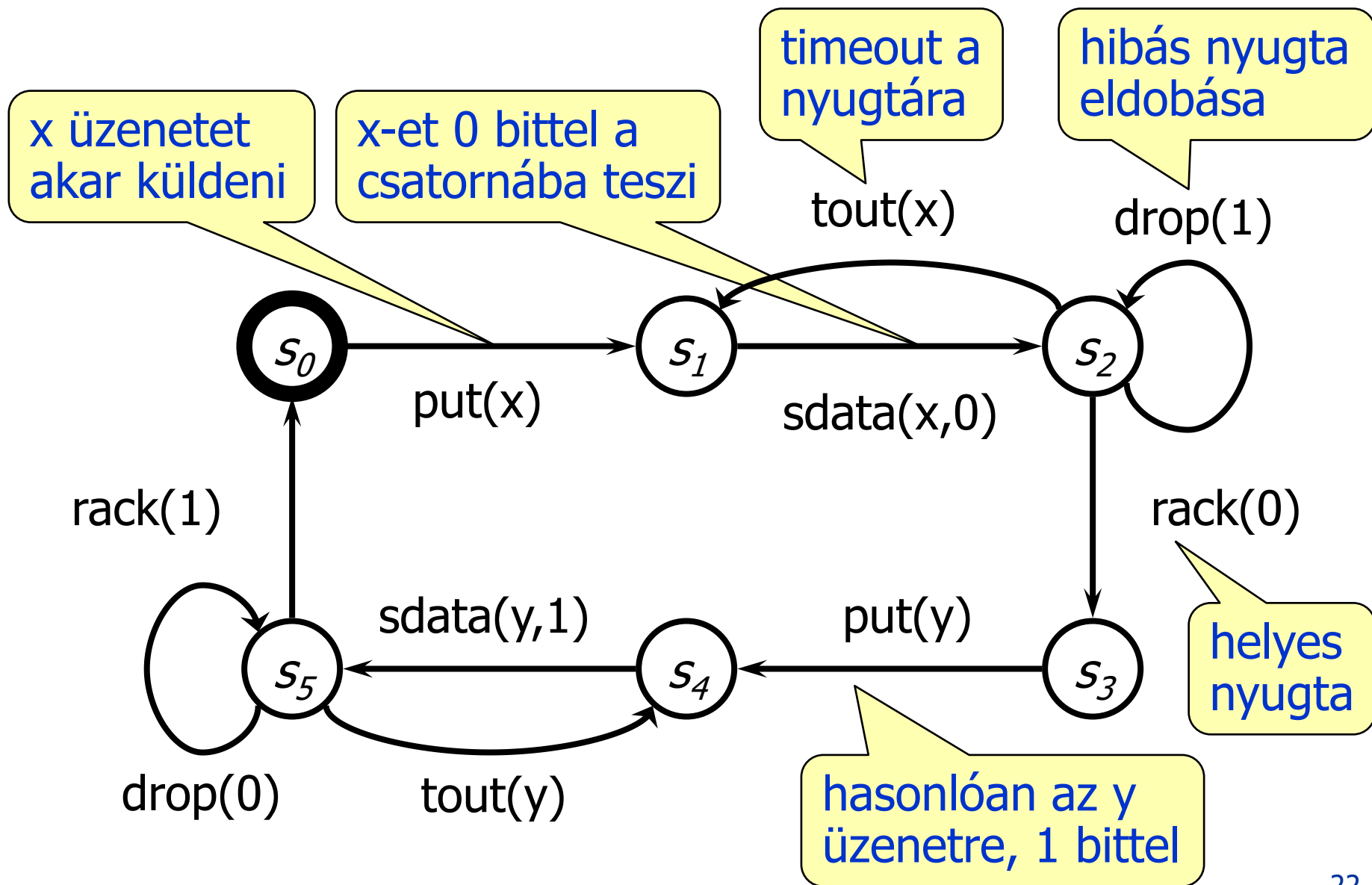
# A modellalkotás lépései

1. A feladat felbontása folyamatokra és erőforrásokra
2. Folyamatok állapotainak meghatározása
3. Erőforrások állapotainak meghatározása
4. Állapot alapú modellekből Petri-háló modell készítése
5. Folyamatok és erőforrások modelljeinek integrálása
  - Folyamat lépése módosítja az erőforrás állapotát
  - Erőforrás állapota feltétel egy folyamat lépéséhez
6. Integrált modell helyességének ellenőrzése
7. Modell felhasználása a feladat megoldására

