

Petri hálók analízise

dr. Bartha Tamás

dr. Majzik István

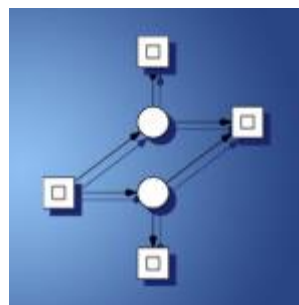
dr. Pataricza András

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

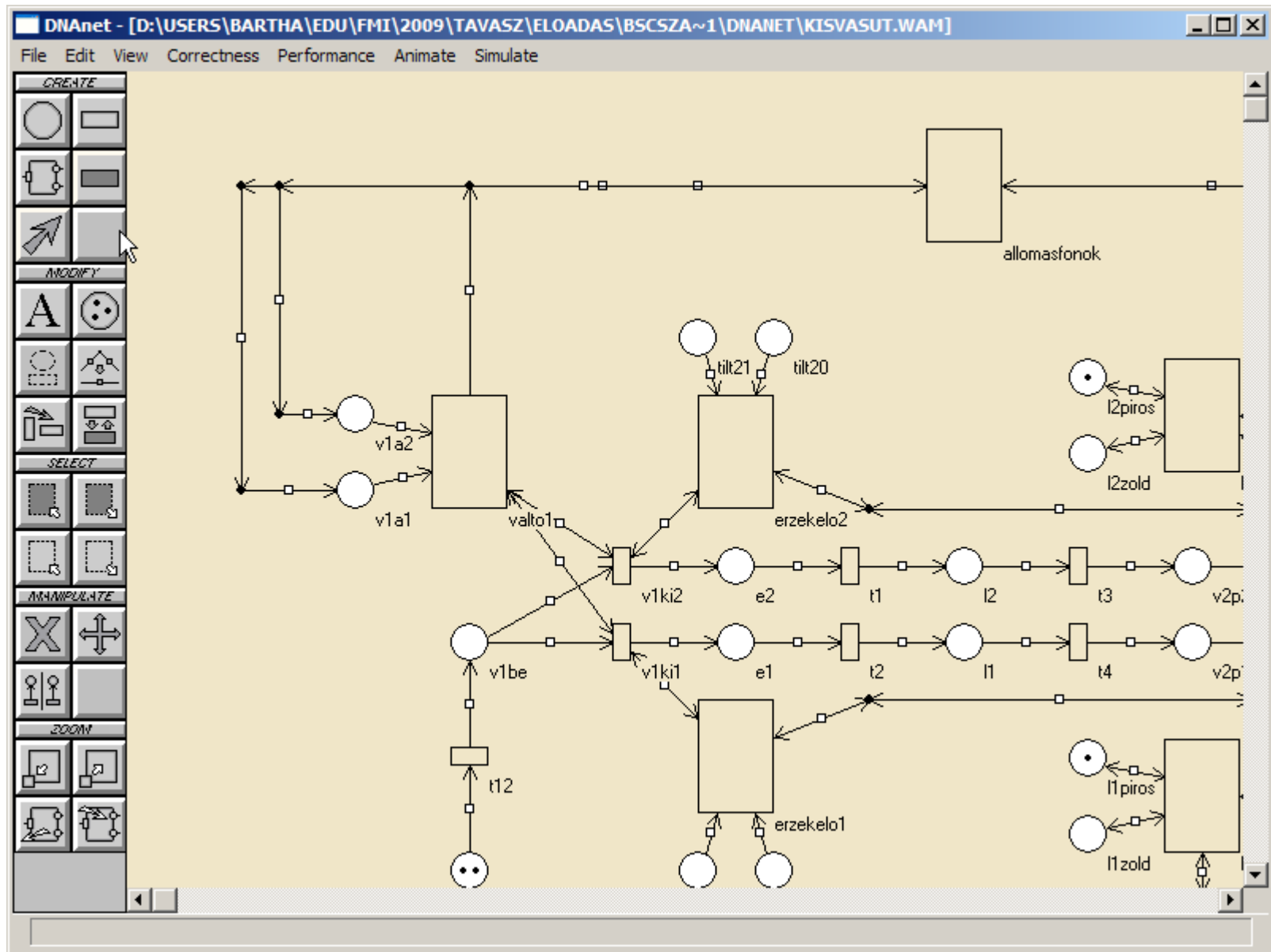
petridotnet
from BME-MIT



A DNAnet modellező program

- **Képességei**
 - Grafikus szerkesztő
 - 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**
 - Nem túl stabil ☹️

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

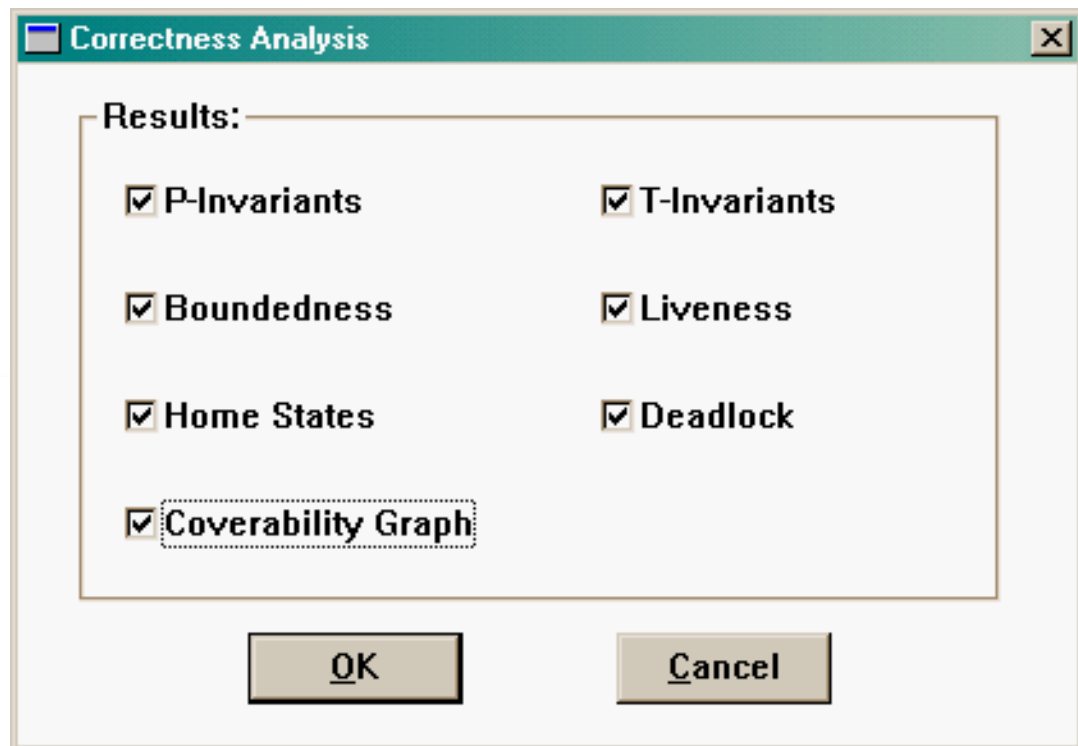
108 hash table entries used

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

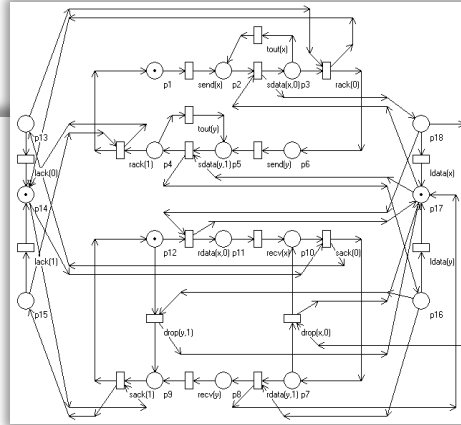
P-invariants:

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

- nyer**: A green rectangular node on the left.
- veszi t**: A red rectangular node on the right.
- jatekban**: A white circular node at the bottom.
- uzemben**: A white circular node in the middle.
- zseton**: A white circular node at the top, containing two black dots.

