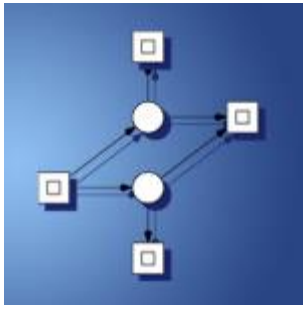
Modellezés Petri hálókkal

dr. Bartha Tamás
dr. Majzik István
dr. Pataricza András
BME Méréstechnika és Információs Rendszerek Tanszék

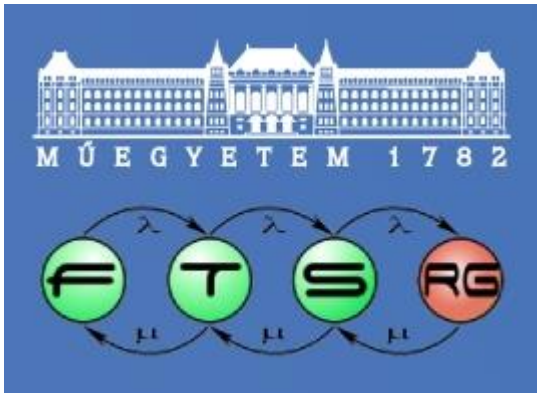
Modellező eszközök:
DNAnet, Snoopy, PetriDotNet

UNIVERSITY OF CAPE TOWN
Department of Computer Science



b-tu
Brandenburgische
Technische Universität
Cottbus

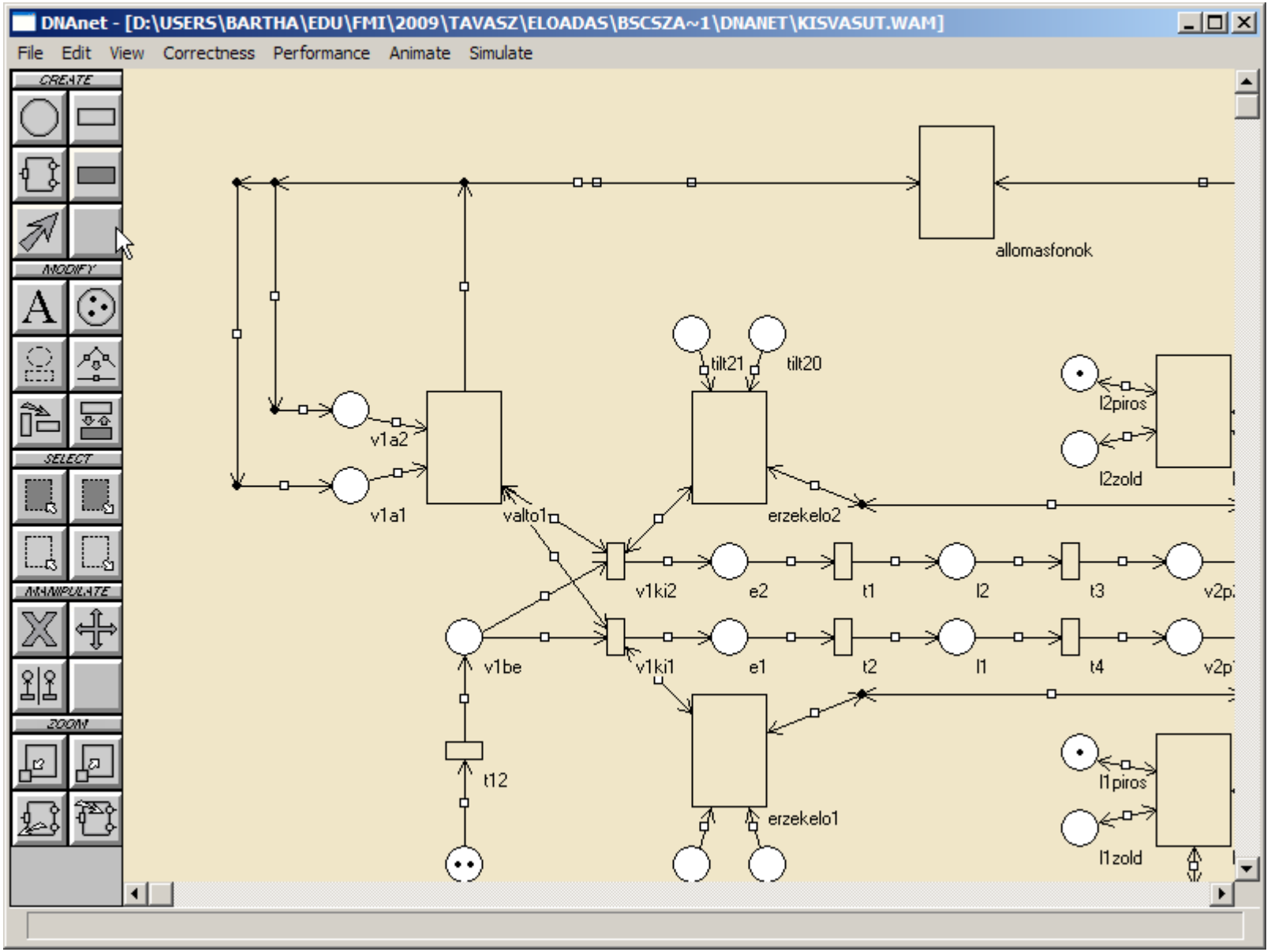
petridotnet
from BME-MIT



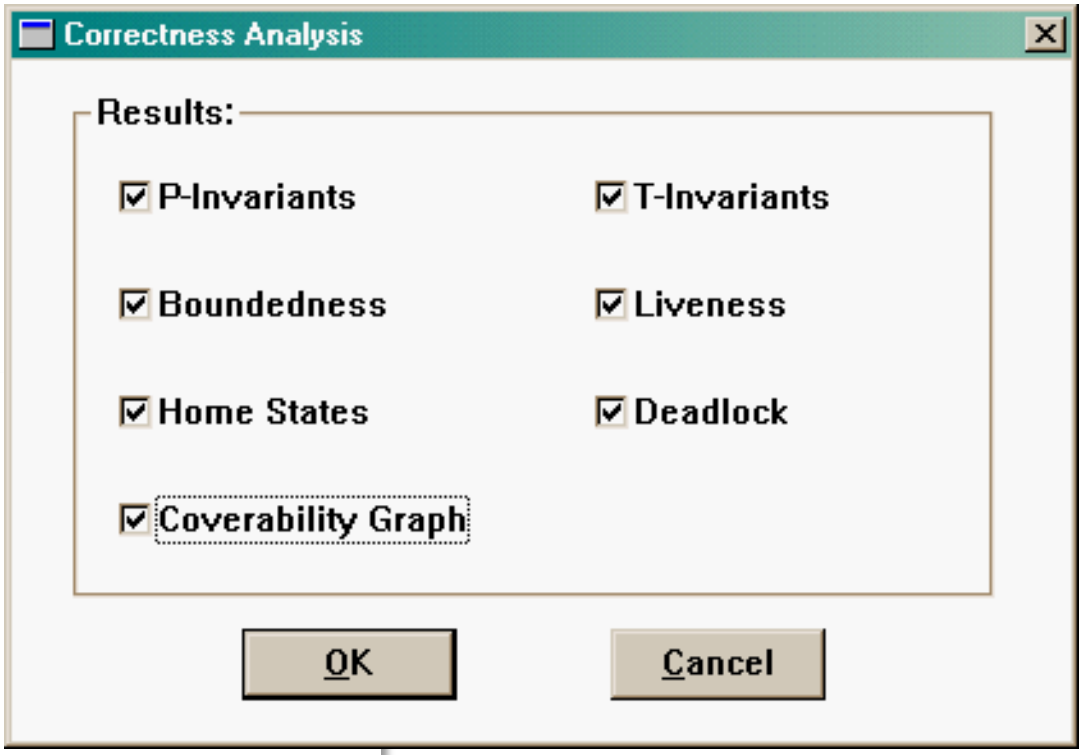
A DNAnet modellező program

- 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

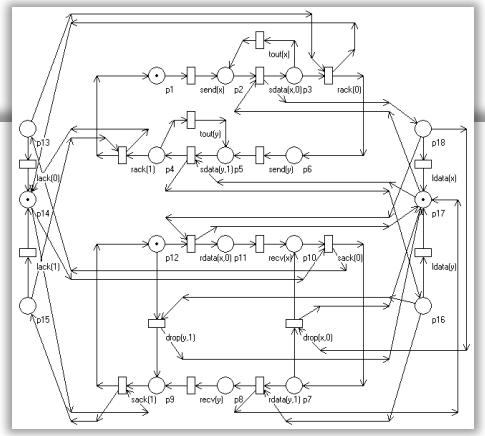


A DNAnet analízis képességei




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

All places are covered by P-invariants.

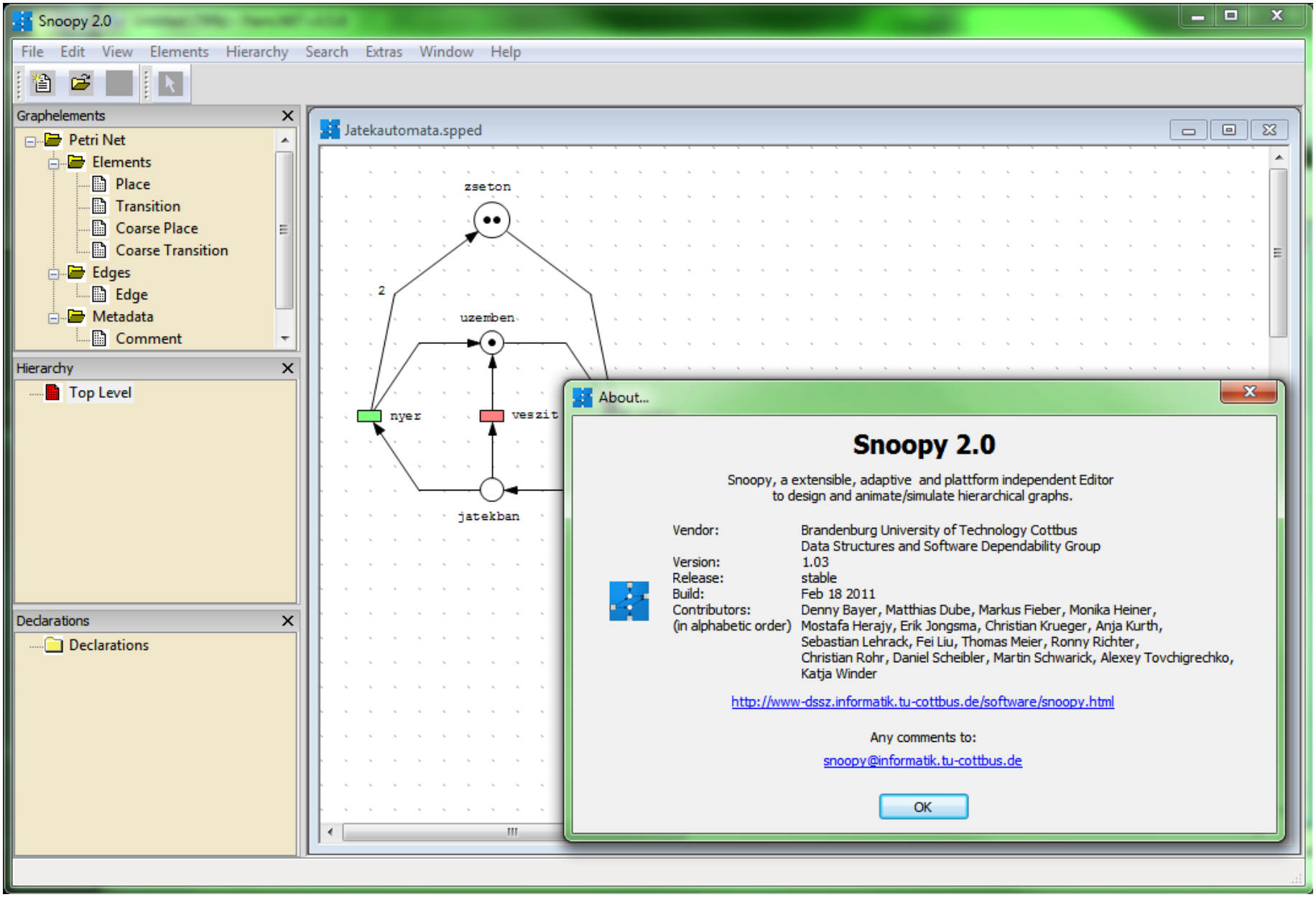
T invariants:

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

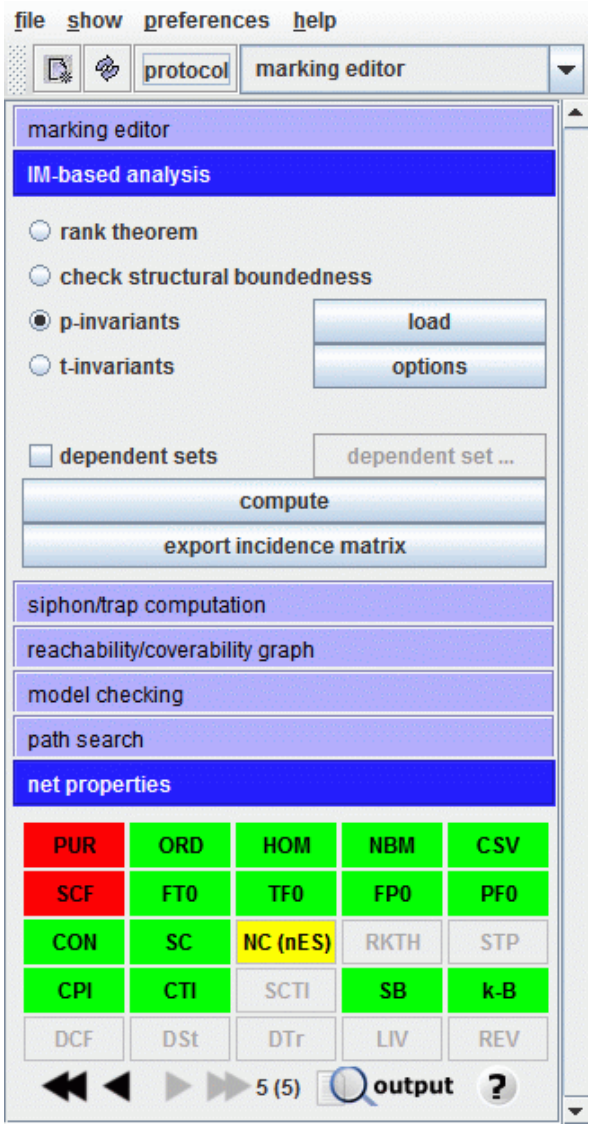
- Snoopy (Windows, Linux)
  - Grafikus szerkesztő + token game (animált)
  - Egyszerűen kezelhető
  - Kényelmi funkciók: copy / paste, undo / redo
  - Kiterjesztések: tiltó él, olvasó él, reset él, egyenlőség él
  - Számos háló típus, többek között színezett háló is
  - Támogatja hierarchikus Petri hálók készítését
  - Elemek színezése, méretezése, élsúlyok kijelzése
  - On-line help
  - Külső analízis eszköz: Charlie (Java)

A Snoopy modellező program képe



Analízis eszközök Snoopy-hoz

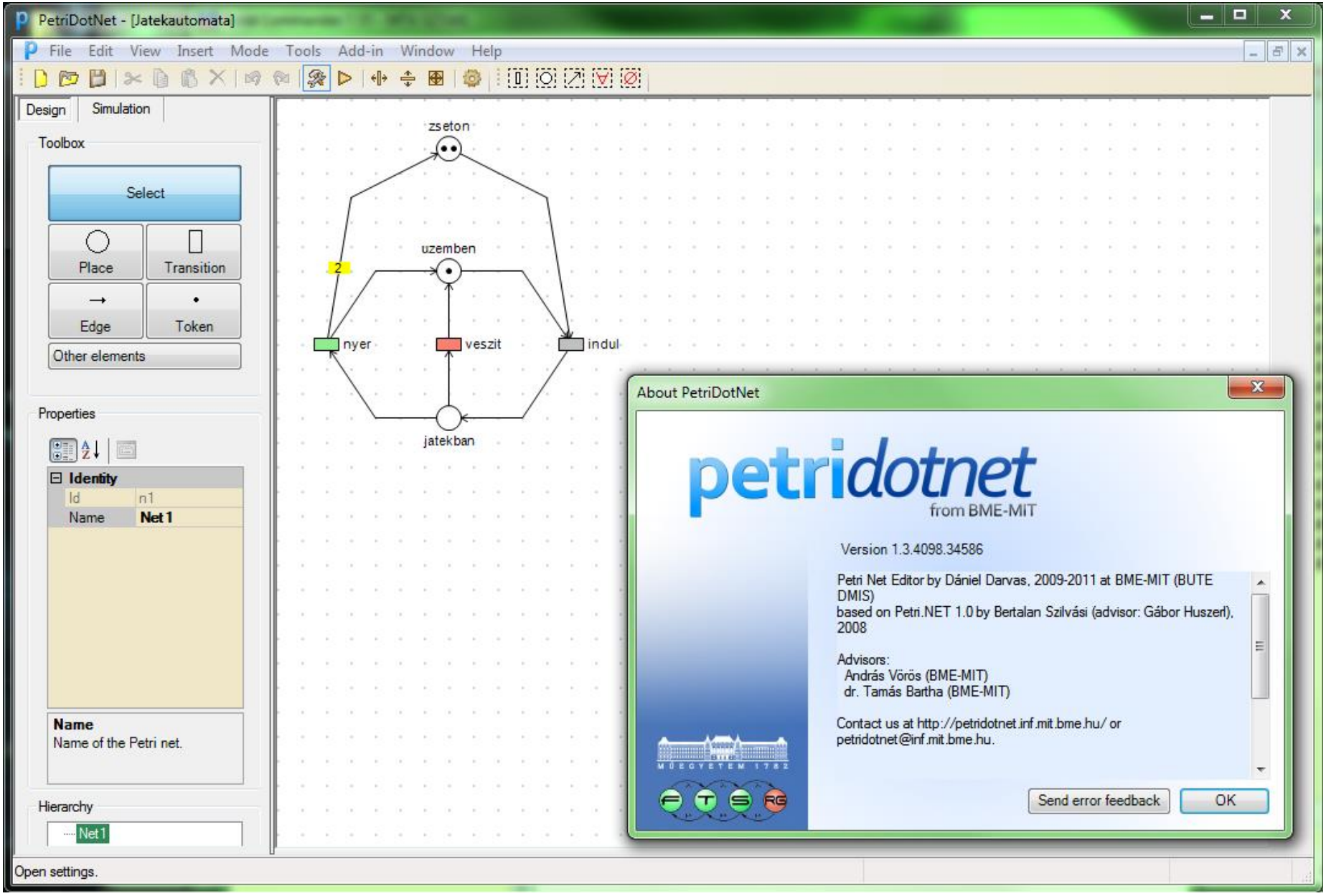
- Charlie (Java)
  - Dinamikus tulajdonságok, elérhetőség
  - Strukturális tulajdonságok, invariánsok
  - Explicit CTL és LTL modellellenőrző
- INA (Windows, Linux)
  - Szöveges felületű parancssori program
  - Token game (szöveges)
  - Invariáns analízis, elérhetőségi gráf generálás
  - Strukturális 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



A PetriDotNet analízis képességei

A(z) AlterBit háló tulajdonságai

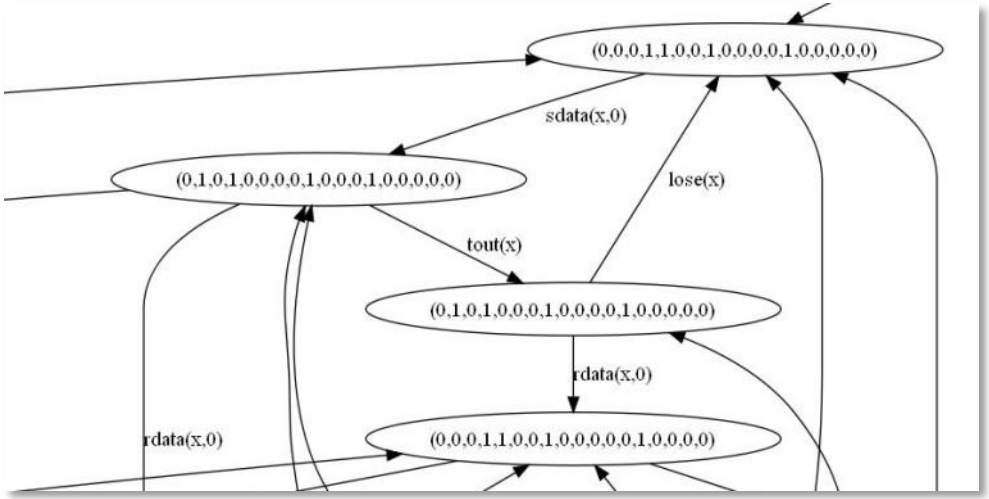
Dinamikus tulajdonságok

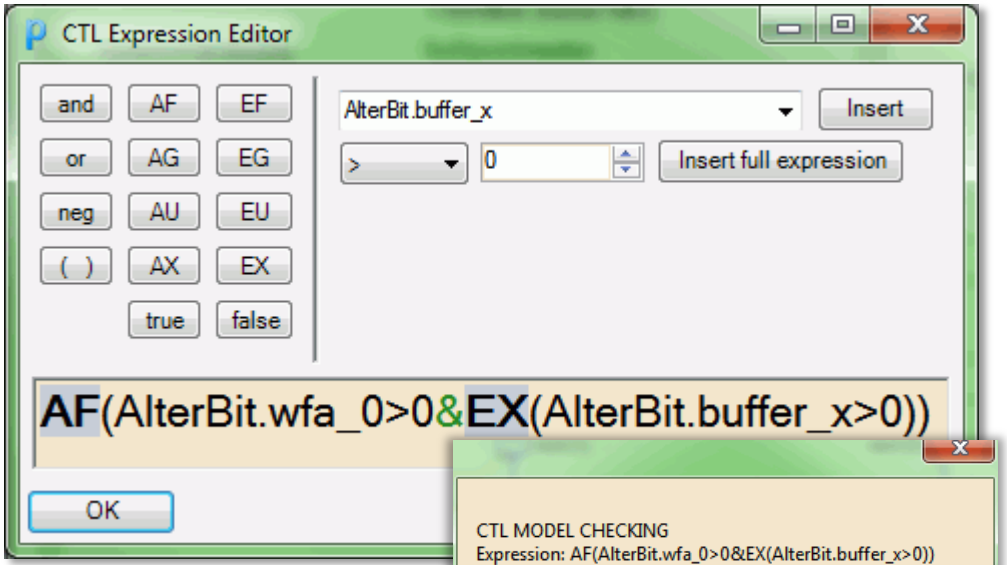
Állapotok száma:	108
Korlátosság:	**korlátos**
	1-korlátos (biztos háló)
Holtpontmentesség:	**holtpontmentes**
Megfordíthatóság:	**megfordítható**
Perszisztencia:	**nem perzisztens**