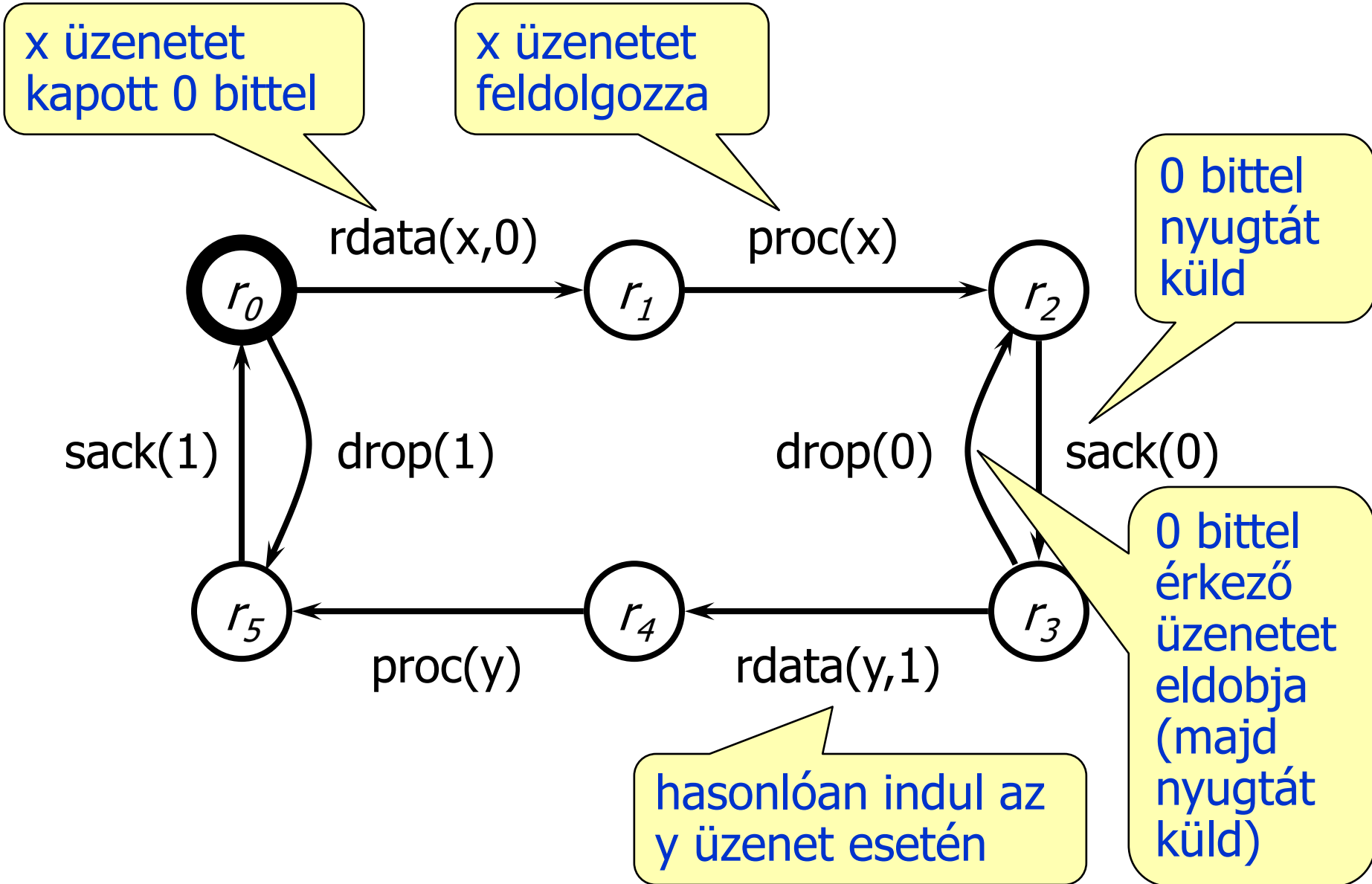
# Komponensek és állapotaik

- **Komponensek (alrendszerek)**
  - Folyamatok: küldő folyamat, fogadó folyamat
  - Erőforrások: adat csatorna, nyugtázó csatorna
- **Minden komponens saját állapotokkal rendelkezik**
  - Informális állapotgráf: állapotok körökkel, állapotátmenetek az eseményekkel címkézett nyilakkal
- **Azonos események egy időben mennek végbe: szinkronizáció**
  - Pl. a fogadó megkapja az üzenetet és ez egyúttal a csatornából is kikerül