Edges connect the nodes: "nyer" to "zseton" (weight 2), "nyer" to "uzemben", "veszi t" to "uzemben", "jatekban" to "uzemben", "uzemben" to "zseton", and "zseton" to "jatekban".

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: (in alphabetic order)	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

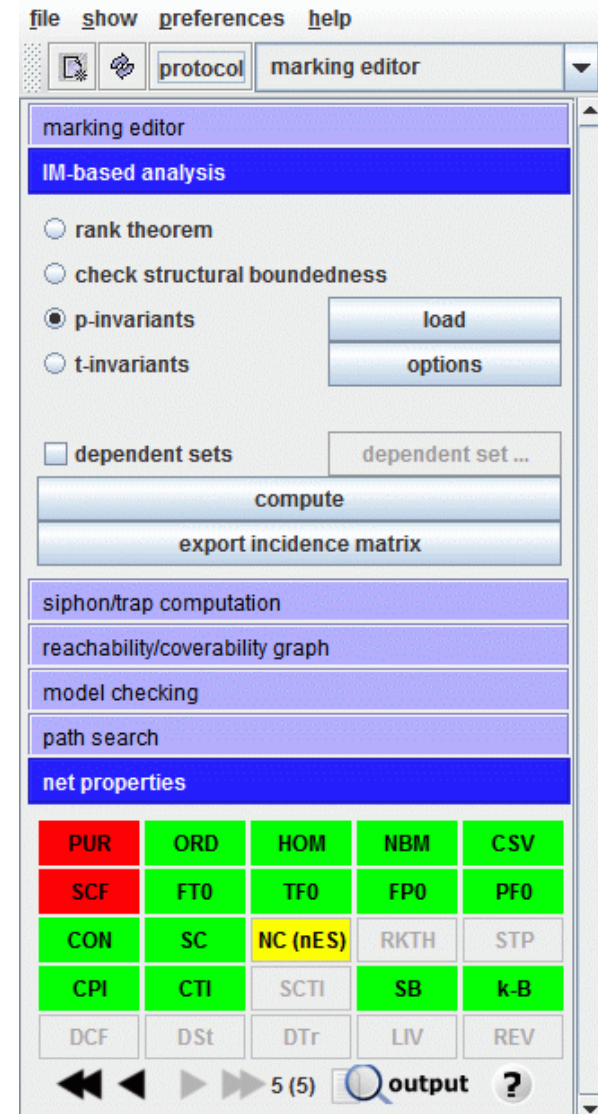
<http://www-dssz.informatik.tu-cottbus.de/software/snoopy.html>

Any comments to:
snoopy@informatik.tu-cottbus.de

OK

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 képességei nincsenek



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
- **Saját fejlesztés**: 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), nyer (left), veszit (center), indul (right), jatekban (bottom).
- Transitions:** uzemben (center).
- Edges:** Directed edges connect the places and transitions. A weight of 2 is shown on the edge from nyer to uzemben.
- Tokens:** Two tokens are in zseton, one in uzemben, and one in jatekban.

The interface includes a menu bar (File, Edit, View, Insert, Mode, Tools, Add-in, Window, Help), a toolbar, and a left sidebar with a **Toolbox** (containing Select, Place, Transition, Edge, Token, and Other elements) and a **Properties** panel. The Properties panel shows the **Identity** section with the following data:

Identity	
Id	n1
Name	Net 1

The **Name** field is labeled "Name of the Petri net." and contains "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.

Buttons: Send error feedback, OK

At the bottom of the dialog box, there is a logo for MŰEGYETEM 1782 and a small Petri net diagram with nodes labeled F, T, S, and RG.

A PetriDotNet analízis képességei

A(z) AlterBit háló tulajdonságai

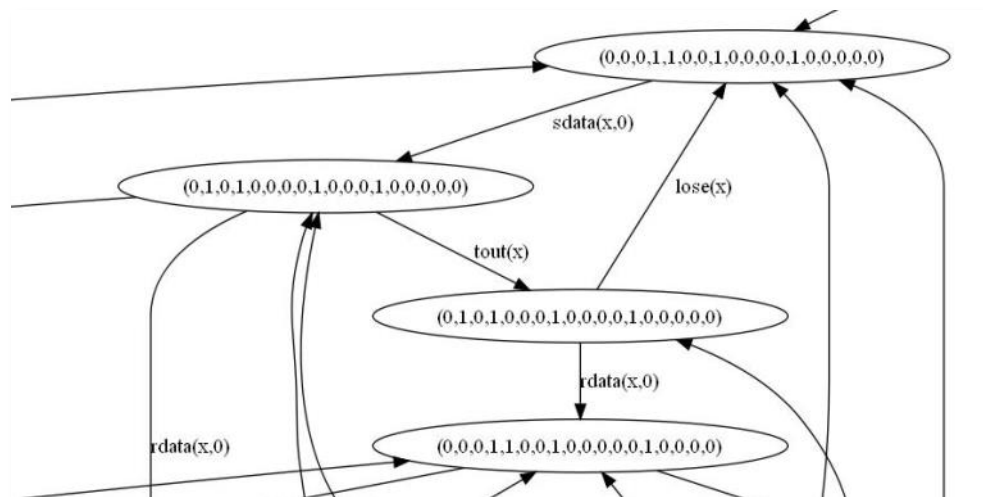
Dinamikus tulajdonságok

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

Strukturális tulajdonságok

Legszűkebb alosztály:	Petri-háló
Tisztaság:	nem tiszta (van hurokél)

[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 edited is $AF(AlterBit.wfa_0 > 0 \& EX(AlterBit.buffer_x > 0))$. The window includes a toolbar with logical operators and temporal logic symbols, and a text input field for the expression.