Strukturális tuladjonságok

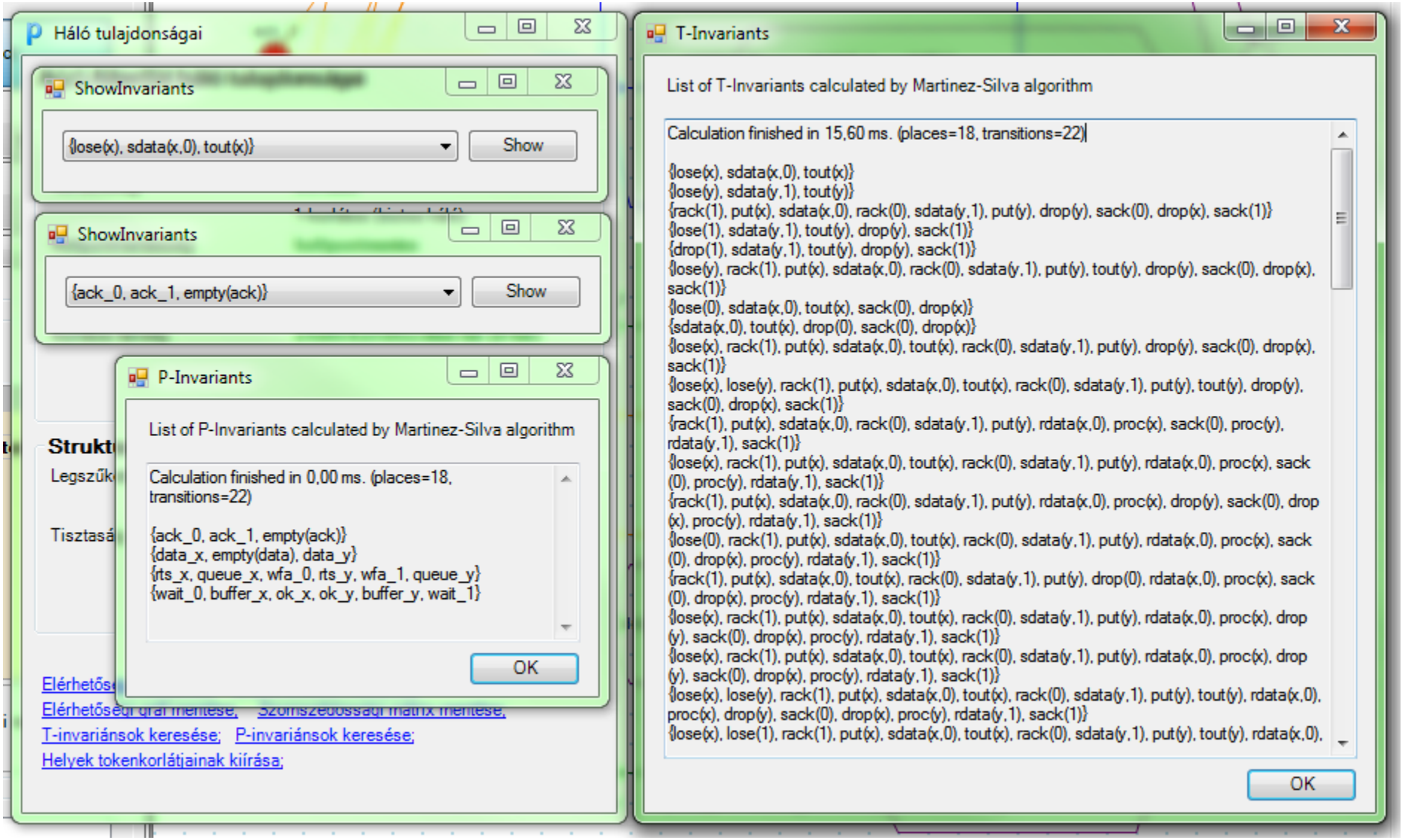
Legszűkebb alosztály:	Petri-háló
Tisztaság:	**nem tiszta (van hurokél)**

Elérhetőség vizsgálata; CTL-kifejezés vizsgálata;
Elérhetőségi gráf mentése; Szomszédossági mátrix mentése;
T-invariánsok keresése; P-invariánsok keresése;
Helyek tokenkorlátjainak kiírása;





A PetriDotNet invariáns analízis



A modellezés alapelvei

A rendszermodellezés célja

- Informatikai rendszerek: általában jól tagoltak
  - Rendszerépítés a komponensek integrációjával
    - Lépések, folyamatok, szálak, …
  - Elemi komponensek kapcsolata:
    - Explicit logikai kapcsolat: sorrendiség, ok-okozati függőség
    - Implicit függőség: pl. osztott erőforrás használata
- Modell alapú analízis: minőségi vagy/és mennyiségi
  - Kvalitatív: logikai helyességbizonyítás
  - Kvantitatív: teljesítményelemzés, megbízhatóság, rendelkezésre állás, biztonságosság analízise

A modellépítés folyamata

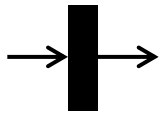
- A három fő modellelem-fajta:
  - Tevékenységek, illetve ezekből álló folyamatok
  - Erőforrások (beleértve: adatok, üzenetek, csatorna)
  - Interakciók a folyamatok és erőforrások között
- Modellezés: hierarchikus és funkcionális
  - Alulról felfelé:
    Elemi tevékenységek -> (Összetett tevékenységek ->)
    Részfolyamatok -> Összetett folyamatok
- Lépések:
  - Egyedi modellelemek felvétele
  - Integrálás

Rendszermodellezés tipikus lépései

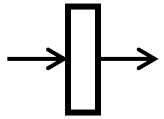
1. A folyamat modellje (az erőforrás használat, illetve üzenetváltás részletes feltüntetése nélkül)
2. Az erőforrások modellje
   - A foglalt/szabad/... állapotot jellemző véges automata modellrész
   - Az üzenetek tárolója (ha szükséges)
3. Integrálás: A folyamat és erőforrás modelljében a megfelelő állapotátmenetek összevonása
   - Pl.: Foglalás összevonva a szabad $\rightarrow$ foglalt átmenettel
   - Pl.: Üzenetküldés az üzenetet a csatornába teszi

Tevékenységek modellezése Petri-hálókban

- Elemi tevékenység: tranzíció tüzelése
- Felhasznált erőforrások: bemeneti / kimeneti helyek
- Végrehajtási idő
  - determinisztikus
  - sztochasztikus



determinisztikus időzítésű tranzíció



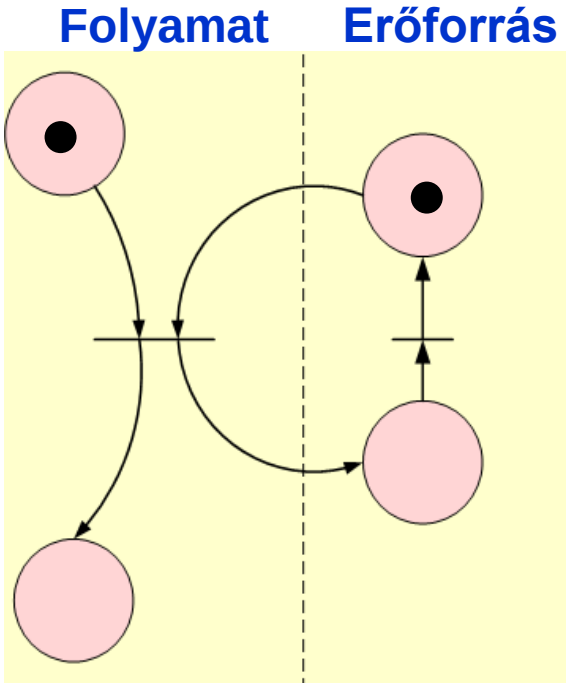
exponenciális időzítésű tranzíció

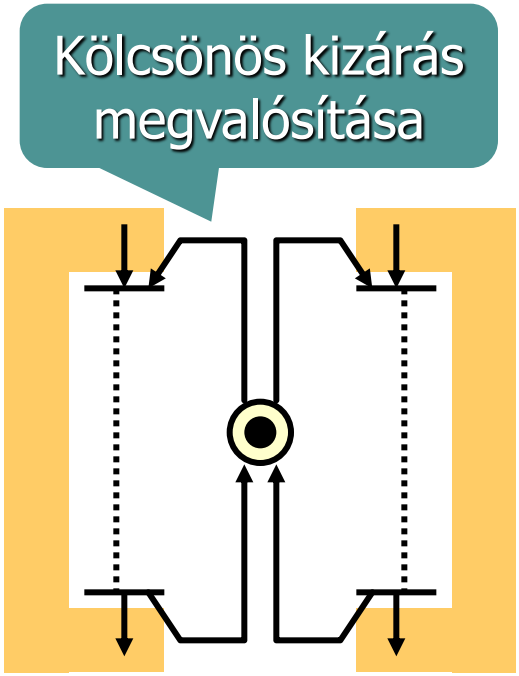
Az engedélyezettséggel kapcsolatban felmerülő kérdések:

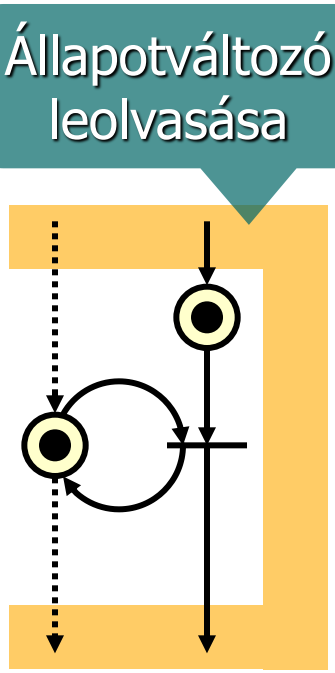
- Időzítetlen tranzíciók előbb tüzelnek (nagyobb „prioritás”)
- Engedélyezettség megszűnése: mi lesz a tüzelési idővel?
  - Újraindul (újra sorsol): „újrakezdi” a tevékenységet
  - Megőrzi (már sorsoltat): „folytatja” a megkezdett tevékenységet

Példa: Erőforrás allokáció modellezés

- Szükséges erőforrás foglalása
- Kölcsönös kizárás
- Korlátos kapacitású erőforrás igénybe vétele

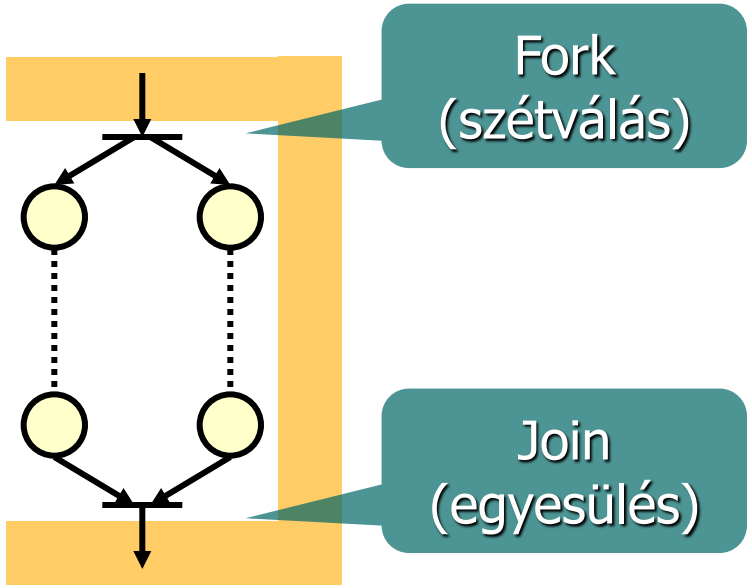


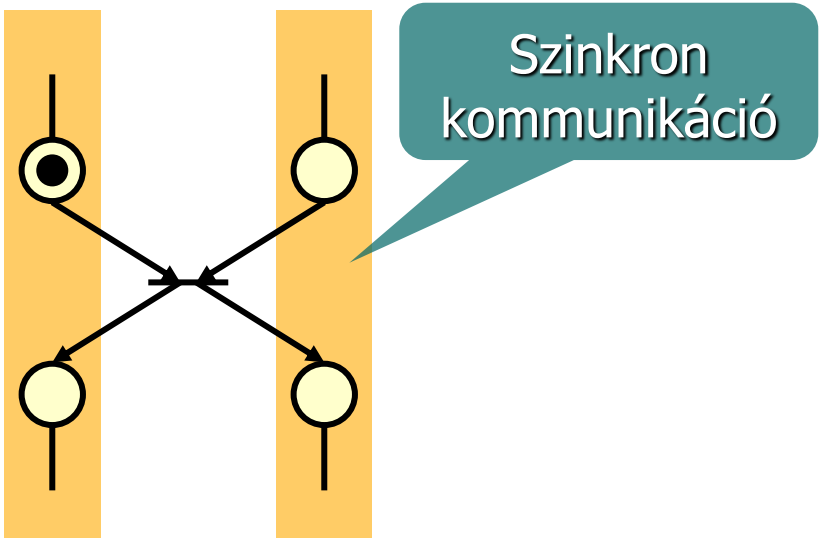


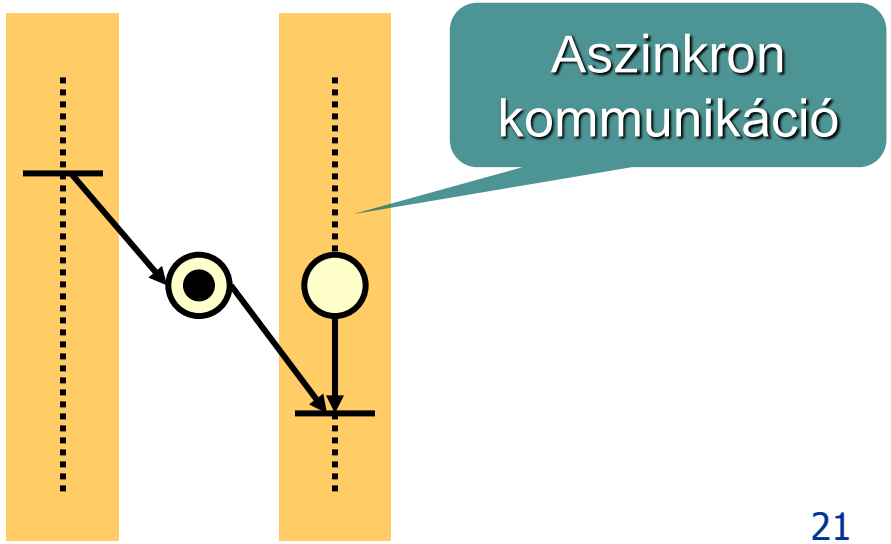


Példa: Folyamatok közötti kapcsolatok

- Párhuzamosság
  - Fork és join
- Szinkron kommunikáció
  - Egymás bevárása
- Aszinkron kommunikáció
  - Levelesláda jellegű