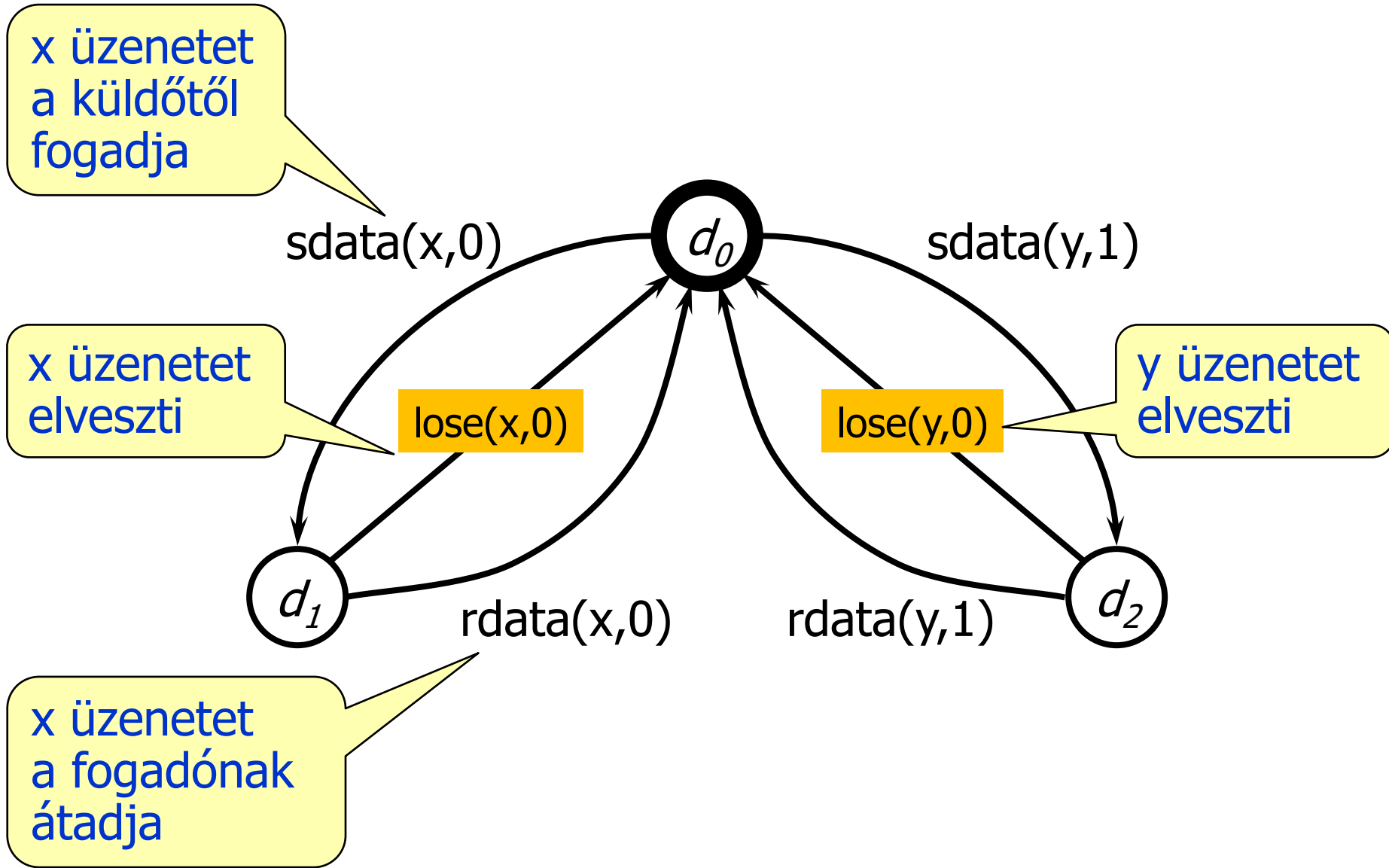
# Küldő folyamat állapotgráfja



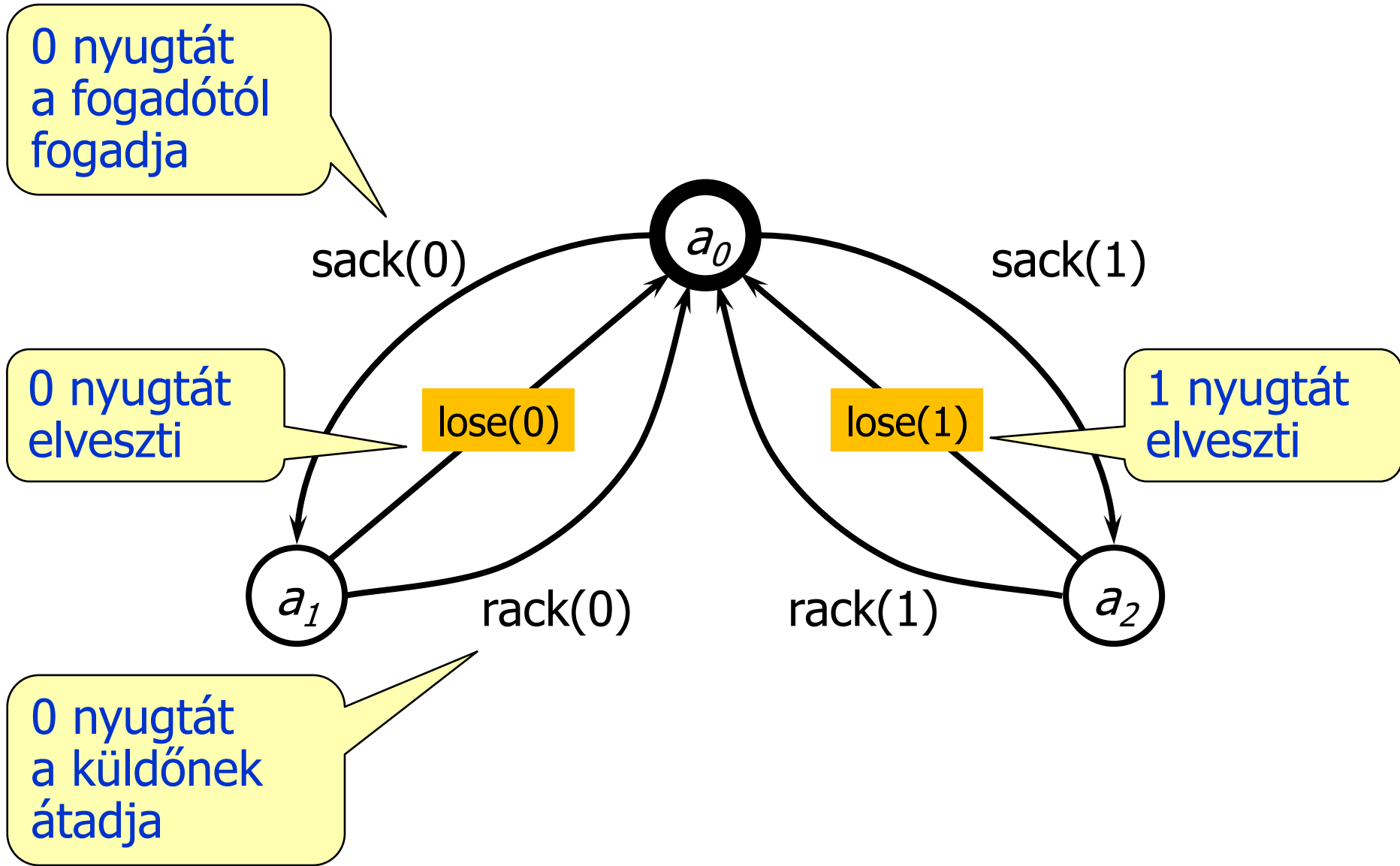
# Fogadó folyamat állapotgráfja



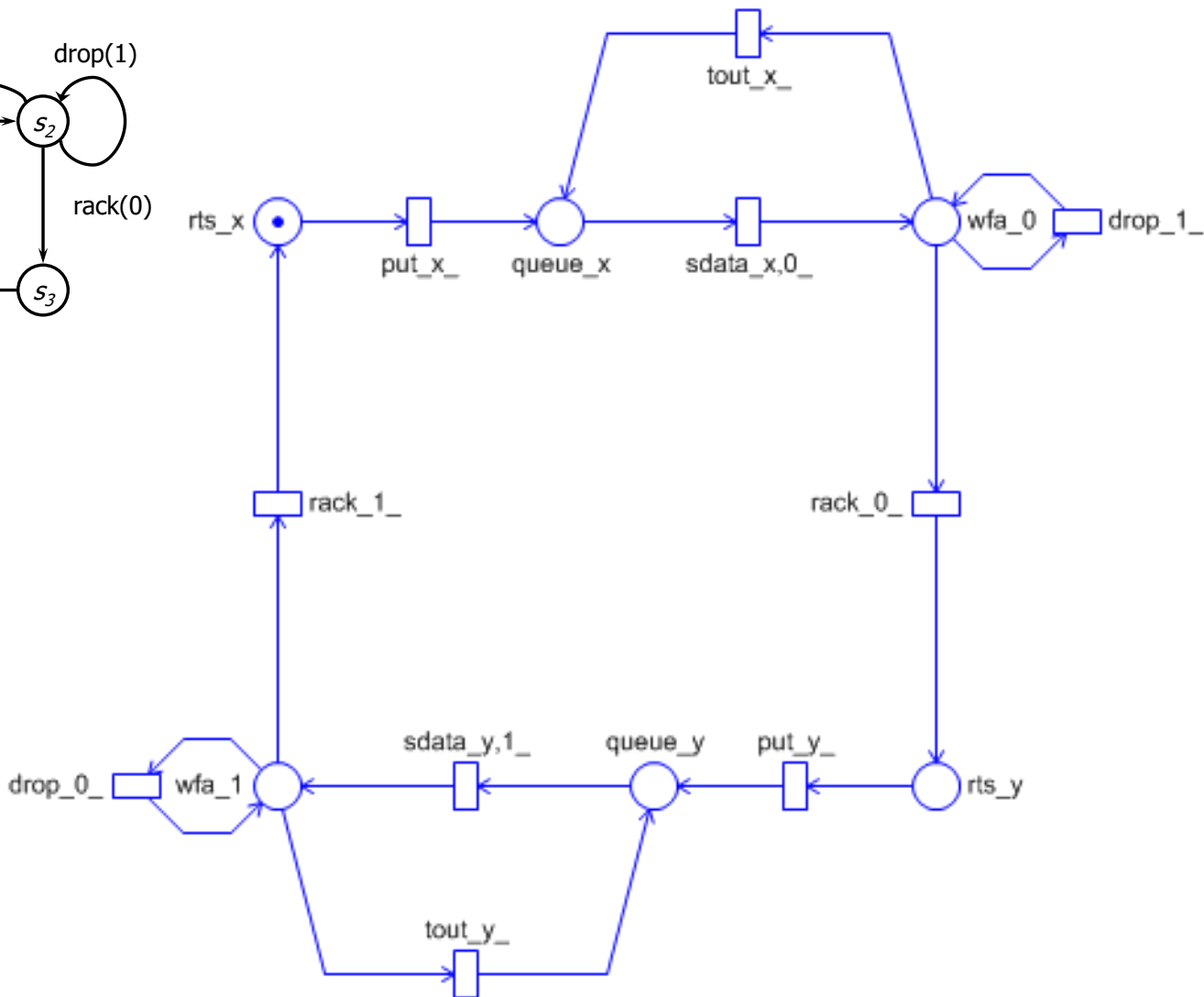
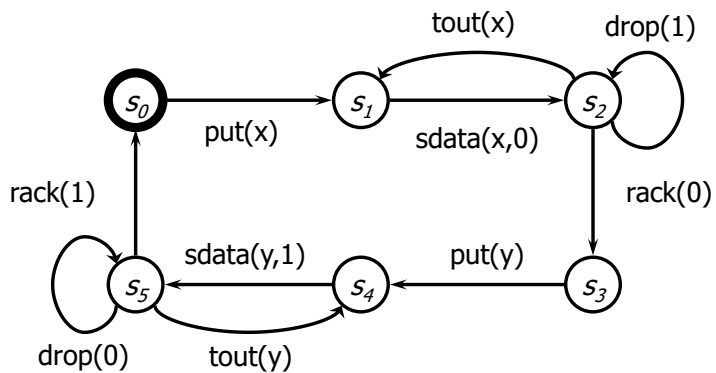
# Adat csatorna állapotgráfja