The dialog box displays the results of the CTL model checking process. The expression is $AF(AlterBit.wfa_0 > 0 \& EX(AlterBit.buffer_x > 0))$, the model is AlterBit, and the result is True. The runtime is 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 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)}  
...
```

At the bottom 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
- 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
- Ha a küldött üzenethez csatolt bit **b**, akkor
 - Ha az üzenet elvész, a küldő időtúllepéssel észleli a nyugta hiányát → újra küldi
 - Ha a folyamat **b** bittel ellátott nyugtát kap (ilyet várt), akkor a következő üzenethez $\neg b$ bitet csatol
 - Ha a folyamat $\neg b$ bittel ellátott nyugtát kap (nem ilyet várt), akkor egyszerűen eldobja (majd időtúllépés lesz a **b** bites nyugta hiánya miatt)

Fogadó folyamat

- Az üzenet vételét nyugtázza a kapott ellenőrző bit visszaküldésével
- Ha **b** ellenőrző bittel jelölt üzenetet kap valamint ezt a **b** bit visszaküldésével nyugtázza, akkor
 - Ha a következő üzenetben az ellenőrző bit értéke $\neg b$ (helyesen), akkor az új üzenetet is feldolgozza és a $\neg b$ bit visszaküldésével nyugtázza
 - Ha a következő üzenetben az ellenőrző bit értéke **b** (azaz nem megfelelő), akkor az üzenetet eldobja, de **b** nyugtát küld (mondván, hogy bizonyára újraküldés történt a nyugta hiánya miatt)

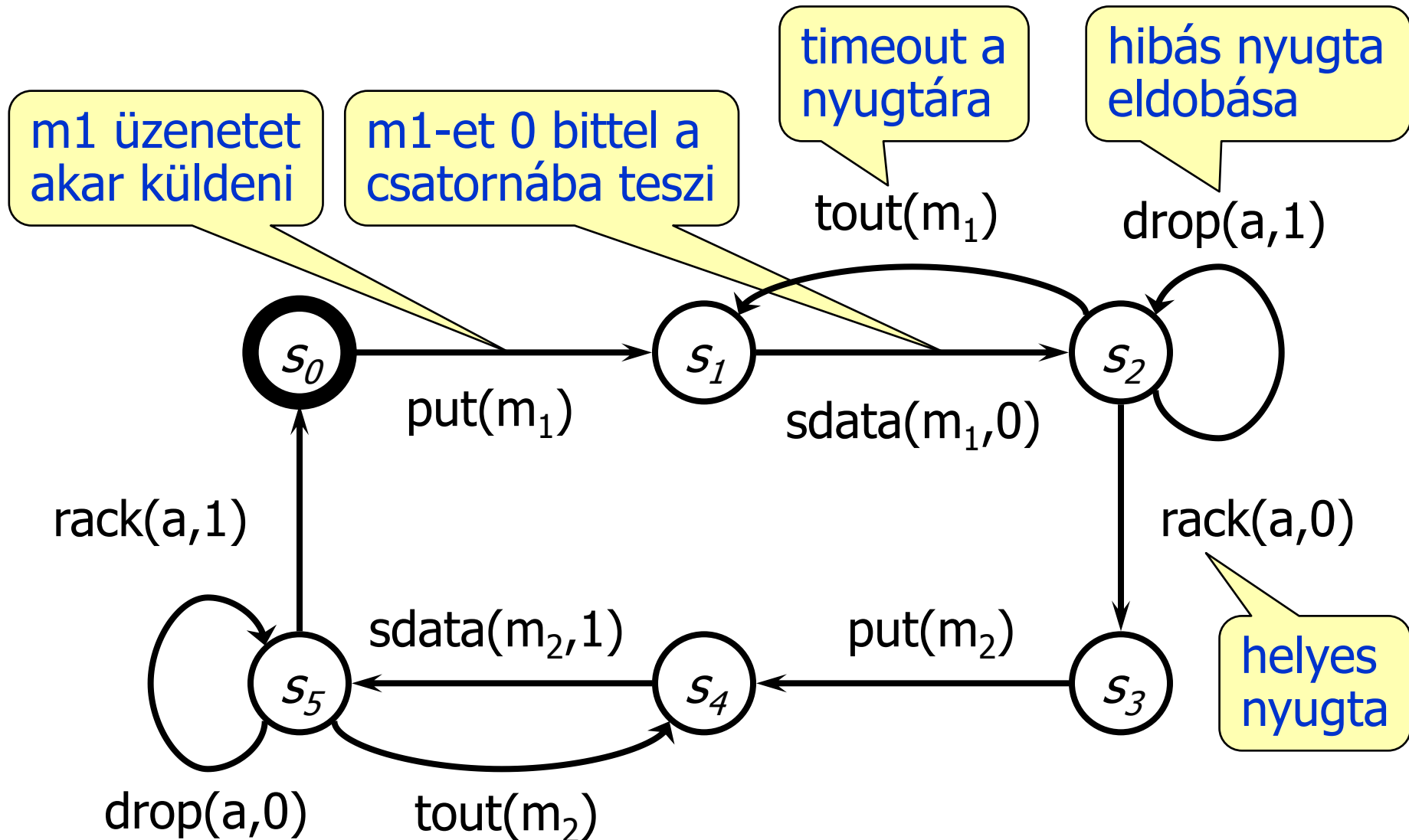
A modellalkotás lépései

1. A feladat felbontása szereplőkre és erőforrásokra
2. Szereplők állapotainak meghatározása