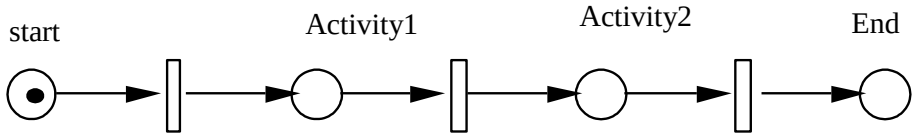




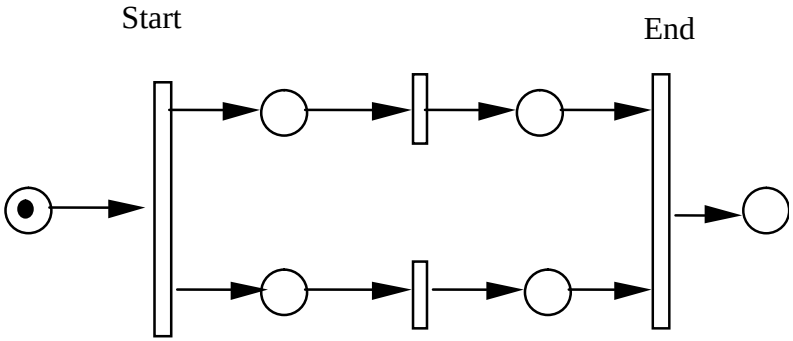
Példa: Gyártócella modellezésének modellezési mintái

Feldolgozási folyamatok

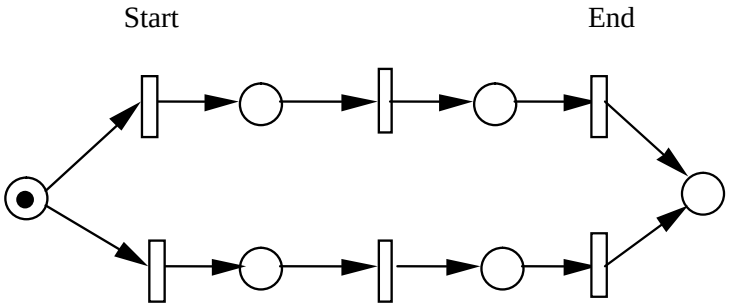
- Szekvenciális feldolgozás:



- Párhuzamos feldolgozás:

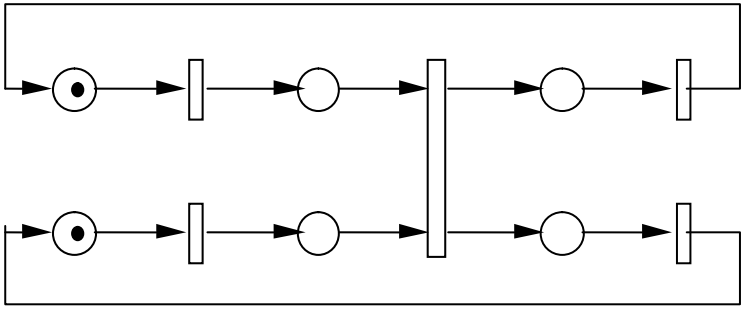


- Alternatív feldolgozás:

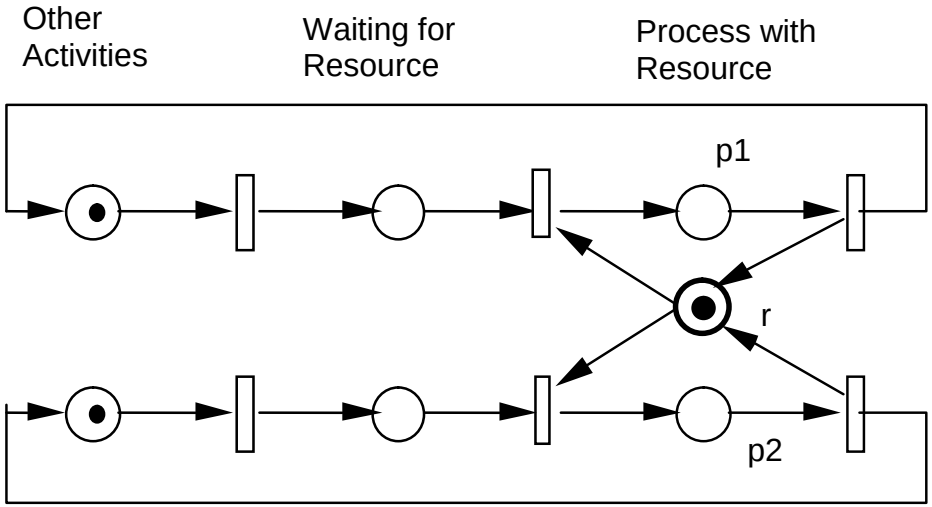


Interakciók

- Szinkronizálás:

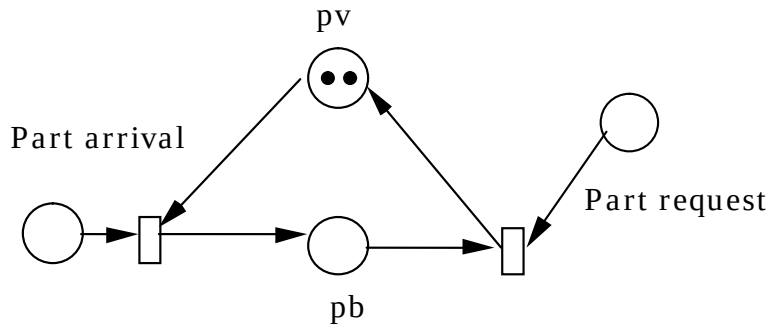


- Megosztott erőforrás:

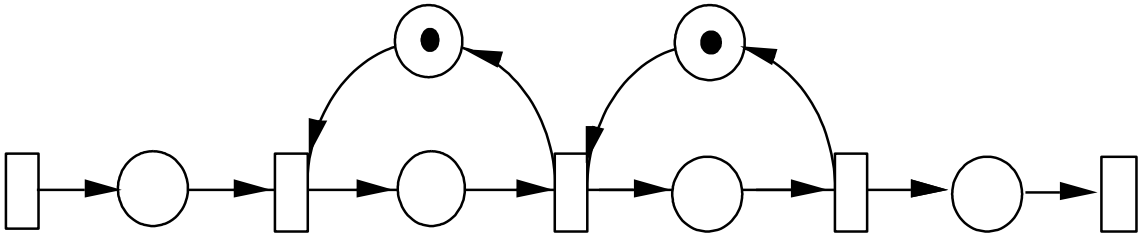


Tárolók feldolgozáshoz

- Véges kapacitású tároló:

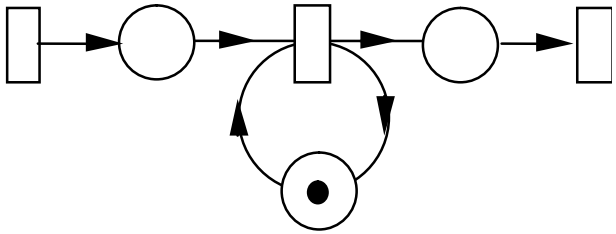


- FIFO tároló:

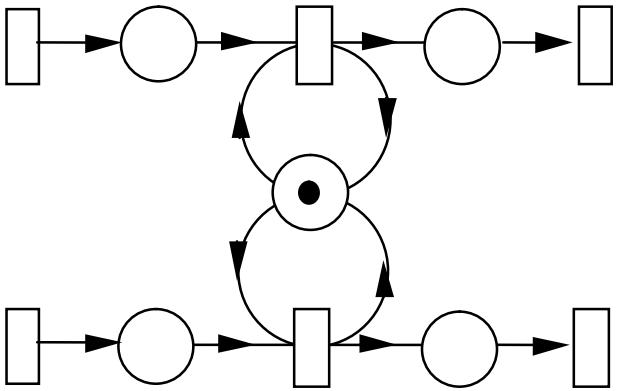


Gépek használata

- Feldolgozás dedikált géppel:

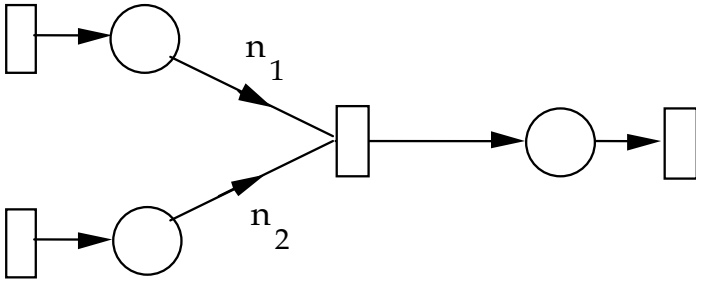


- Feldolgozás megosztott géppel:

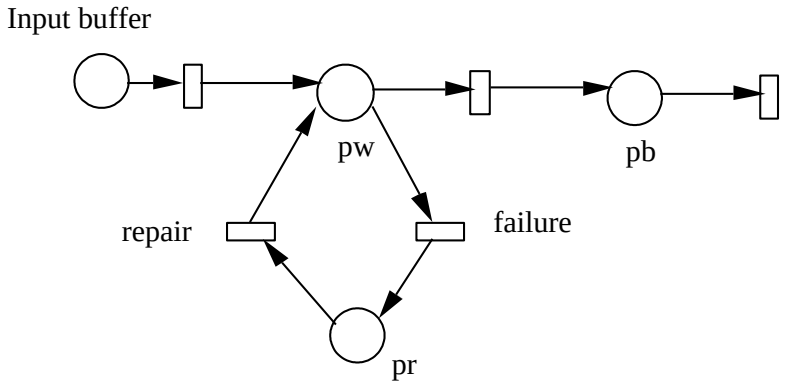


Szerelés

- Alkatrészek összeszerelése:

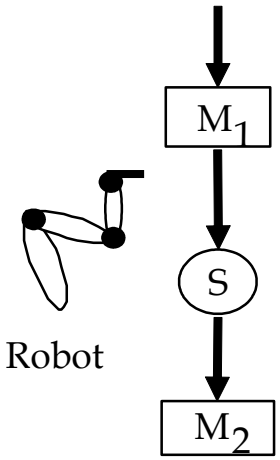


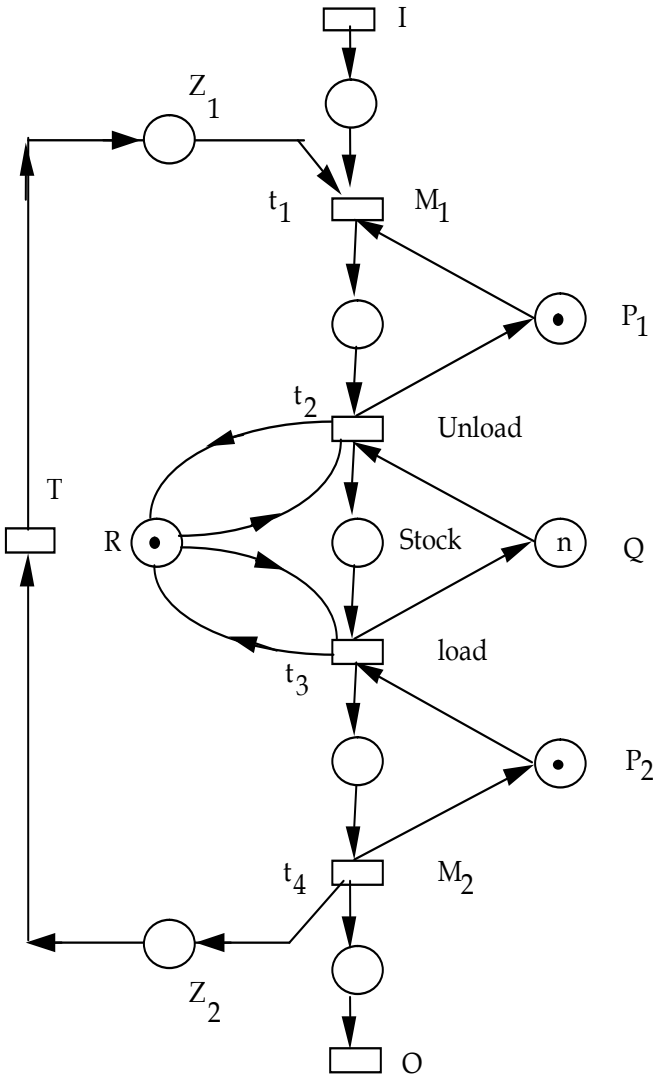
- Meghibásodás a feldolgozás során:



Robotcella

- Tevékenységek
- Tárolók (kapacitáskorlát)
- Erőforrások
- Ciklikus működés





Példa Petri háló modellek készítésére: 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

- Üzenetekhez egy ellenőrző bitet kapcsol
- Az üzenetek megérkezését a vevőtől nyugta jelzi, ugyanazzal az ellenőrző bittel
- Ha a küldött üzenethez csatolt bit $\mathbf{b^0}$, akkor
  - ha az üzenet elvész, a küldő időtúllepéssel észleli a nyugta hiányát $\rightarrow$ újra küldi
  - ha a küldő $\mathbf{b^0}$ bittel ellátott nyugtát kap (ilyet várt), akkor a következő üzenethez $\mathbf{b^1} = \neg\, \mathbf{b^0}$ negált bitet csatol
  - ha a küldő $\mathbf{b^1} = \neg\, \mathbf{b^0}$ bittel ellátott nyugtát kap (pedig $\mathbf{b^0}$ bittel ellátott nyugtát várt), akkor eldobja (majd időtúllépés lesz a nyugta hiánya miatt)

Fogadó folyamat

- Az üzenet vételét nyugtázza, a nyugtával a kapott ellenőrző bitet küldi vissza
- Ha b^0 ellenőrző bittel jelölt üzenetet kap, akkor ezt a b^0 bit visszaküldésével nyugtázza, majd
  - 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, de nyugtát küld (mondván, hogy bizonyára újraküldés történt a nyugta hiánya miatt)

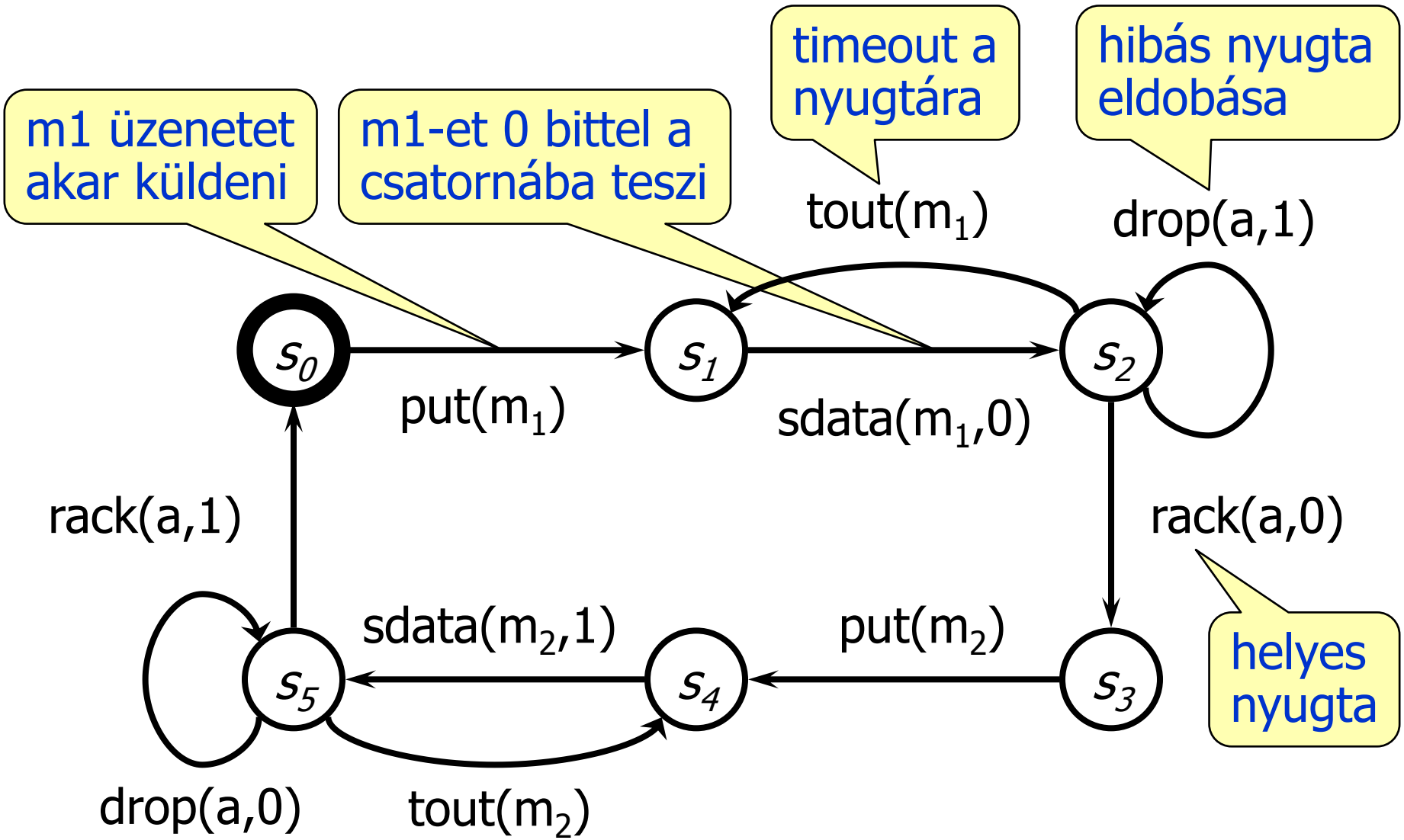
A modellalkotás lépései

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

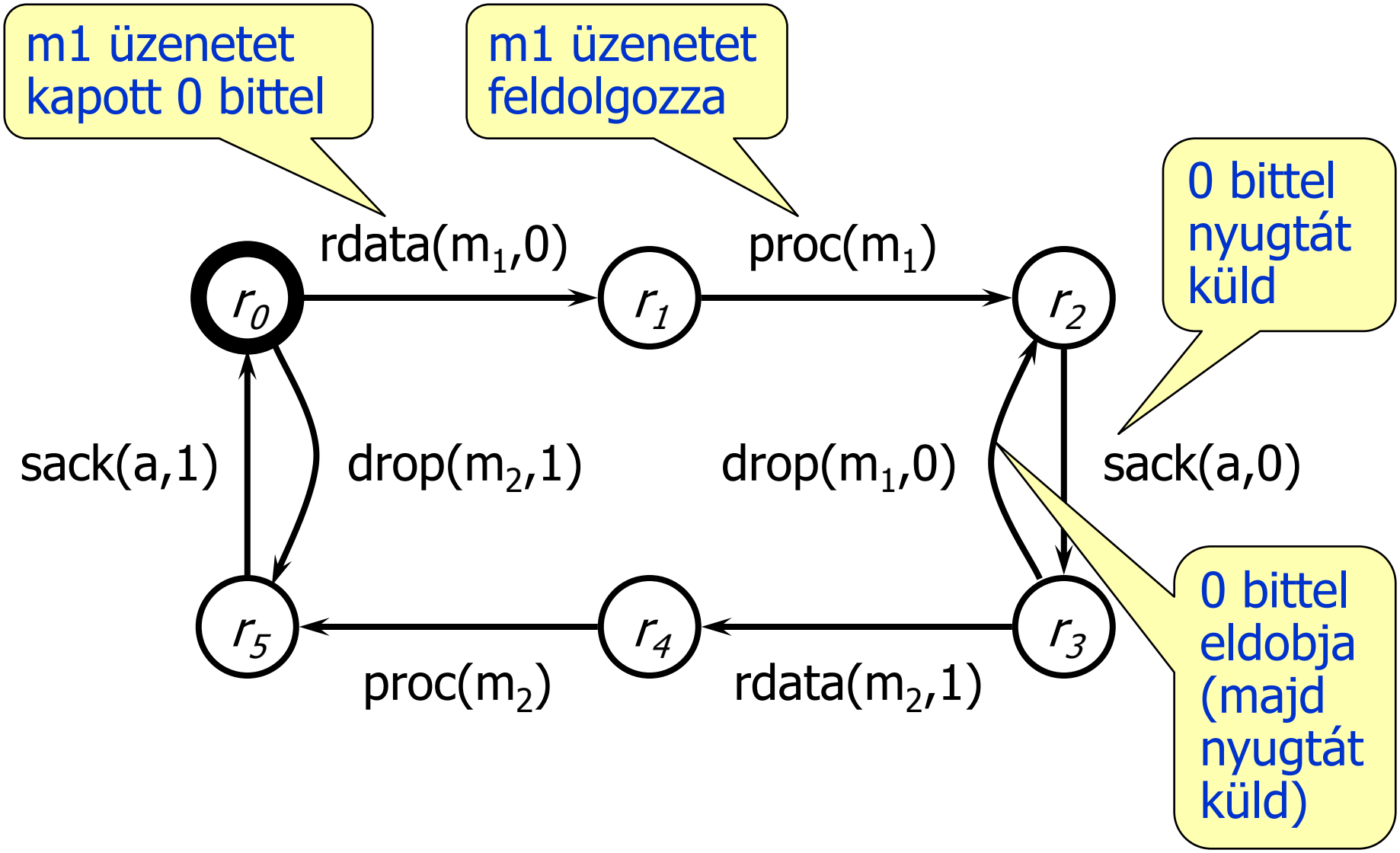
Komponensek és állapotaik

- Komponensek (alrendszerek)
  - 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