# Nyugtázó csatorna állapotgráfja



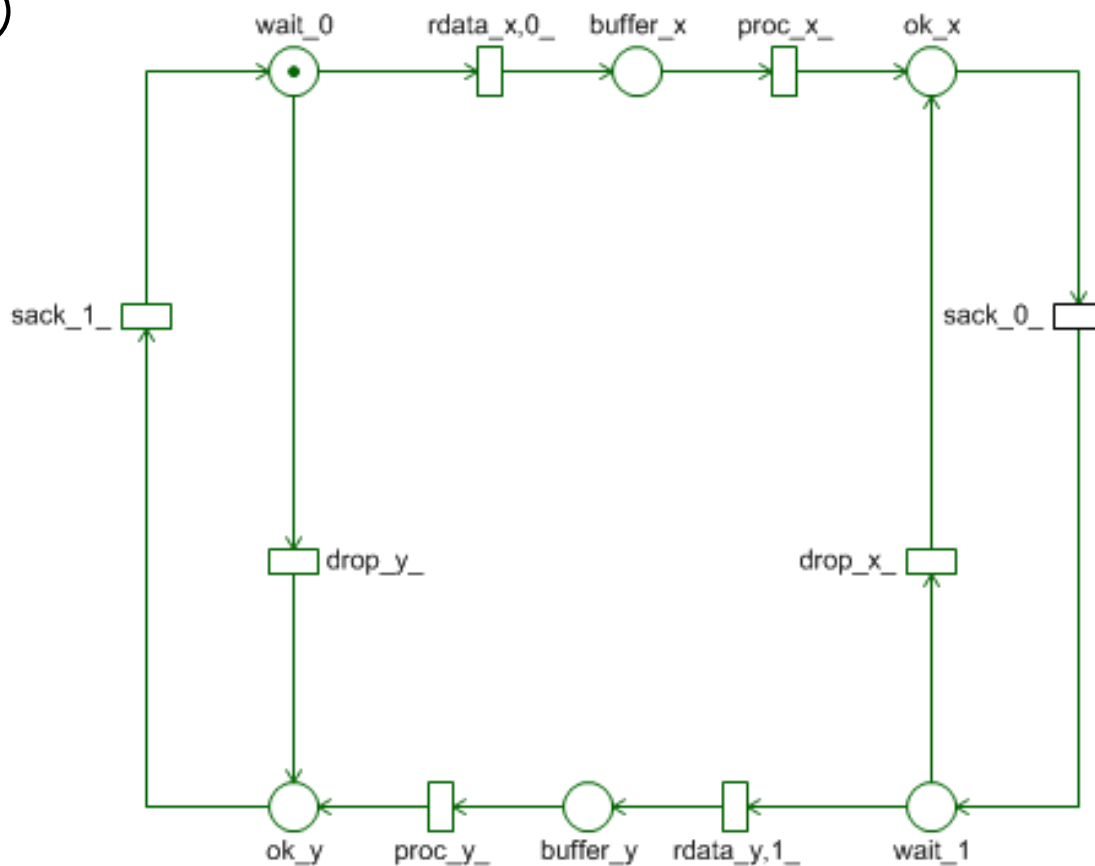
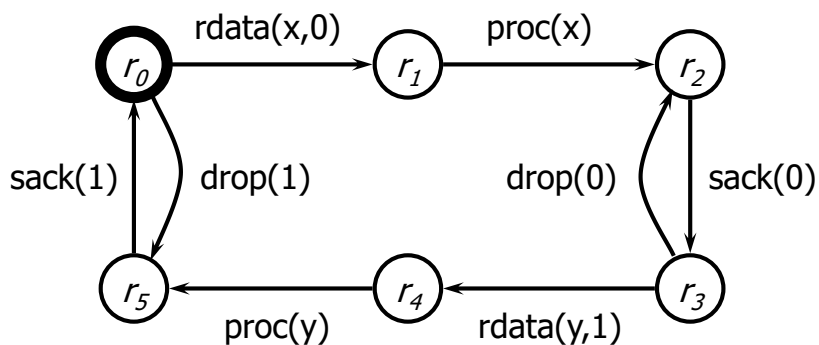
# Küldő folyamat Petri-háló modellje



Zárójelek helyett ' ' van a modellben



# Fogadó folyamat Petri-háló modellje



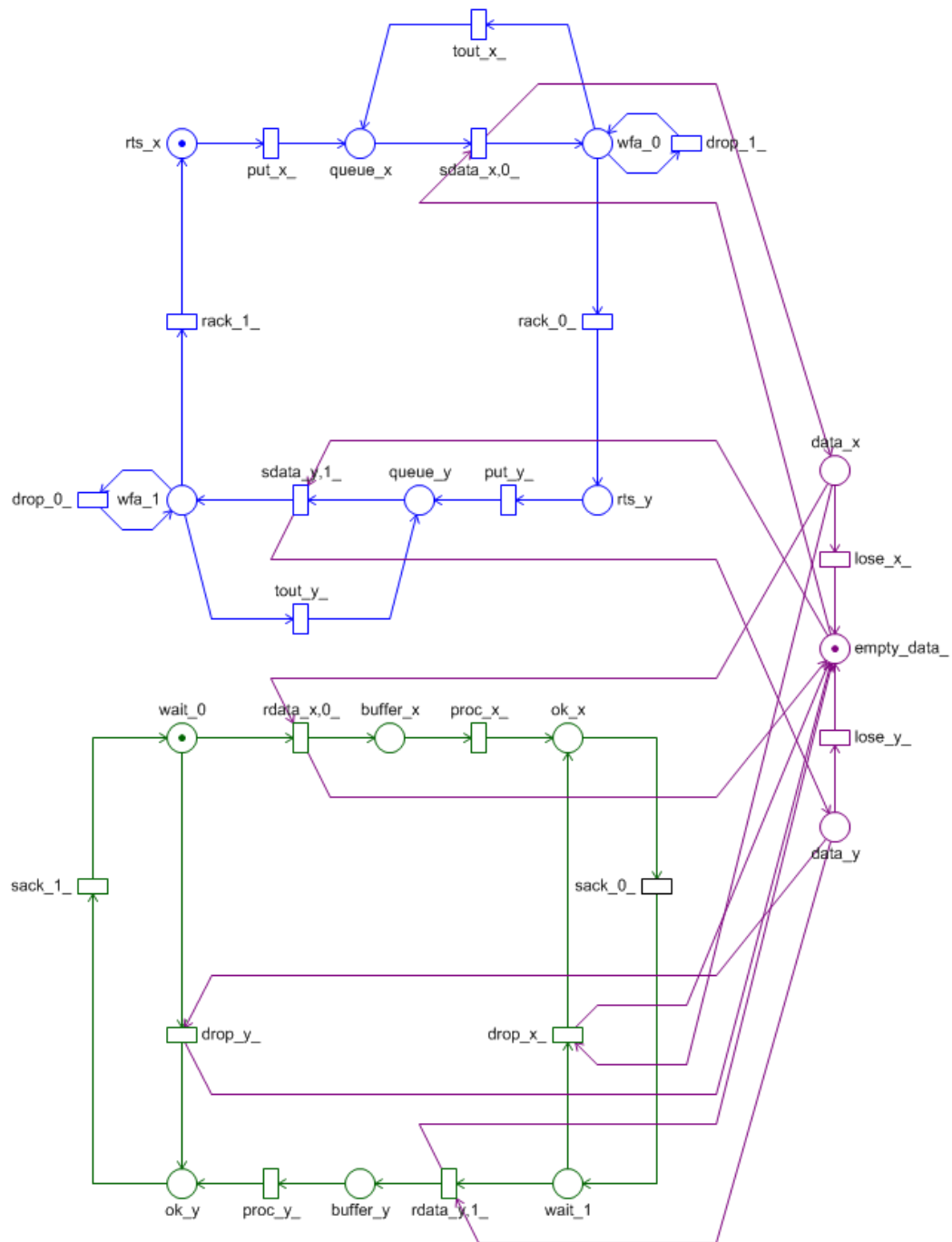
# Adat csatorna és adatátvitel

Üzenet normál  
fogadása és  
átadása a  
folyamatok  
megfelelő  
tranzícióival együtt  
(élekkkel bekötve a  
küldőtől illetve  
fogadótól):