3. Erőforrások, üzenet pufferek állapotainak meghatározása
4. Állapot alapú modellekből Petri-háló modell készítése
5. Szereplők és erőforrások modelljeinek integrálása
6. Integrált modell helyességének ellenőrzése
7. Modell felhasználása a feladat megoldására

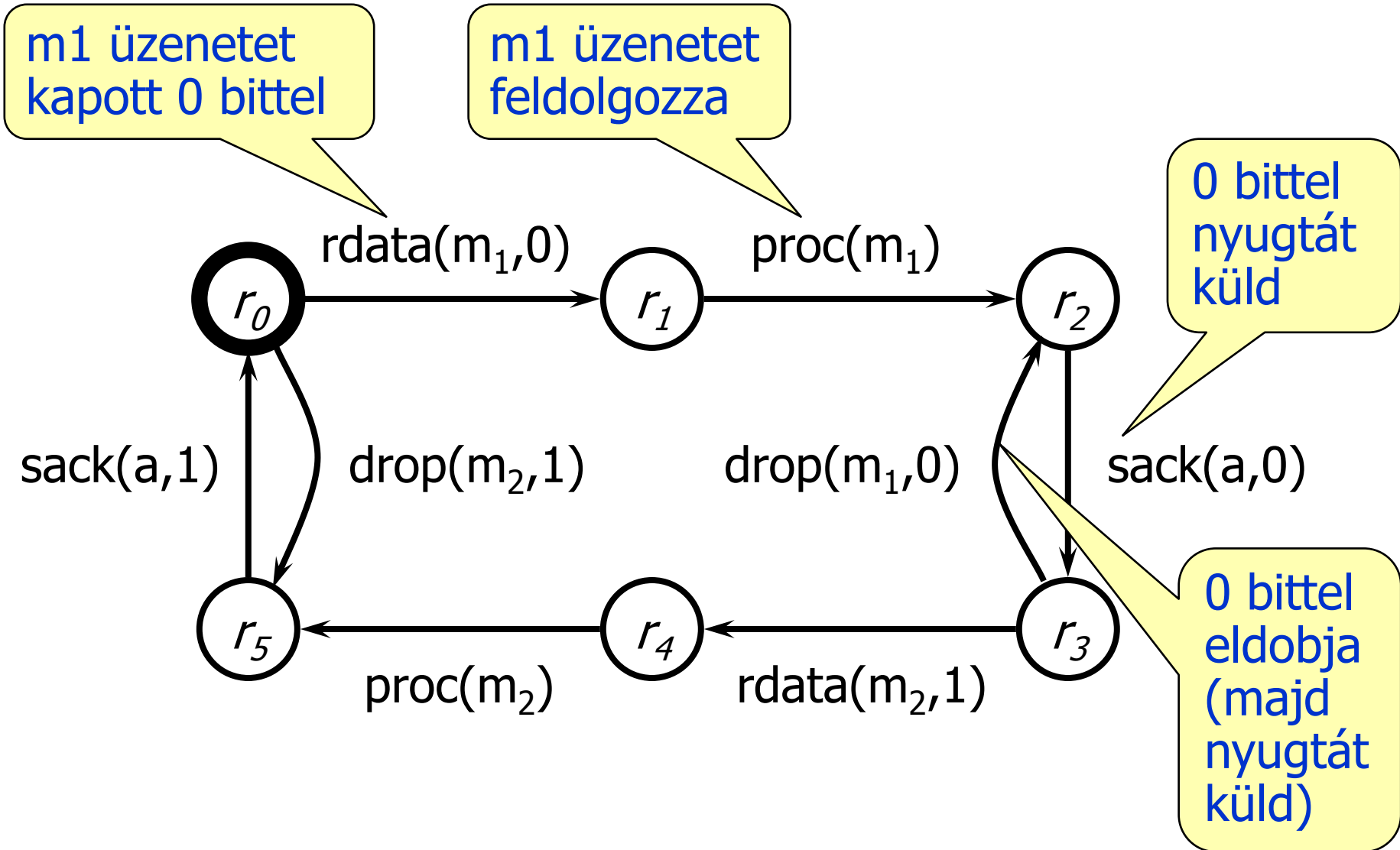
Komponensek és állapotaik

- Komponensek (alrendszer)
- Szereplők: küldő folyamat, fogadó folyamat
- Erőforrások: adat csatorna, nyugtázó csatorna
- Minden komponens saját állapottal rendelkezik
- Állapotgráf: állapotok körökkel, események nyilakkal
- Azonos események egy időben mennek végbe: szinkronizáció

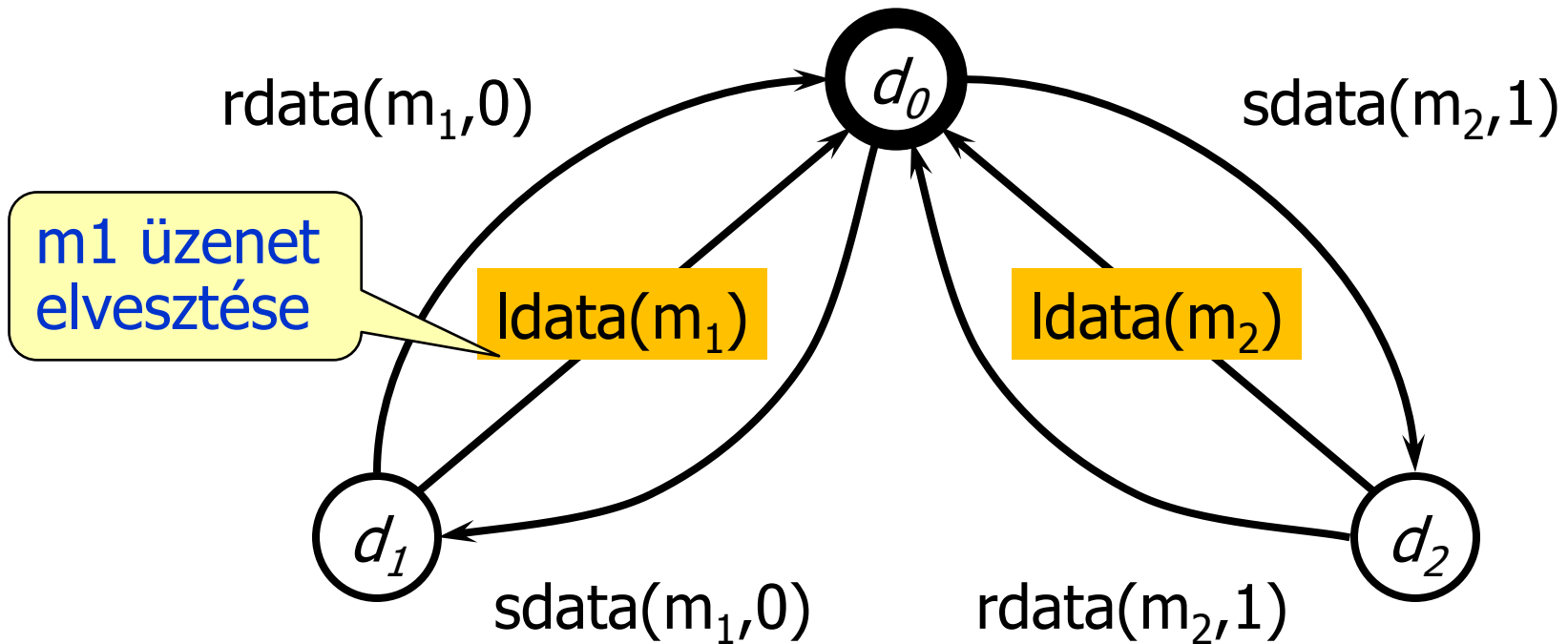
Küldő folyamat állapotgráfja



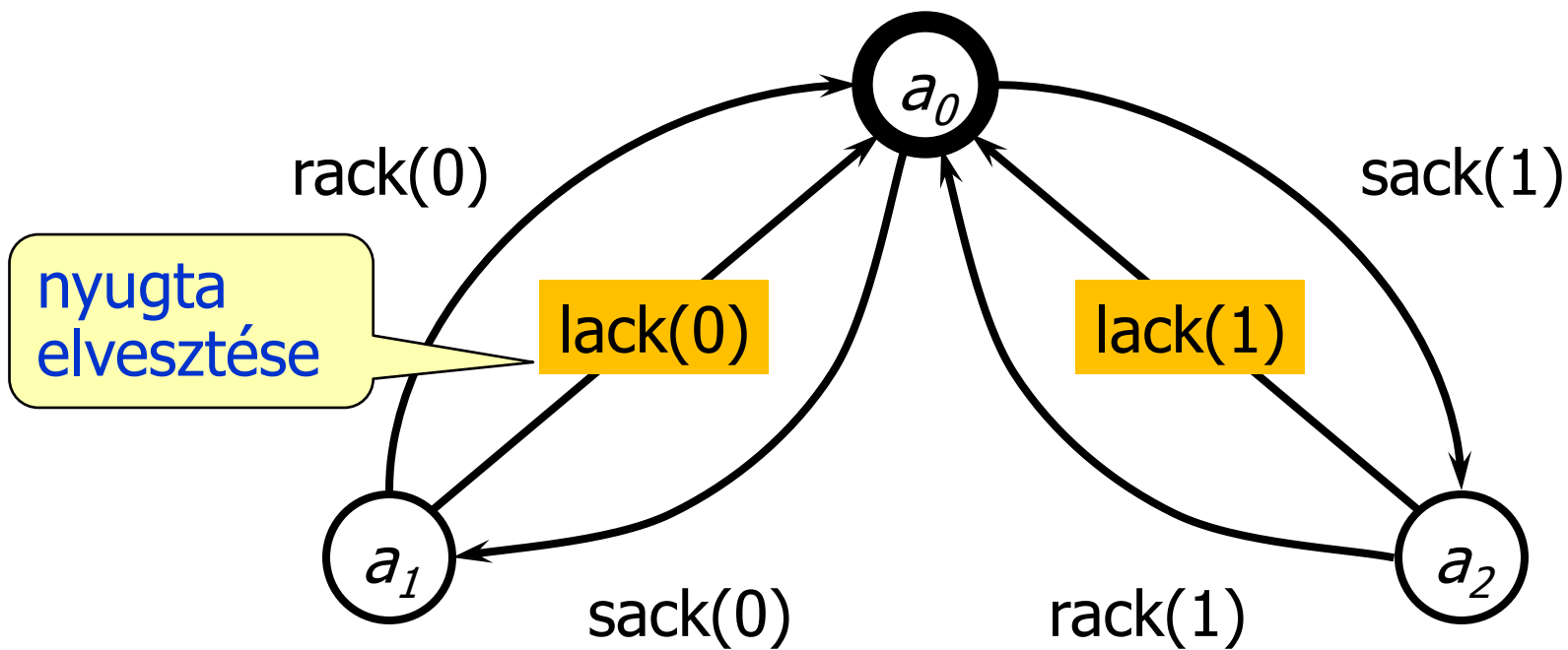
Fogadó folyamat állapotgráfja