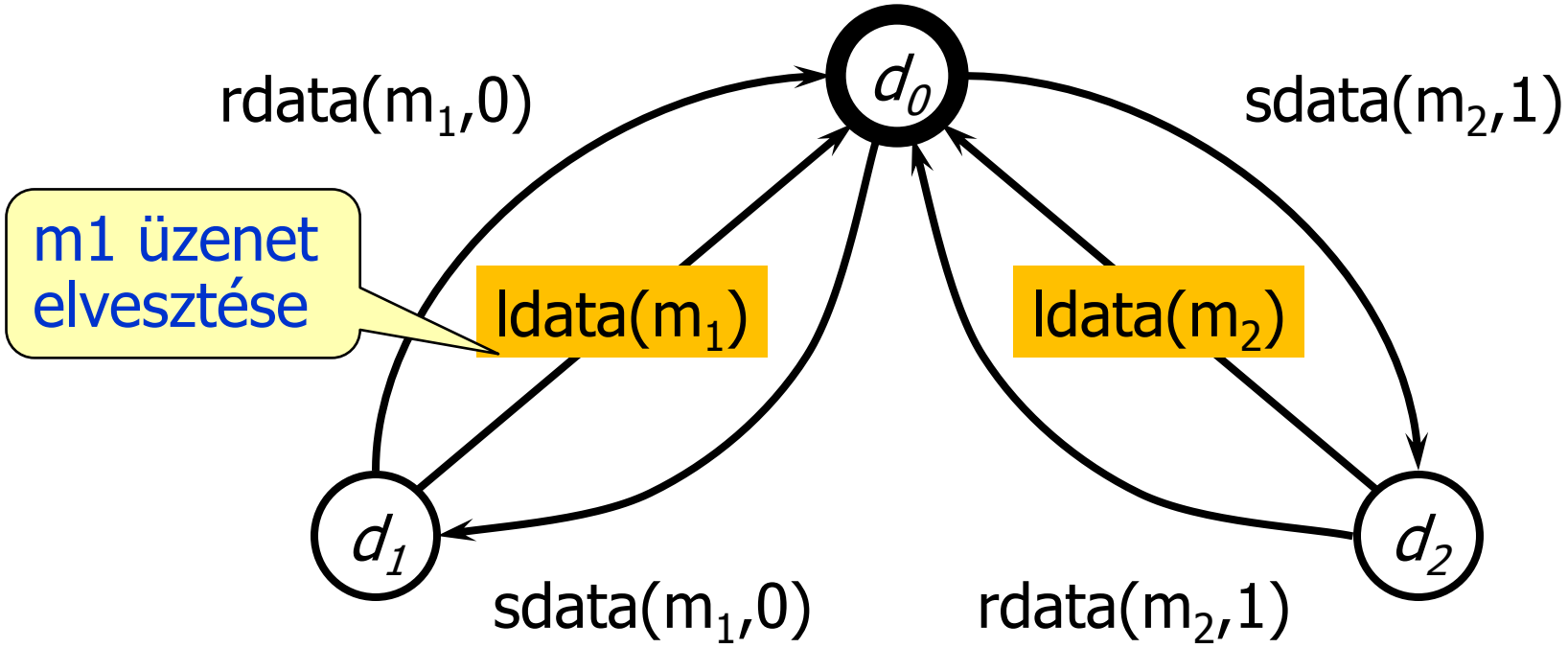
Küldő folyamat állapotgráfja



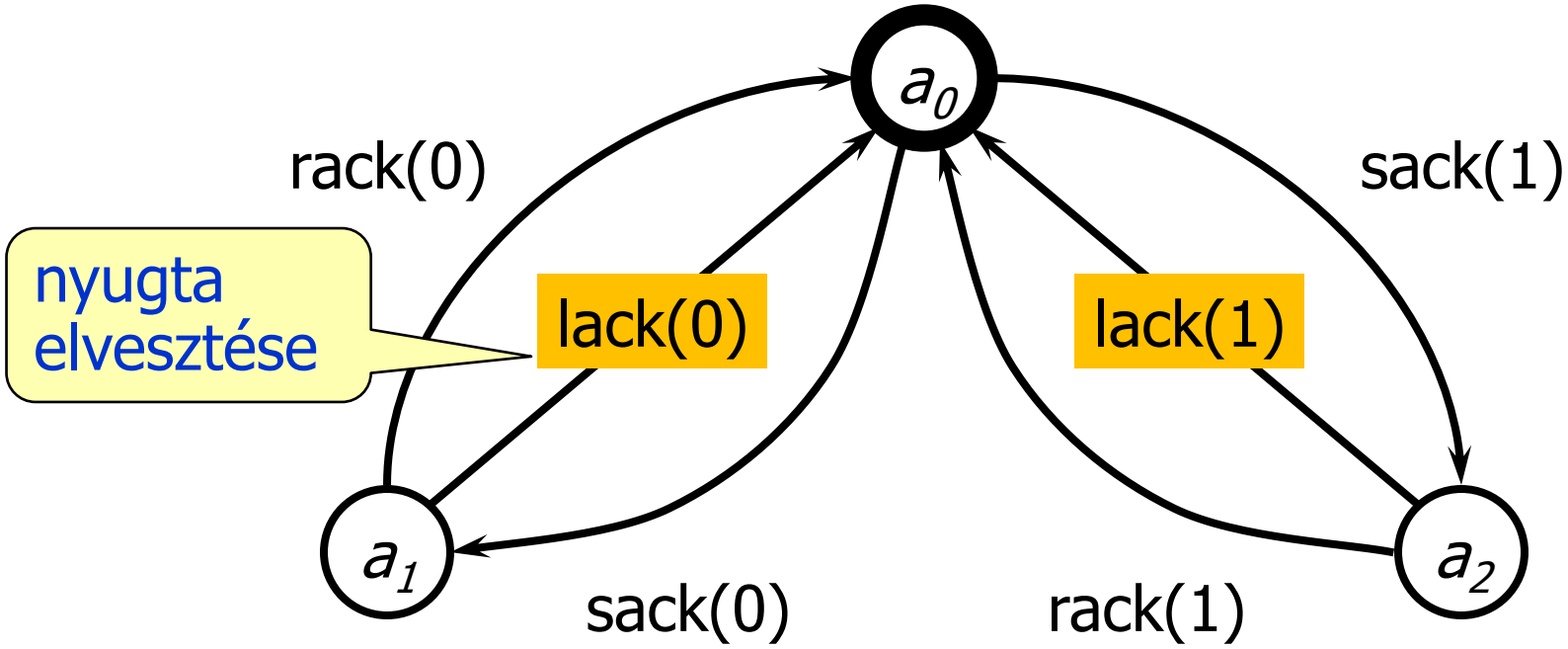
Fogadó folyamat állapotgráfja



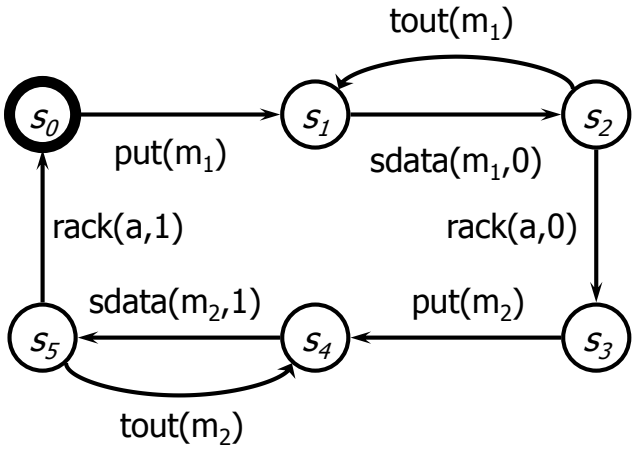
Adat csatorna állapotgráfja

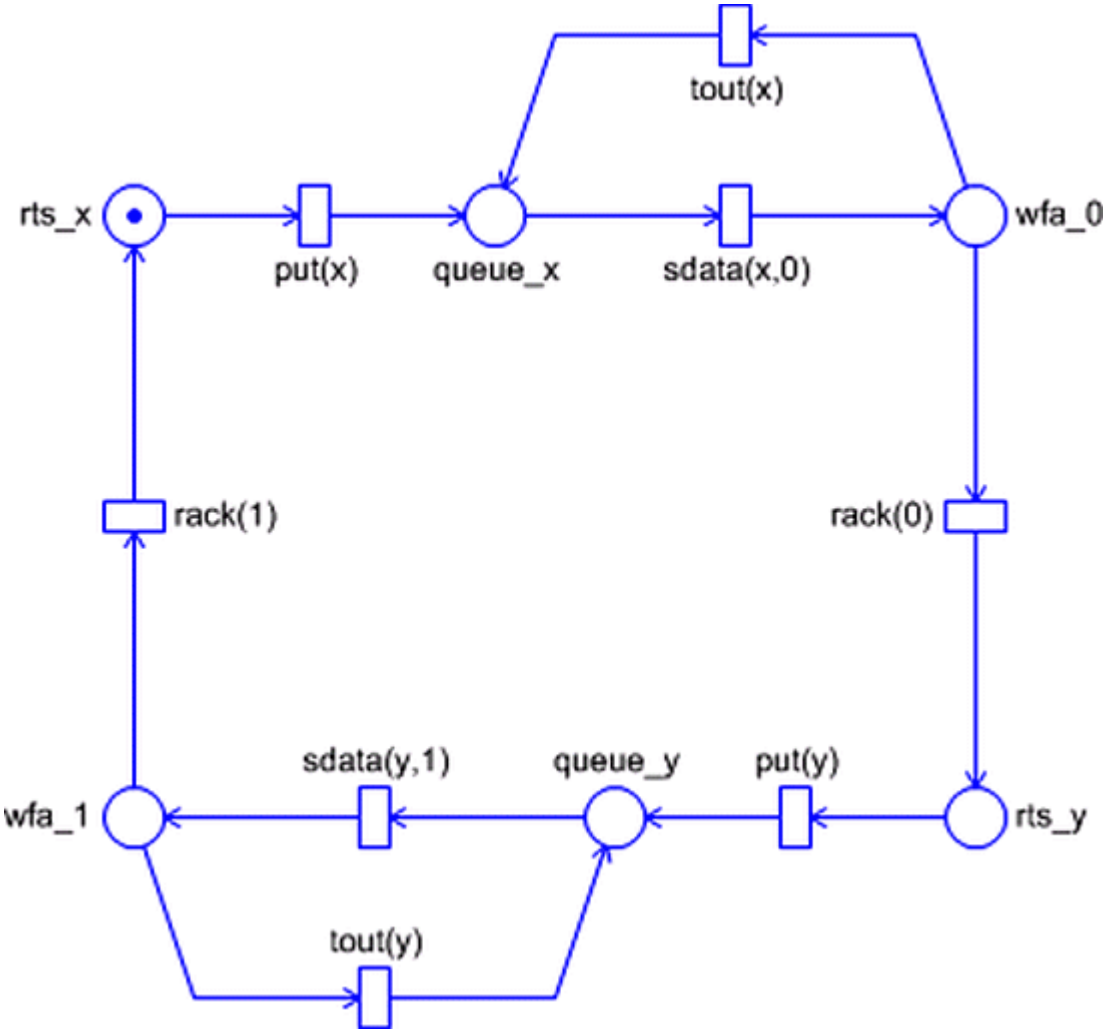


Nyugtázó csatorna állapotgráfja

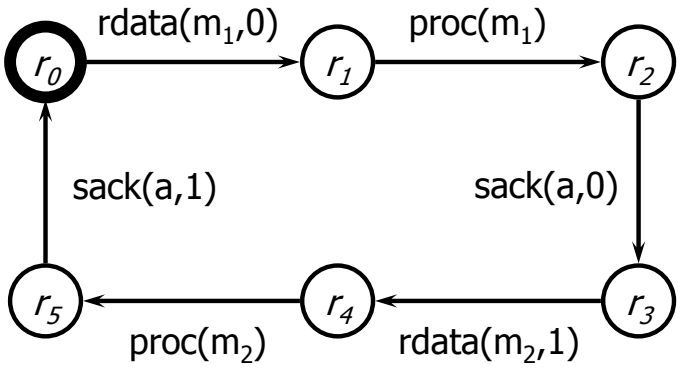


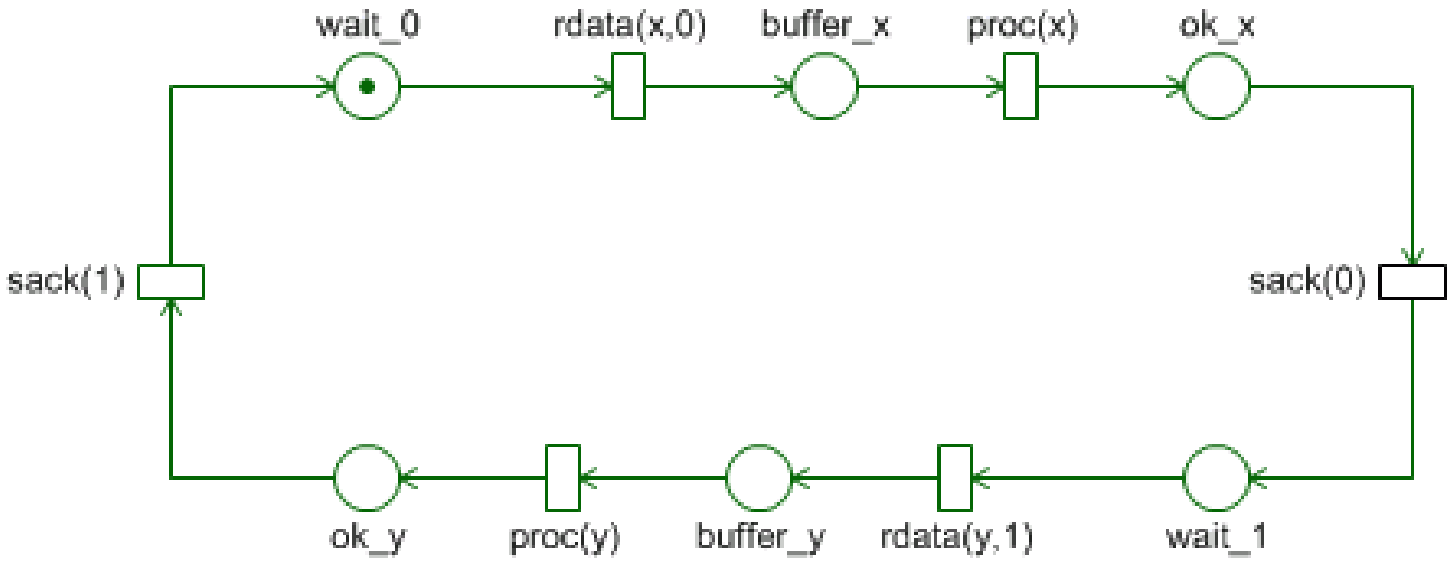
Küldő folyamat Petri háló modellje (fő ciklus)