- sdata()
- rdata()
- drop()

Üzenetvesztés:

- lose()



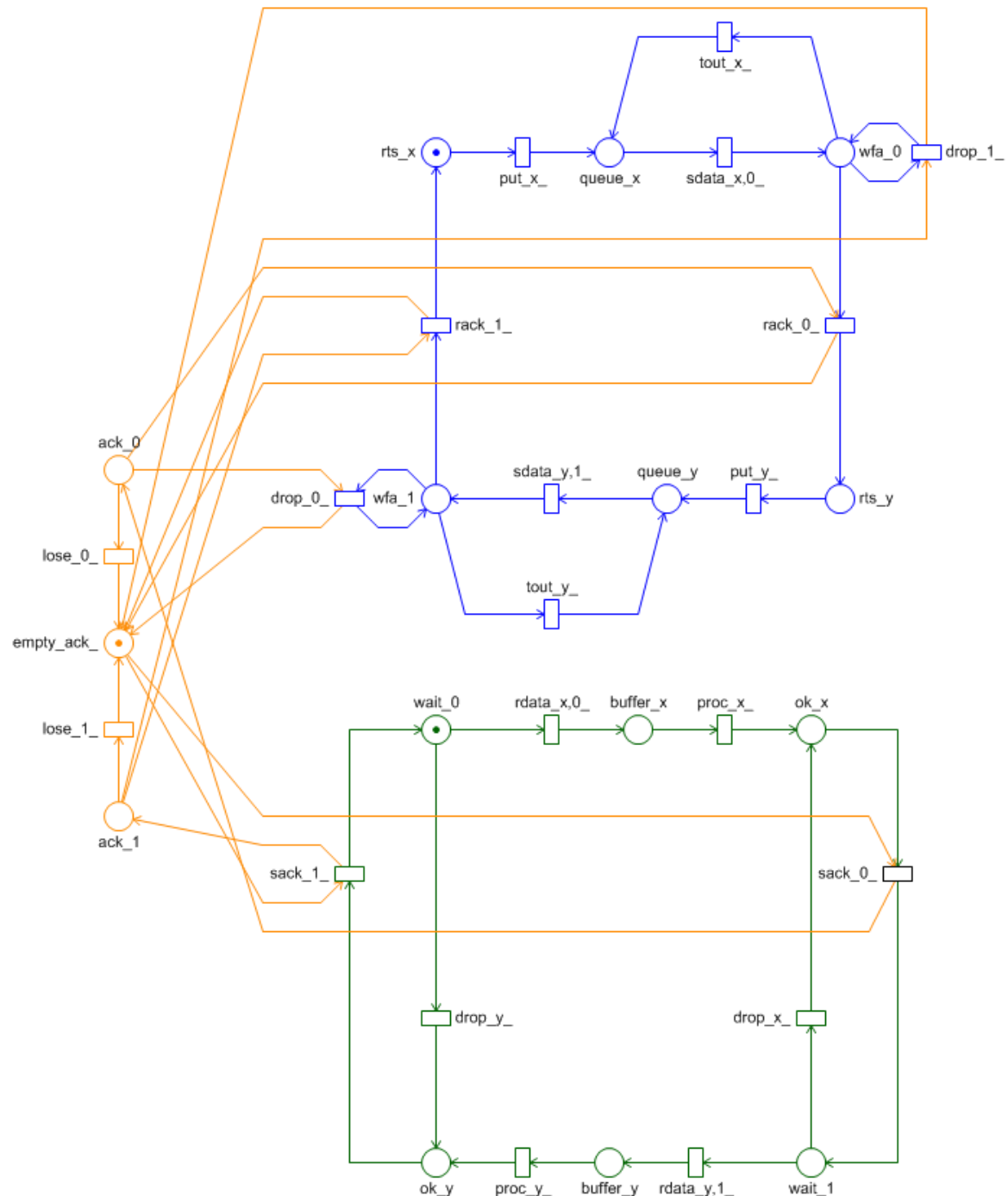
# Nyugtázó csatorna és nyugtázás

Nyugta normál fogadása és átadása a folyamatok megfelelő tranzícióival együtt (élekkkel bekötve a küldőtől illetve fogadótól):

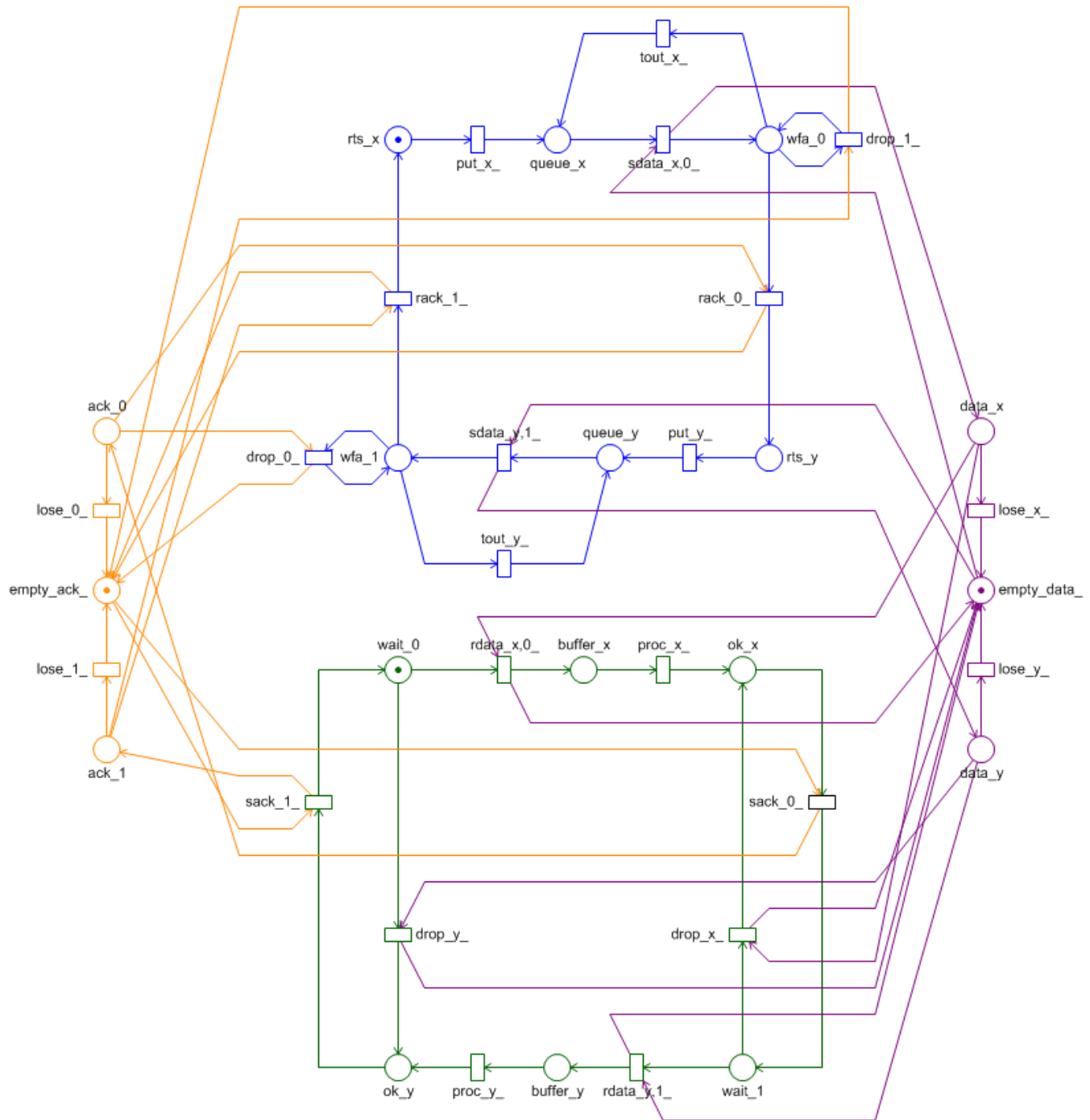
- sack()
- rack()
- drop()