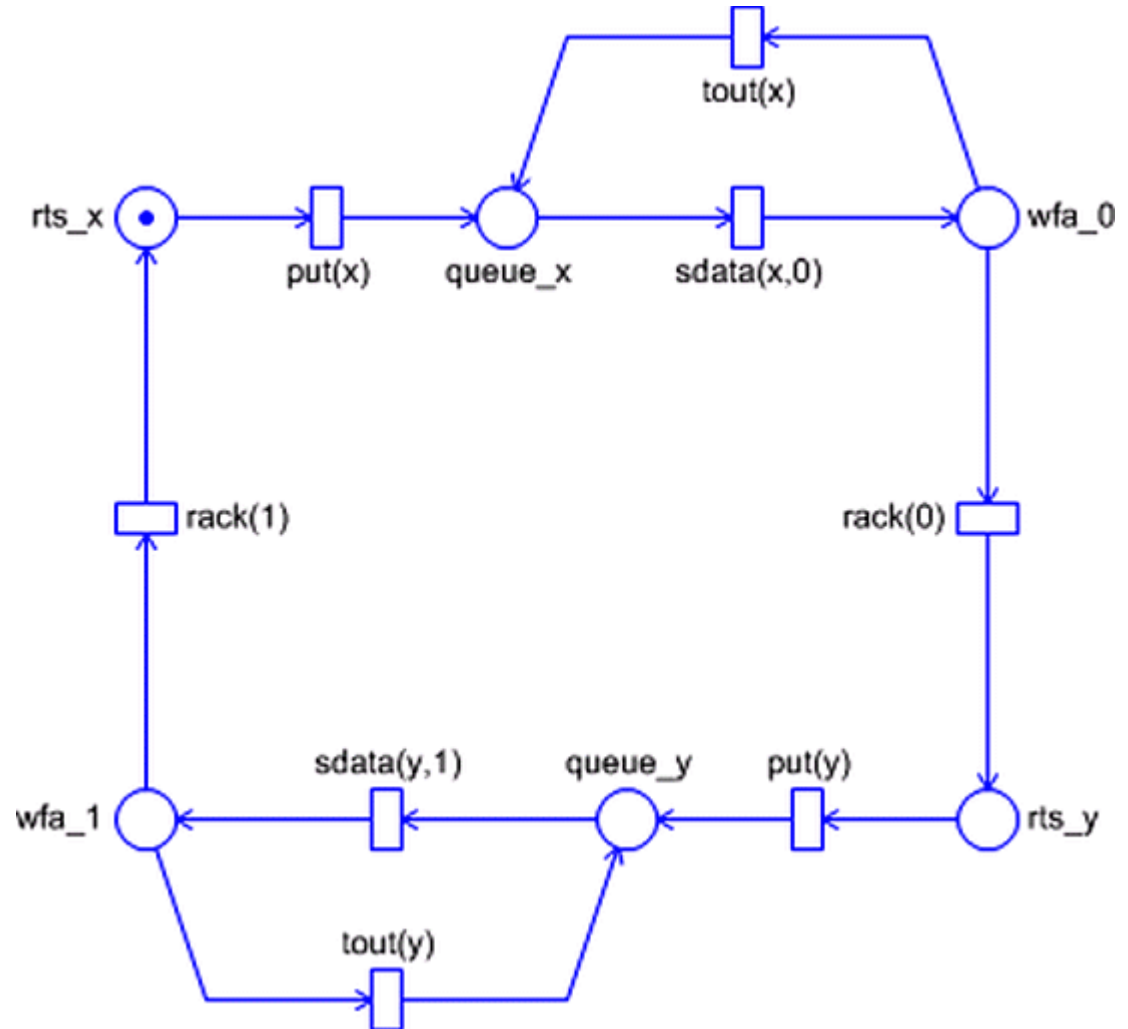
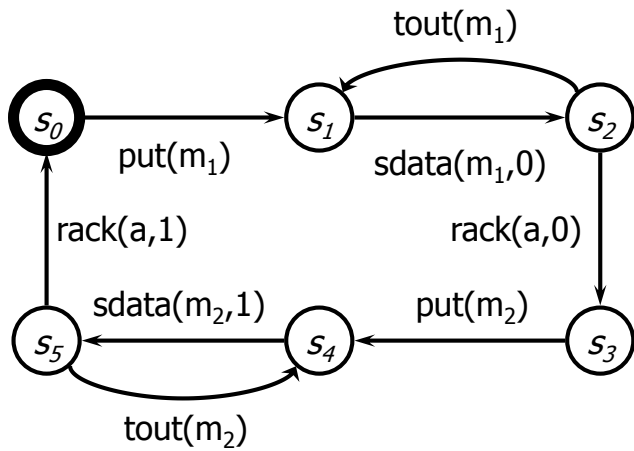
Adat csatorna állapotgráfja



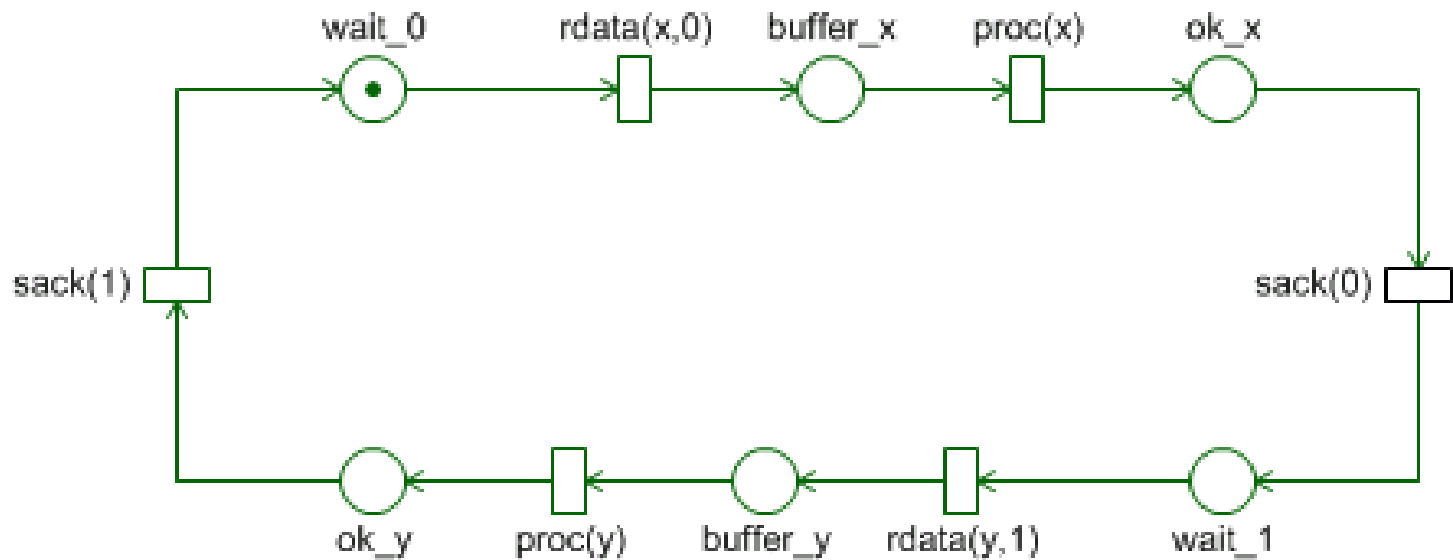
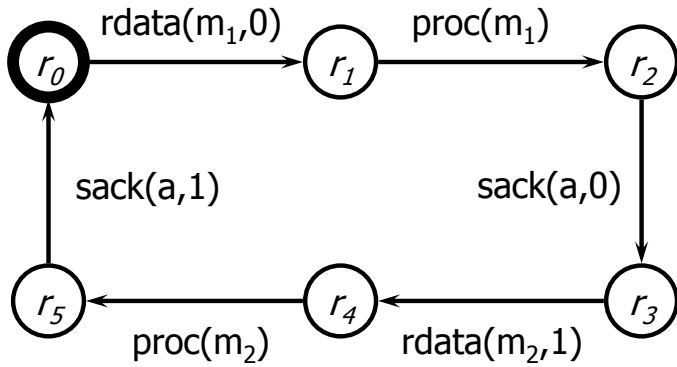
Nyugtázó csatorna állapotgráfja



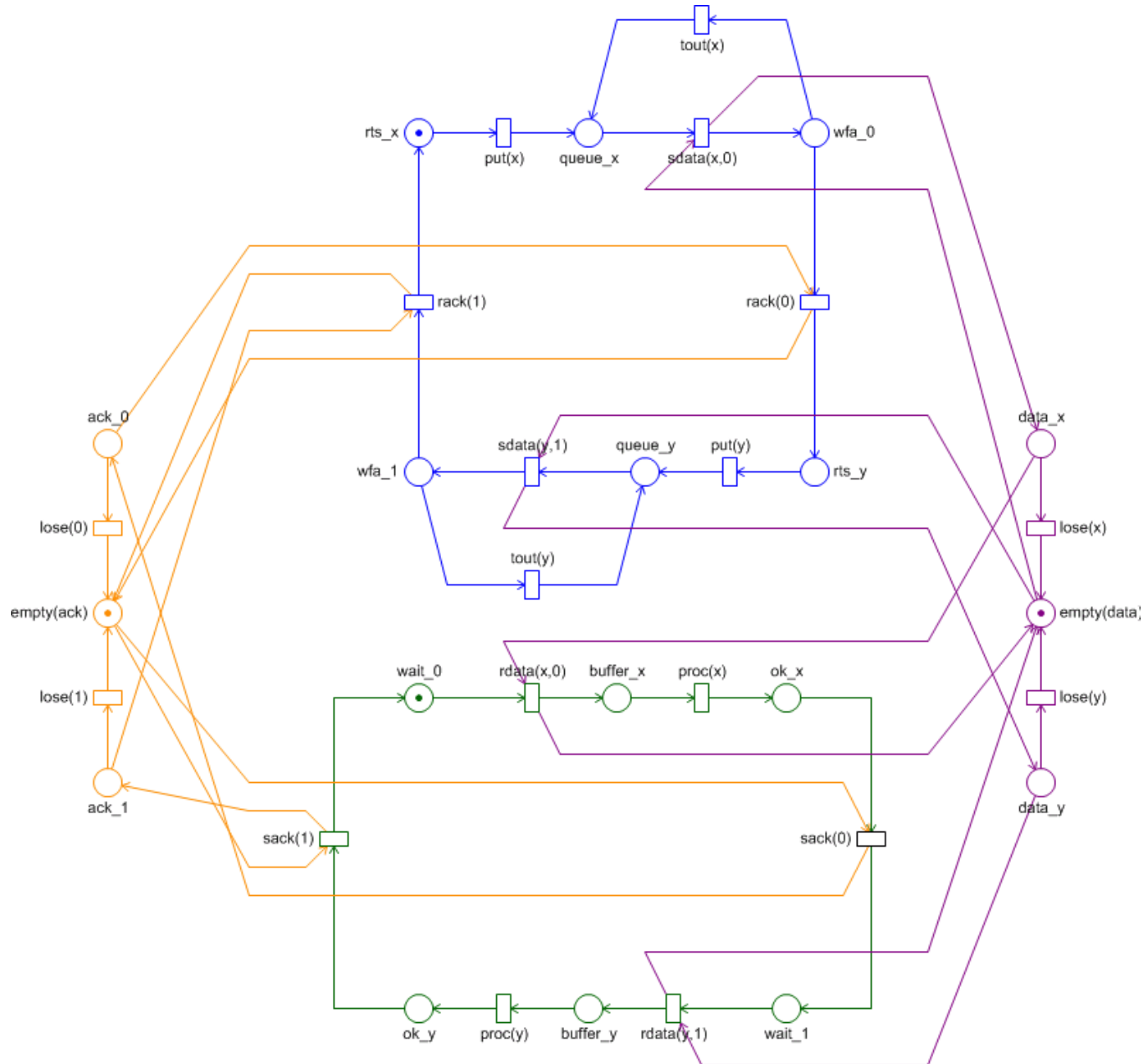
Küldő folyamat Petri háló modellje (fő ciklus)



Fogadó folyamat Petri háló modellje (fő ciklus)



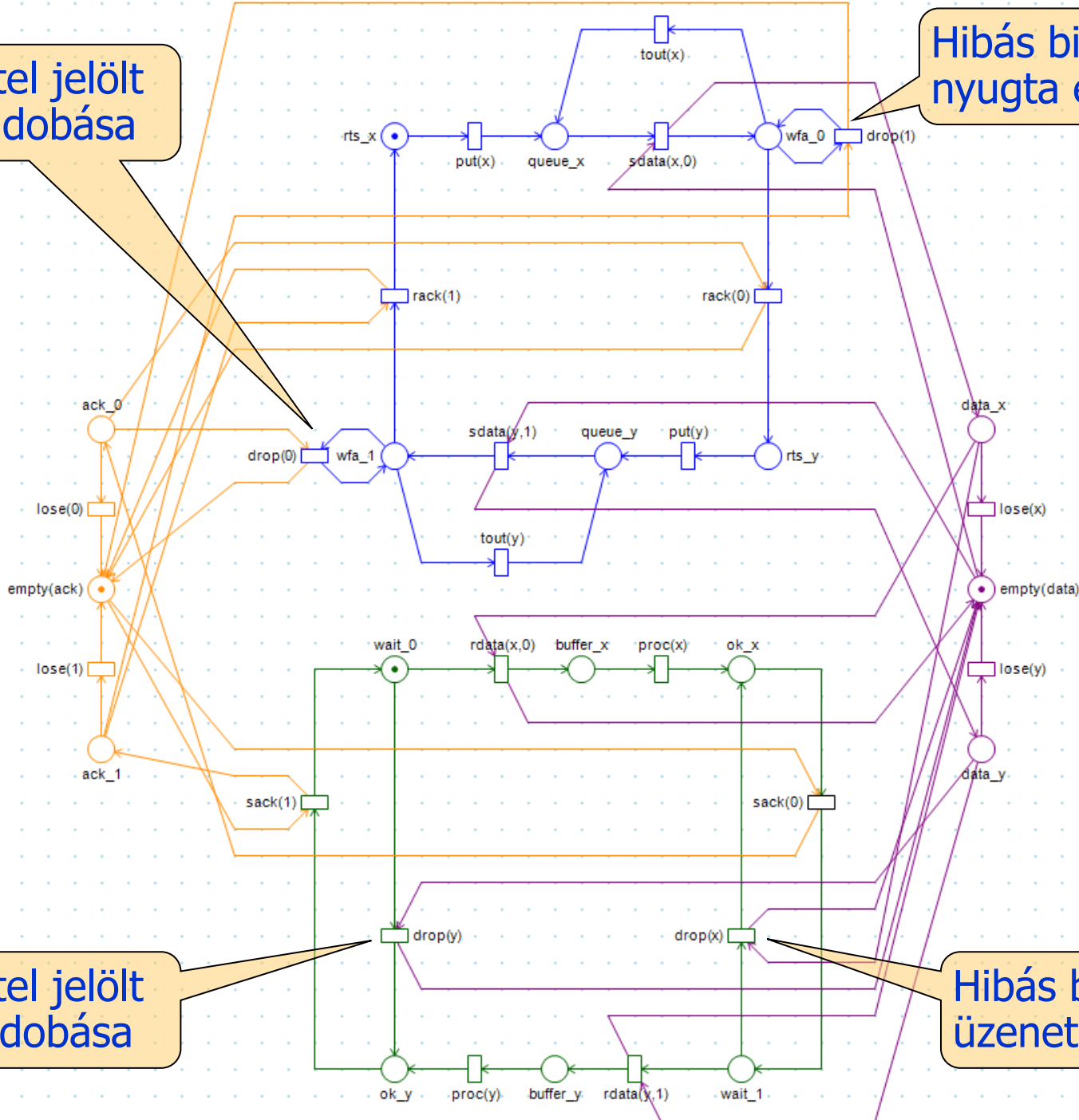
Nyugtázó csatorna és nyugtázás (fő ciklus)



Hibás bittel jelölt nyugta eldobása

Hibás bittel jelölt nyugta eldobása

Kiegészítések



Hibás bittel jelölt üzenet eldobása

Hibás bittel jelölt üzenet eldobása

A példa modell analízise

DNAnet: A modell dinamikus tulajdonságai

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

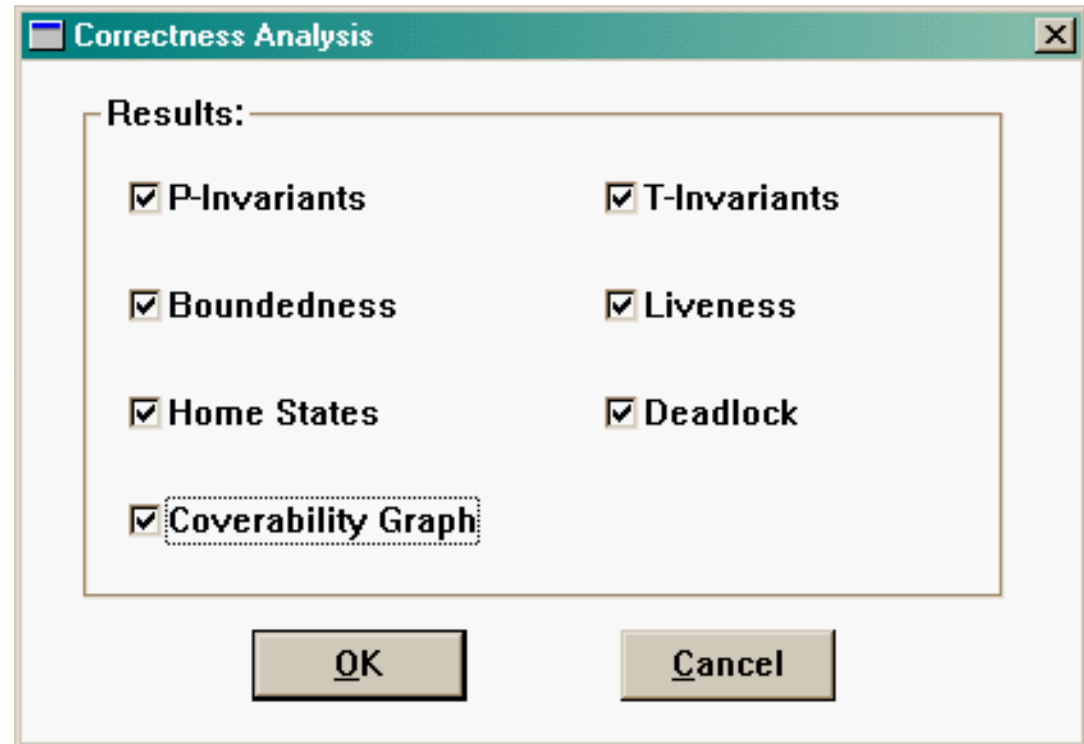