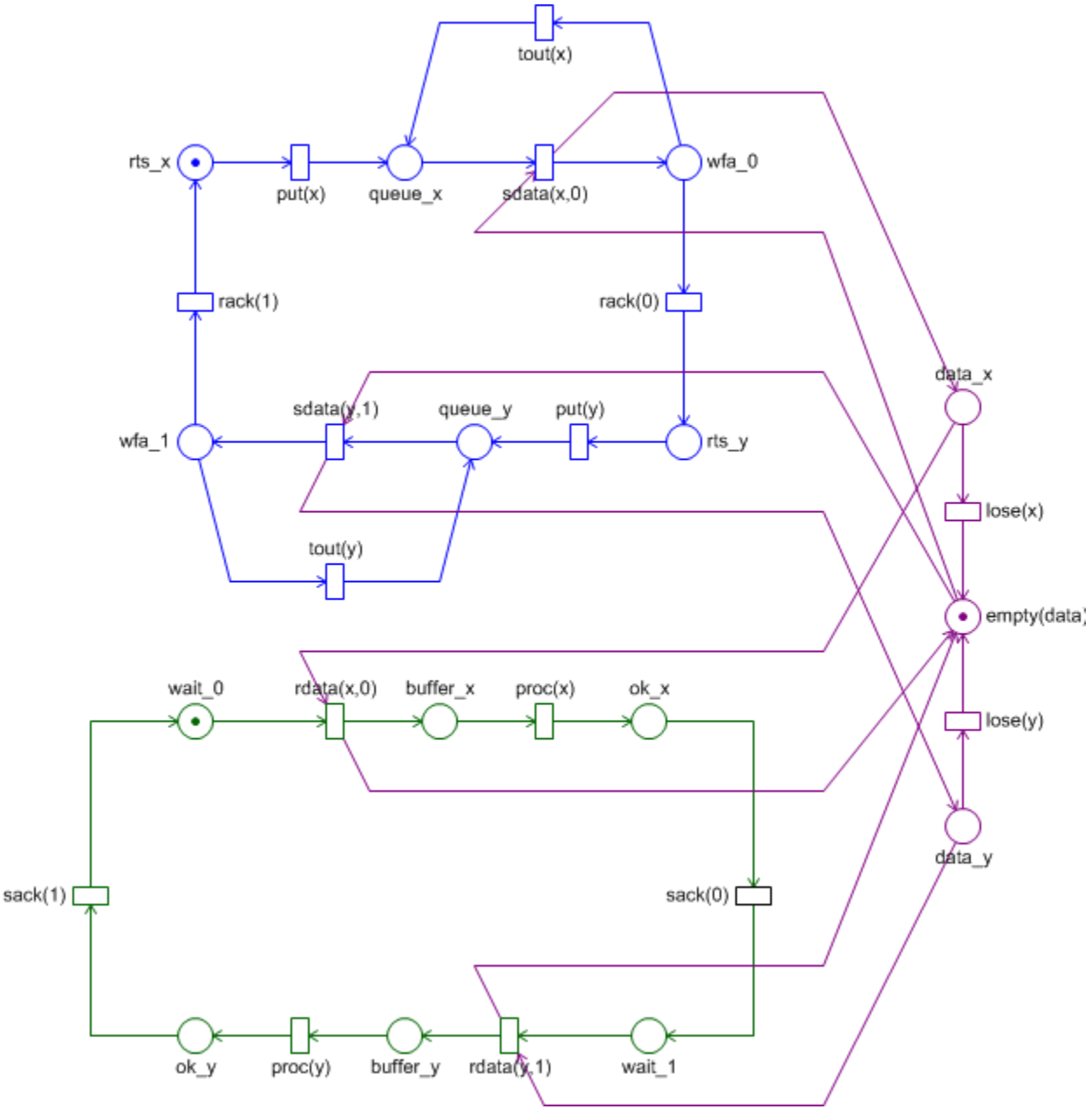


Fogadó folyamat Petri háló modellje (fő ciklus)

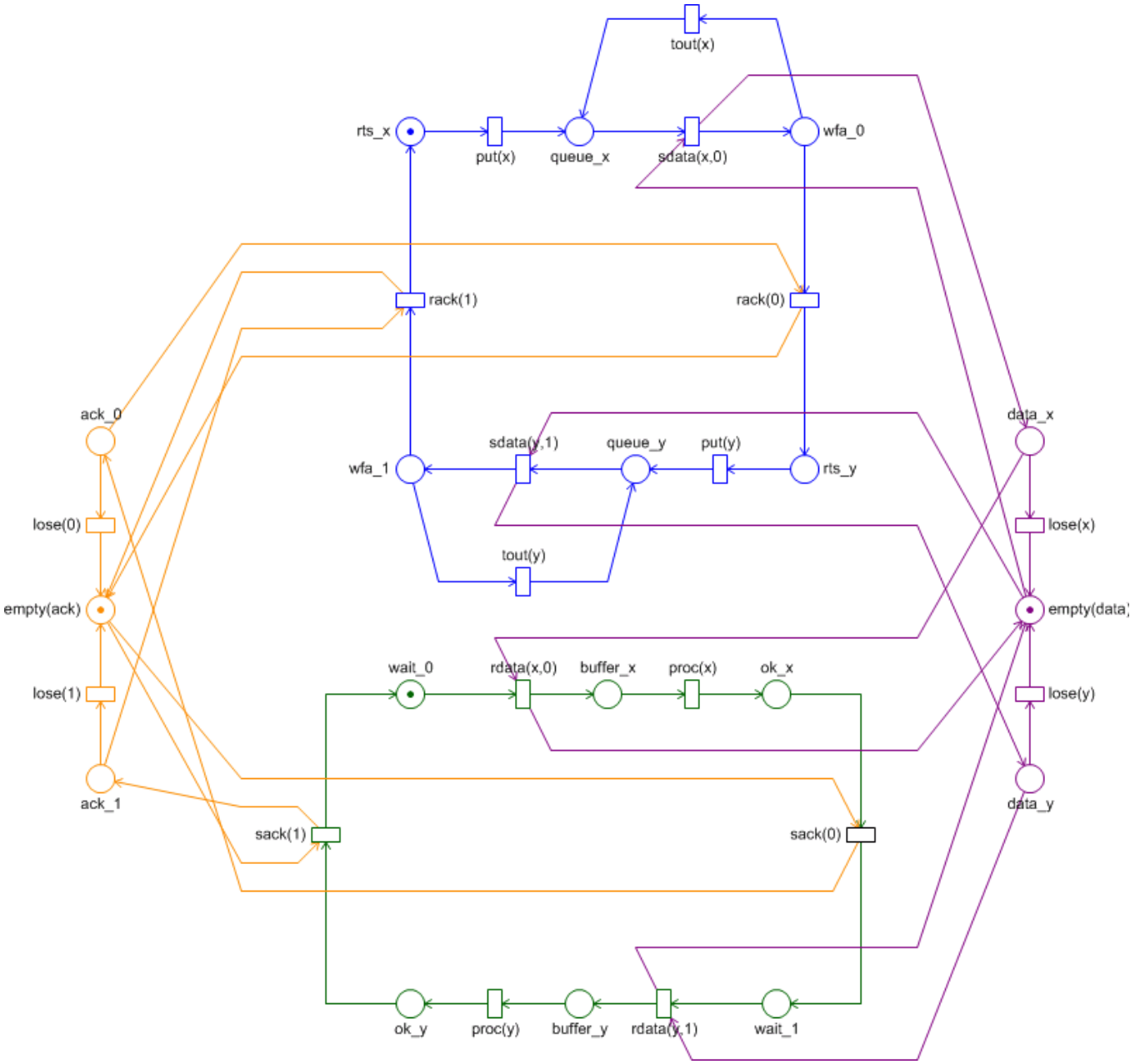




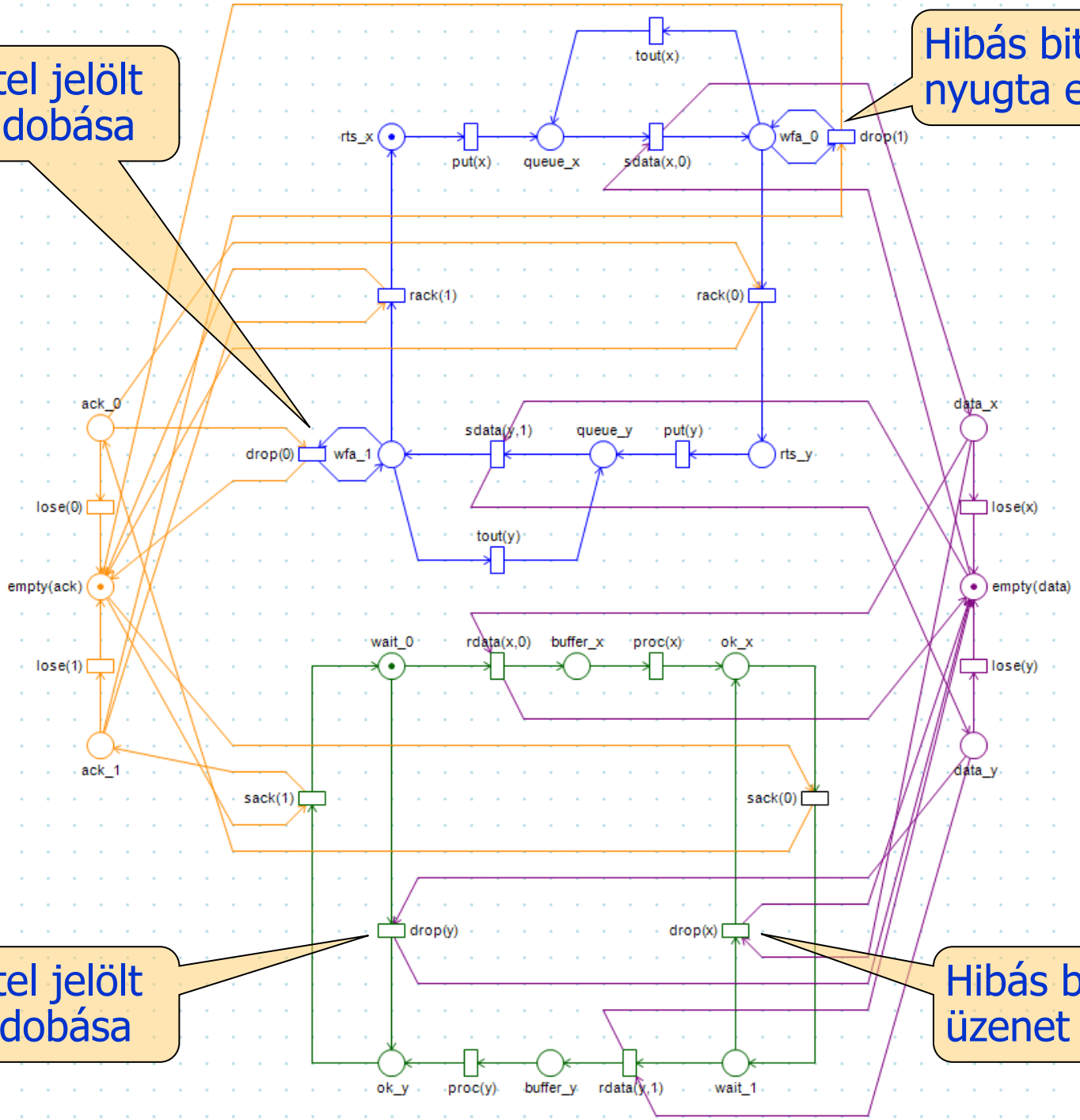
Adat csatorna és adat átvitel (fő ciklus)



Nyugtázó csatorna és nyugtázás (fő ciklus)