Nyugta vesztese:

- lose()



# Teljes modell



# A példa modell analízise

# PetriDotNet: A modell dinamikus tulajdonságai

## A(z) AlterBit háló tulajdonságai

### Dinamikus tulajdonságok

Állapotok száma:	108
Korlátosság:	<b>korlátos</b> 1-korlátos (biztos háló)
Holtpontmentesség:	<b>holtpontmentes</b>
Megfordíthatóság:	<b>megfordítható</b>
Perszisztencia:	<b>nem perzisztens</b>

### Strukturális tulajdonságok

Legszűkebb alosztály:	Petri-háló
Tisztaság:	<b>nem tiszta (van hurokél)</b>

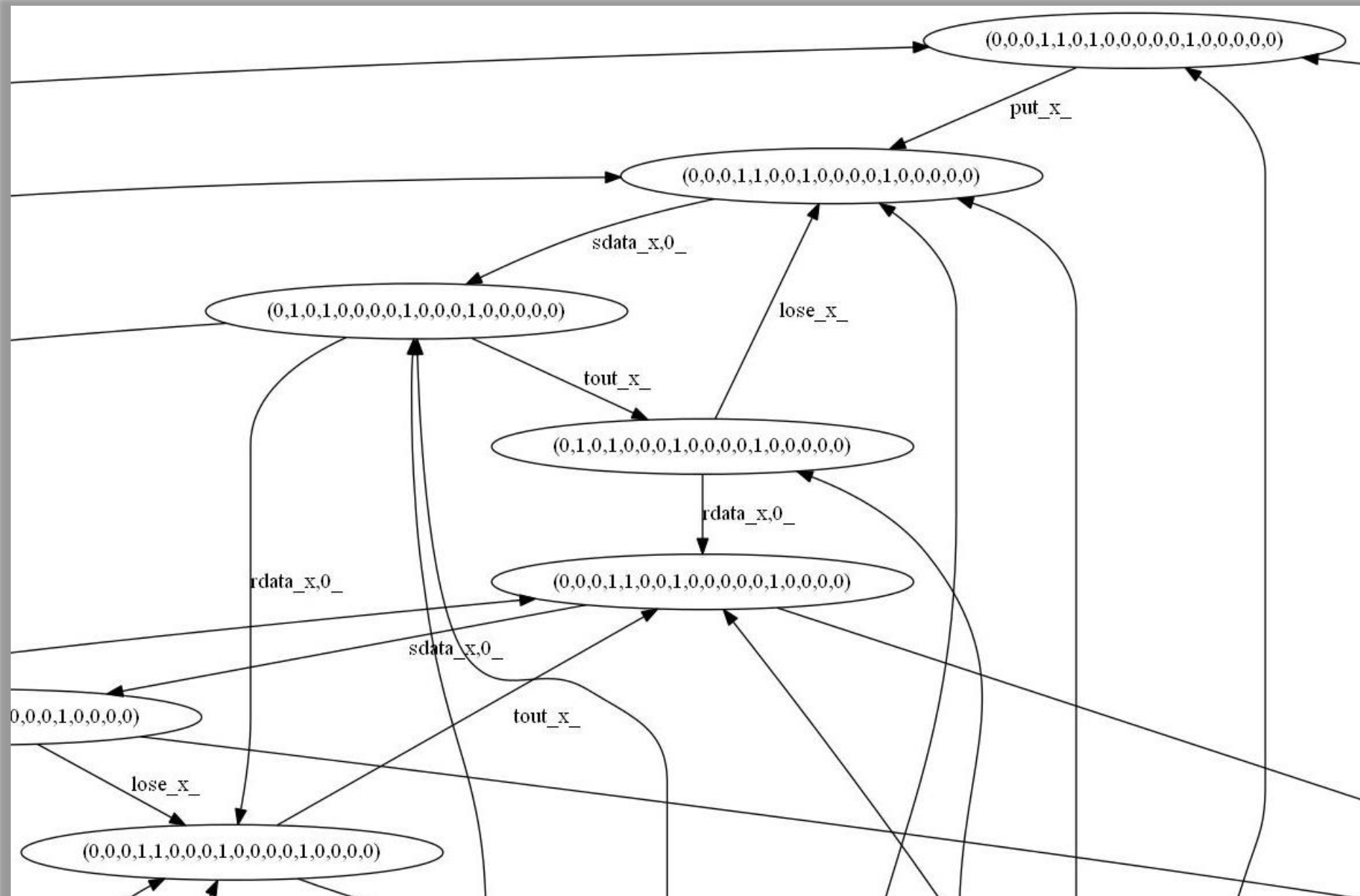
[Elérhetőség vizsgálata:](#) [CTL-kifejezés vizsgálata:](#)

[Elérhetőségi gráf mentése:](#) [Szomszédossági mátrix mentése:](#)