108 hash table entries used

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



PetriDotNet: A modell dinamikus tulajdonságai

A(z) AlterBit háló tulajdonságai

Dinamikus tulajdonságok

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

Strukturális tulajdonságok

Legszűkebb alosztály:	Petri-háló
Tisztaság:	nem tiszta (van hurokél)

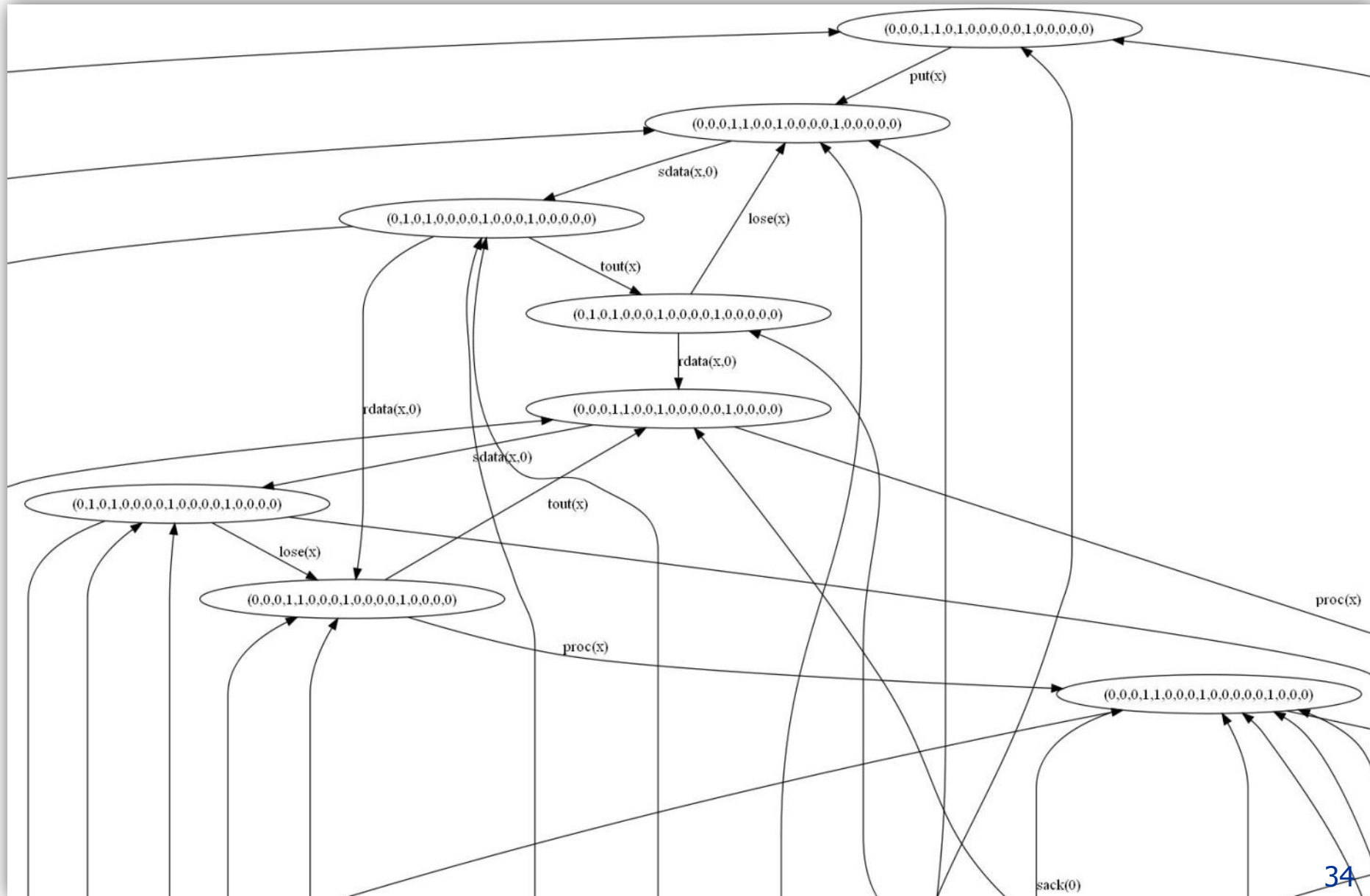
[Elérhetőség vizsgálata;](#) [CTL-kifejezés vizsgálata;](#)

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

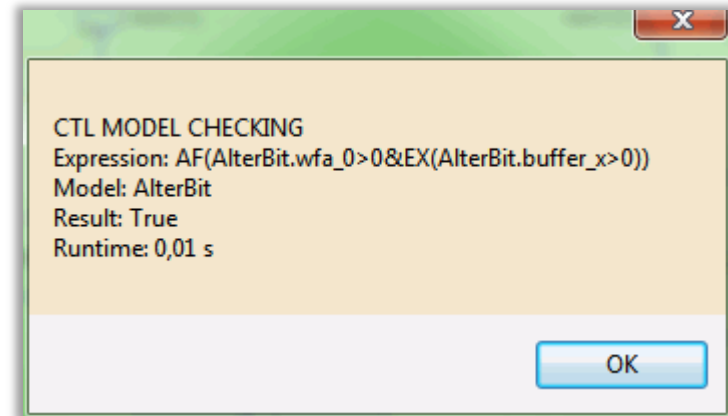
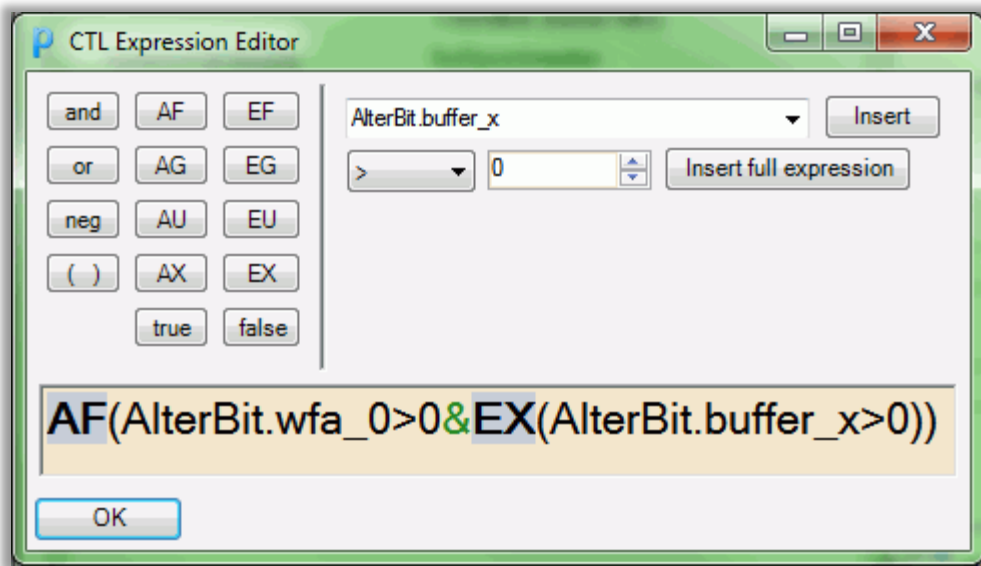
[T-invariánsok keresése;](#) [P-invariánsok keresése;](#)

[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.buffer_y>0))

⇒ False

AF(**EG**(AlterBit.buffer_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:

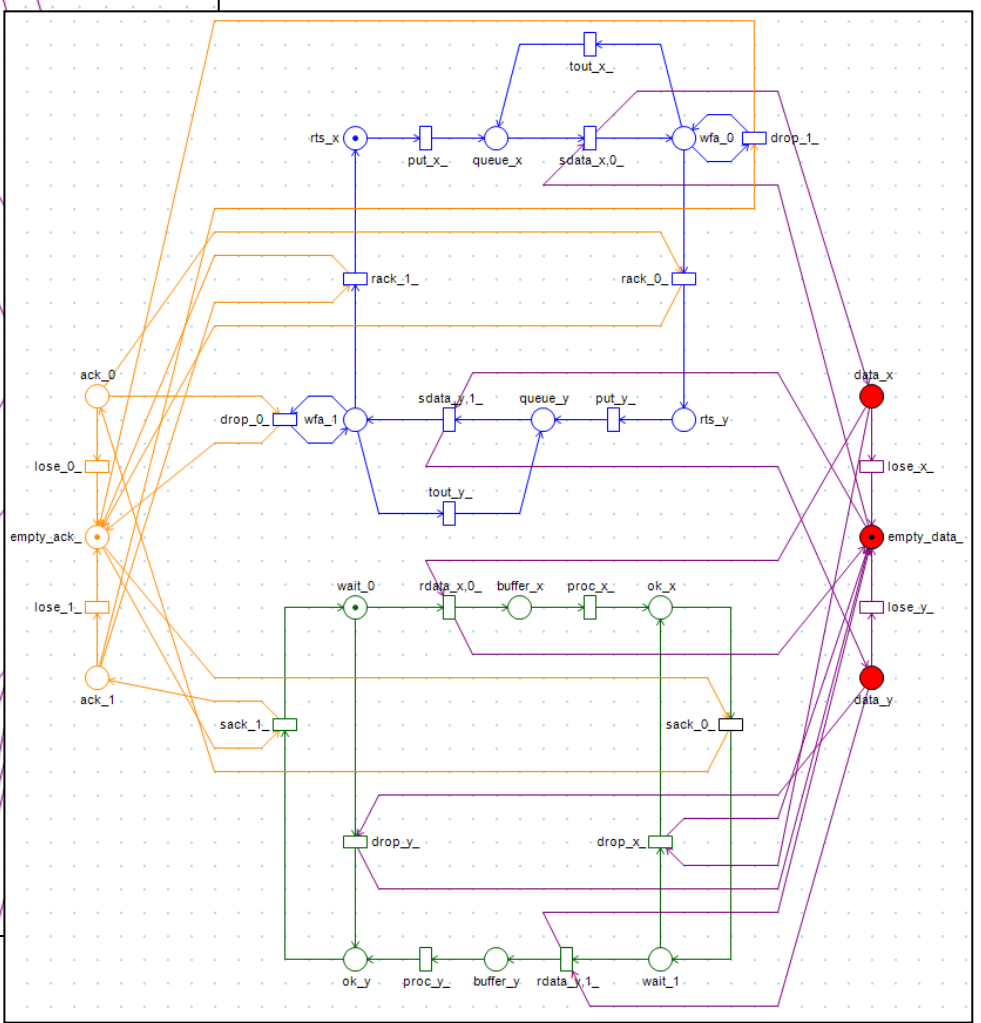
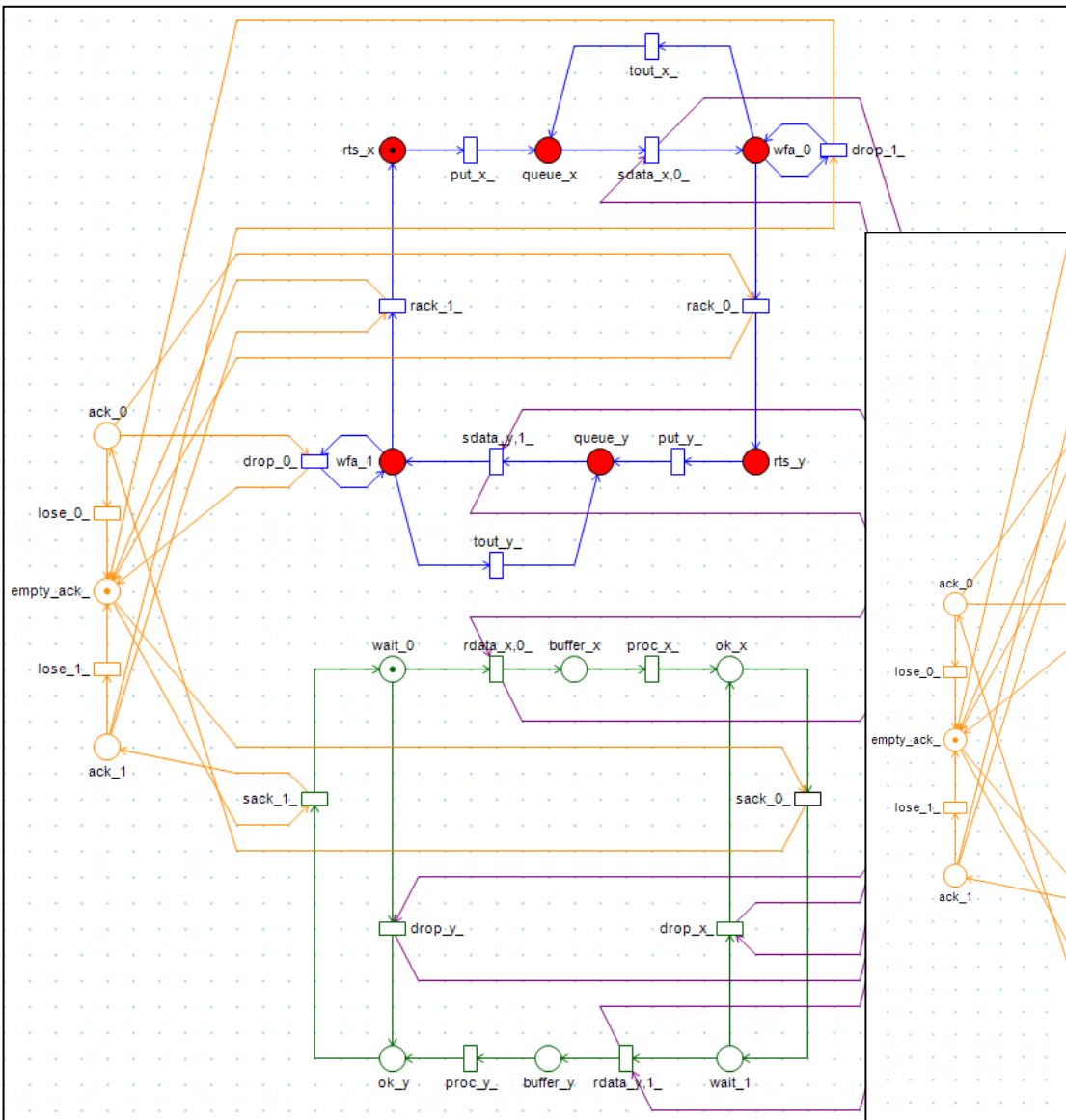
```
{ack_0, ack_1, empty(ack)}
{data_x, empty(data), data_y}
{rts_x, queue_x, wfa_0, rts_y, wfa_1, queue_y}
{wait_0, buffer_x, ok_x, ok_y, buffer_y, wait_1}
```
- T-Invariants**: A dialog box titled "List of T-Invariants calculated by Martinez-Silva algorithm". It reports "Calculation finished in 15,60 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(x), lose(1), 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))}
```

At the bottom 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](#).

PetriDotNet: P-invariánsok (példák)

Komponensek
állapotgépei



PetriDotNet: T-invariánsok (példák)

Ciklikus működések (itt:
hibátlan ill. adatvesztés)