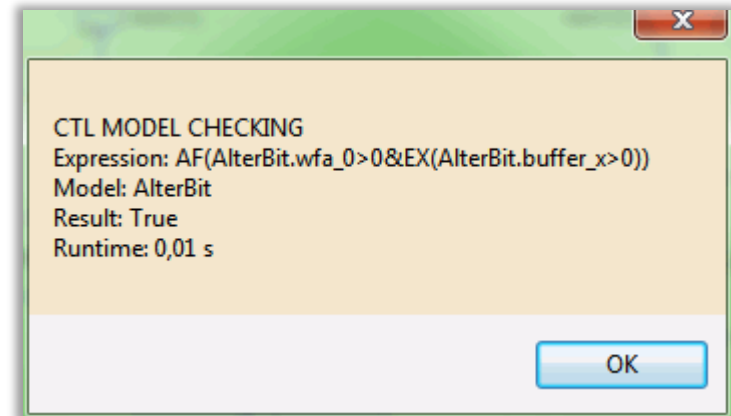
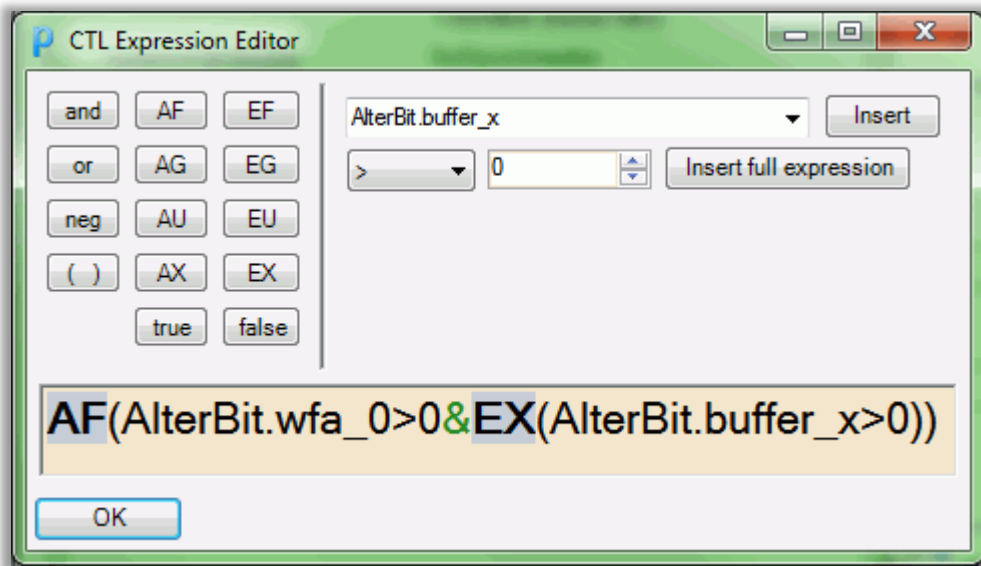
[T-invariánsok keresése:](#) [P-invariánsok keresése:](#)

[Helyek tokenkorlátjainak kiírása:](#)

# PetriDotNet: Elérhetőségi gráf kirajzolása (GraphViz)



# PetriDotNet: CTL modellellenőrzés



**AF**(AlterBit.wfa\_0>0 & **EX**(AlterBit.buffer\_x>0))

⇒ True

**AG**(**AF**(AlterBit.queue\_y>0))

⇒ False

**AF**(**EG**(AlterBit.queue\_x=0))

⇒ True

**EF**(AlterBit.wfa\_0>0 & AlterBit.data\_x=0)

⇒ True

**AF**(AlterBit.queue\_x>0 & **AX**(AlterBit.wfa\_0>0 & AlterBit.data\_x>0)) ⇒ True



# PetriDotNet: Invariáns analízis

The screenshot displays the PetriDotNet application interface with three windows open:

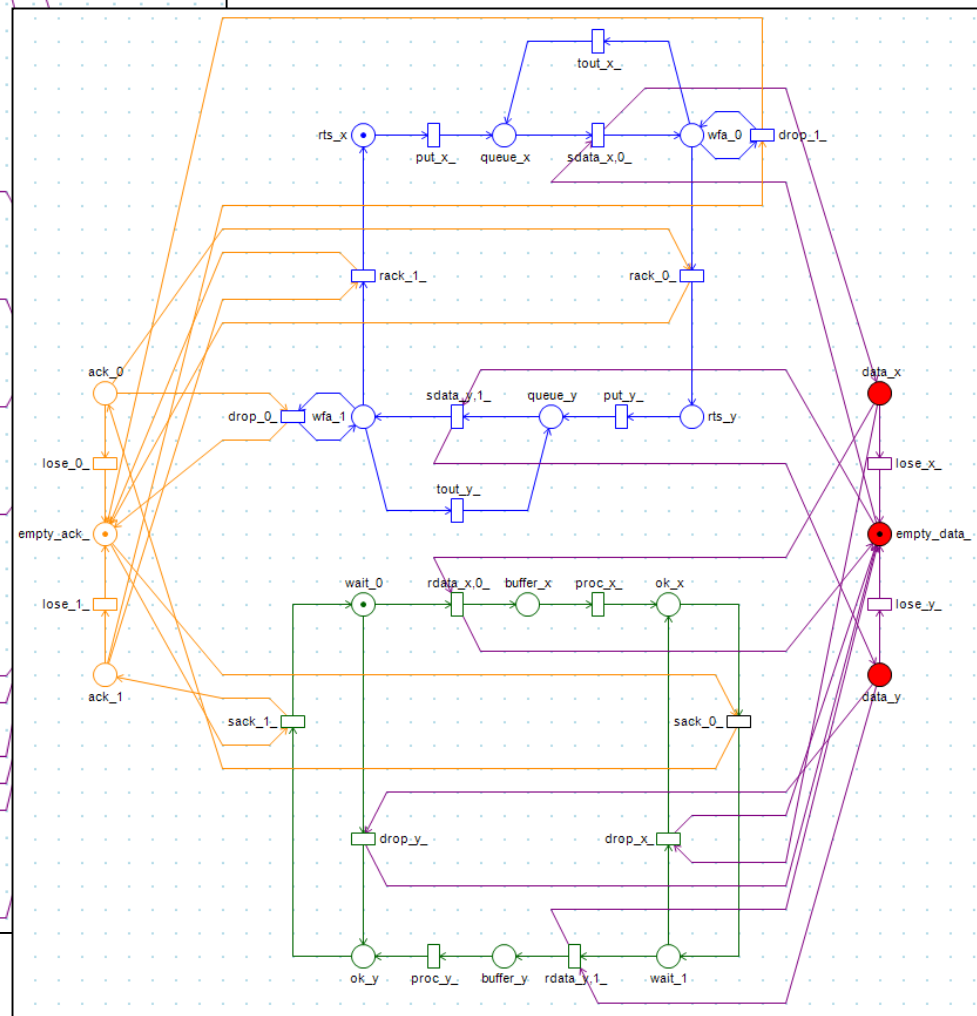
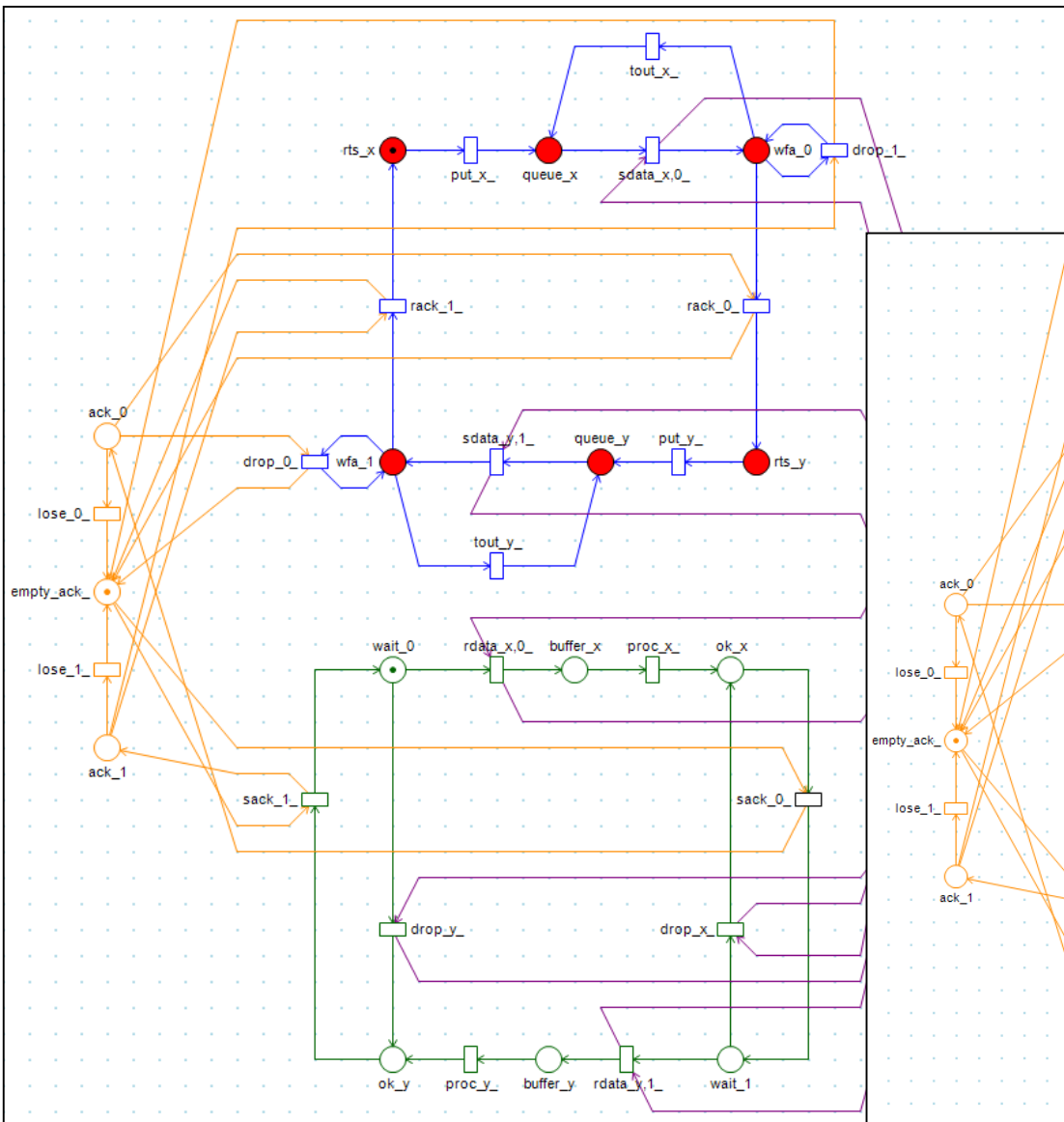
- Háló tulajdonságai**: Contains two "ShowInvariants" windows. The first shows the expression `{lose(x), sdata(x,0), tout(x)}` and the second shows `{ack_0, ack_1, empty(ack)}`.
- P-Invariants**: A dialog box titled "List of P-Invariants calculated by Martinez-Silva algorithm". It reports "Calculation finished in 0,00 ms. (places=18, transitions=22)" and lists the following invariants:

```
{ack_0, ack_1, empty(ack)}
{data_x, empty(data), data_y}
{rts_x, queue_x, wfa_0, rts_y, wfa_1, queue_y}
{wait_0, buffer_x, ok_x, ok_y, buffer_y, wait_1}
```
- T-Invariants**: A dialog box titled "List of T-Invariants calculated by Martinez-Silva algorithm". It reports "Calculation finished in 15,60 ms. (places=18, transitions=22)" and lists 22 invariants, including:

```
{lose(x), sdata(x,0), tout(x)}
{lose(y), sdata(y,1), tout(y)}
{rack(1), put(x), sdata(x,0), rack(0), sdata(y,1), put(y), drop(y), sack(0), drop(x), sack(1)}
{lose(1), sdata(y,1), tout(y), drop(y), sack(1)}
{drop(1), sdata(y,1), tout(y), drop(y), sack(1)}
{lose(y), rack(1), put(x), sdata(x,0), rack(0), sdata(y,1), put(y), tout(y), drop(y), sack(0), drop(x), sack(1)}
{lose(0), sdata(x,0), tout(x), sack(0), drop(x)}
{sdata(x,0), tout(x), drop(0), sack(0), drop(x)}
{lose(x), rack(1), put(x), sdata(x,0), tout(x), rack(0), sdata(y,1), put(y), drop(y), sack(0), drop(x), sack(1)}
{lose(x), lose(y), rack(1), put(x), sdata(x,0), tout(x), rack(0), sdata(y,1), put(y), tout(y), drop(y), sack(0), drop(x), sack(1)}
{rack(1), put(x), sdata(x,0), rack(0), sdata(y,1), put(y), rdata(x,0), proc(x), sack(0), proc(y), rdata(y,1), sack(1)}
{lose(x), rack(1), put(x), sdata(x,0), tout(x), rack(0), sdata(y,1), put(y), rdata(x,0), proc(x), sack(0), proc(y), rdata(y,1), sack(1)}
{rack(1), put(x), sdata(x,0), rack(0), sdata(y,1), put(y), rdata(x,0), proc(x), drop(y), sack(0), drop(x), proc(y), rdata(y,1), sack(1)}
{lose(0), rack(1), put(x), sdata(x,0), tout(x), rack(0), sdata(y,1), put(y), rdata(x,0), proc(x), sack(0), drop(x), proc(y), rdata(y,1), sack(1)}
{rack(1), put(x), sdata(x,0), tout(x), rack(0), sdata(y,1), put(y), drop(0), rdata(x,0), proc(x), sack(0), drop(x), proc(y), rdata(y,1), sack(1)}
{lose(x), rack(1), put(x), sdata(x,0), tout(x), rack(0), sdata(y,1), put(y), rdata(x,0), proc(x), drop(y), sack(0), drop(x), proc(y), rdata(y,1), sack(1)}
{lose(x), rack(1), put(x), sdata(x,0), tout(x), rack(0), sdata(y,1), put(y), rdata(x,0), proc(x), drop(y), sack(0), drop(x), proc(y), rdata(y,1), sack(1)}
{lose(x), lose(y), rack(1), put(x), sdata(x,0), tout(x), rack(0), sdata(y,1), put(y), tout(y), rdata(x,0), proc(x), drop(y), sack(0), drop(x), proc(y), rdata(y,1), sack(1)}
{lose(x), lose(1), rack(1), put(x), sdata(x,0), tout(x), rack(0), sdata(y,1), put(y), tout(y), rdata(x,0),
```

# PetriDotNet: P-invariánsok (példák)

Komponensek  
állapotgépei



# PetriDotNet: T-invariánsok (példák)

Ciklikus működések (itt:  
hibátlan ill. adatvesztés)