Kiegészítések



Hibás bittel jelölt nyugta eldobása

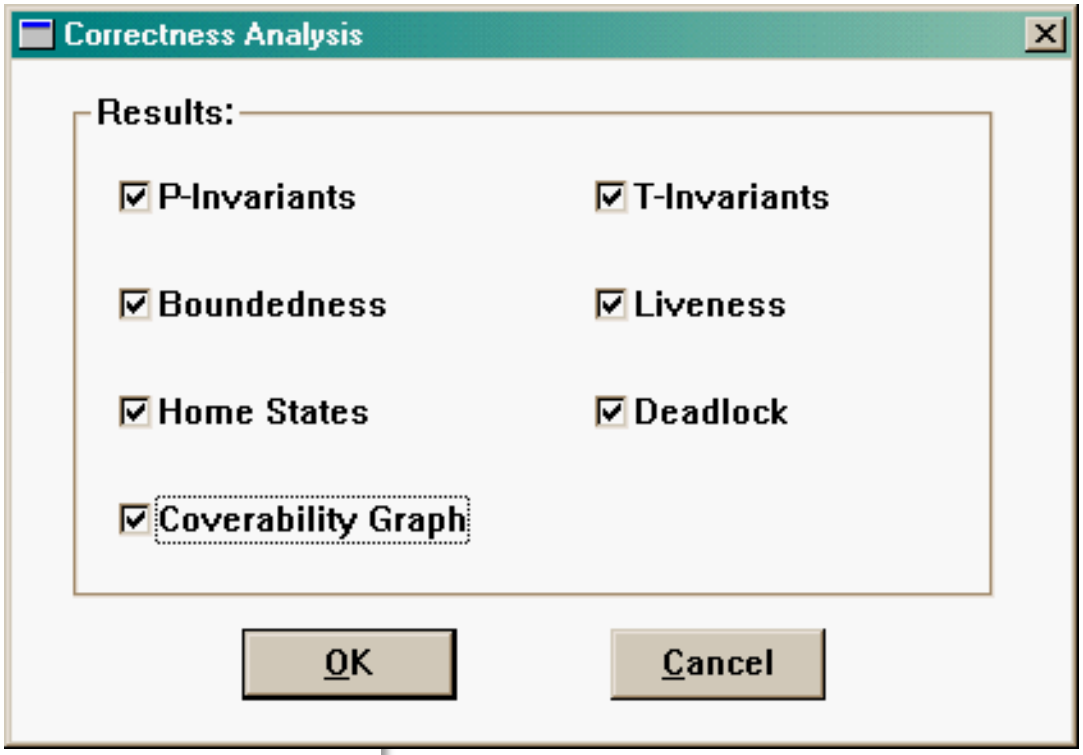
Hibás bittel jelölt nyugta eldobása

Hibás bittel jelölt üzenet eldobása

Hibás bittel jelölt üzenet eldobása

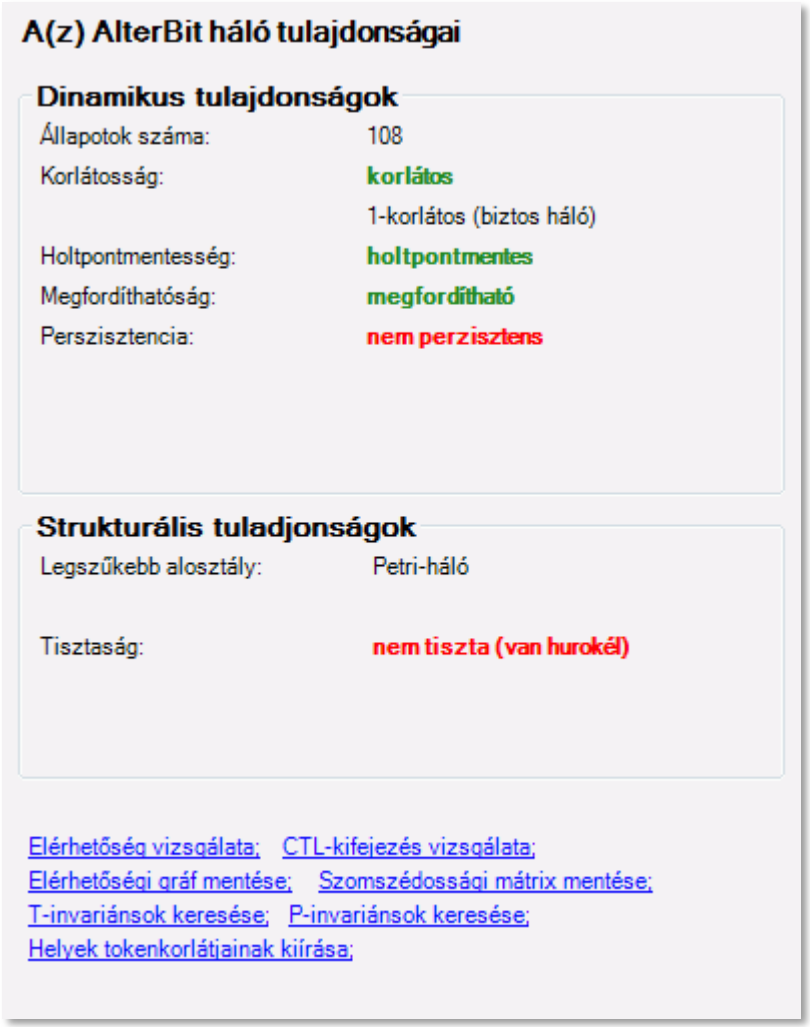
Példa Petri-háló modell analízisére: Alternáló bit protokoll

DNAnet: A modell dinamikus tulajdonságai

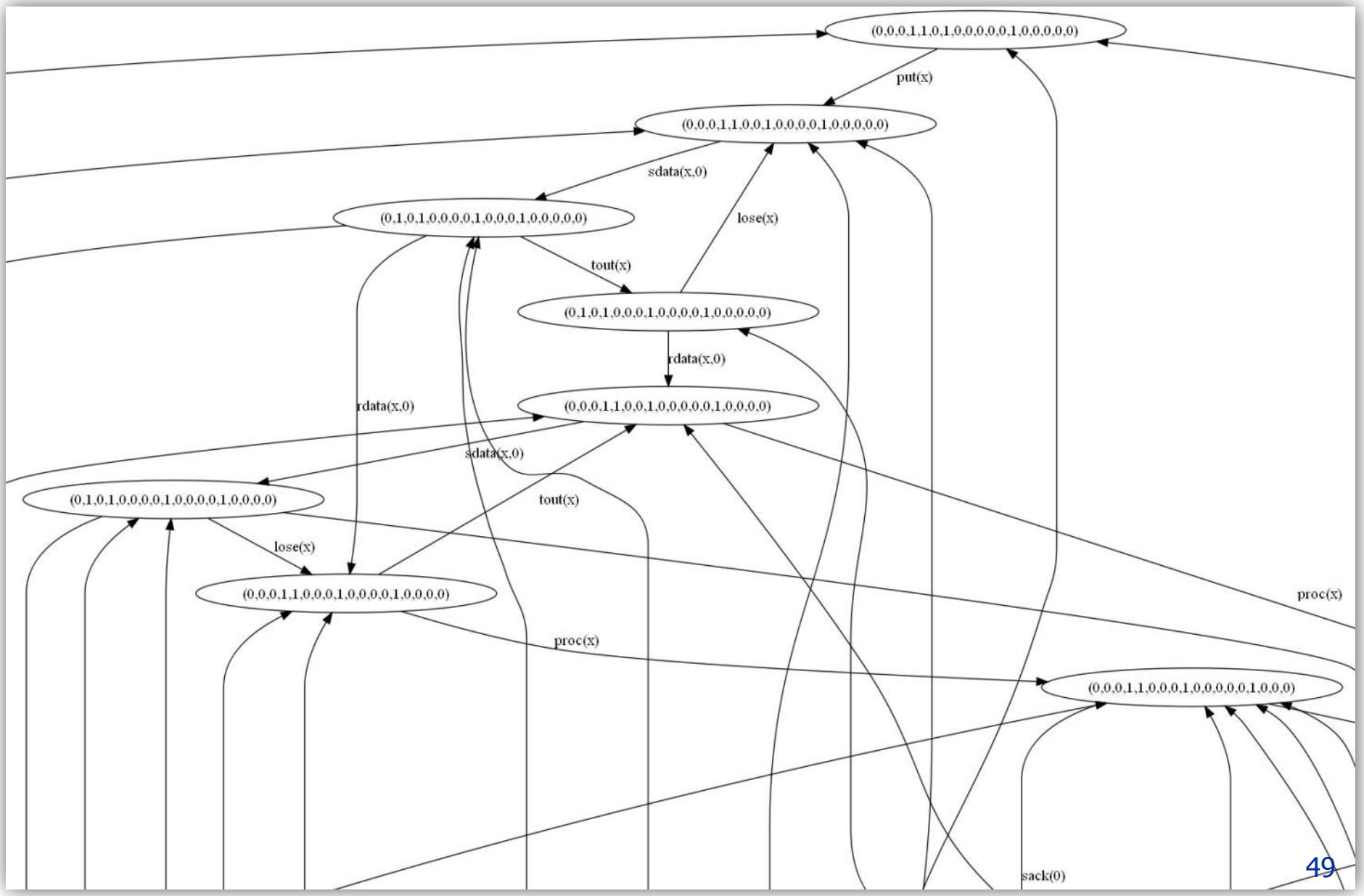



```
Net is bounded.

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
```

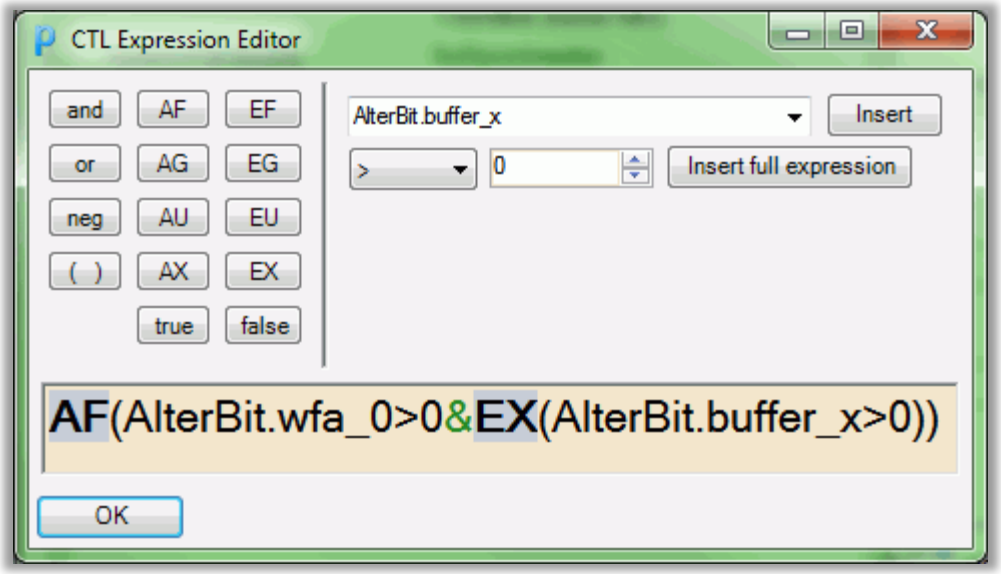
PetriDotNet: A modell dinamikus tulajdonságai

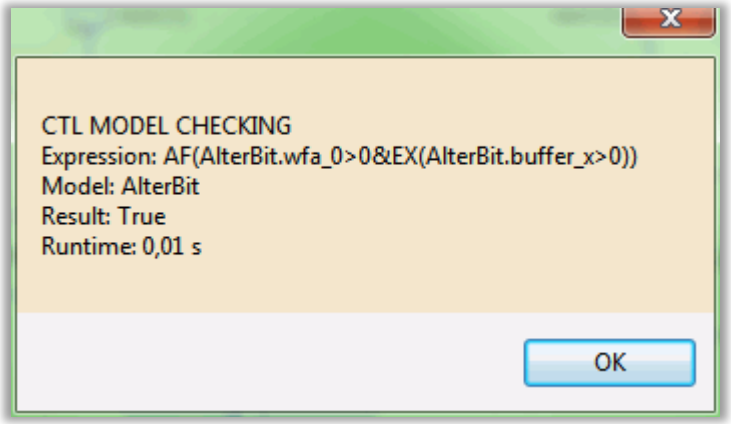


PetriDotNet: Elérhetőségi gráf kirajzolása (GraphViz)



PetriDotNet: CTL modellellenőrzés





AF(AlterBit.wfa_0>0 & **EX**(AlterBit.buffer_x>0)) ⇒ True

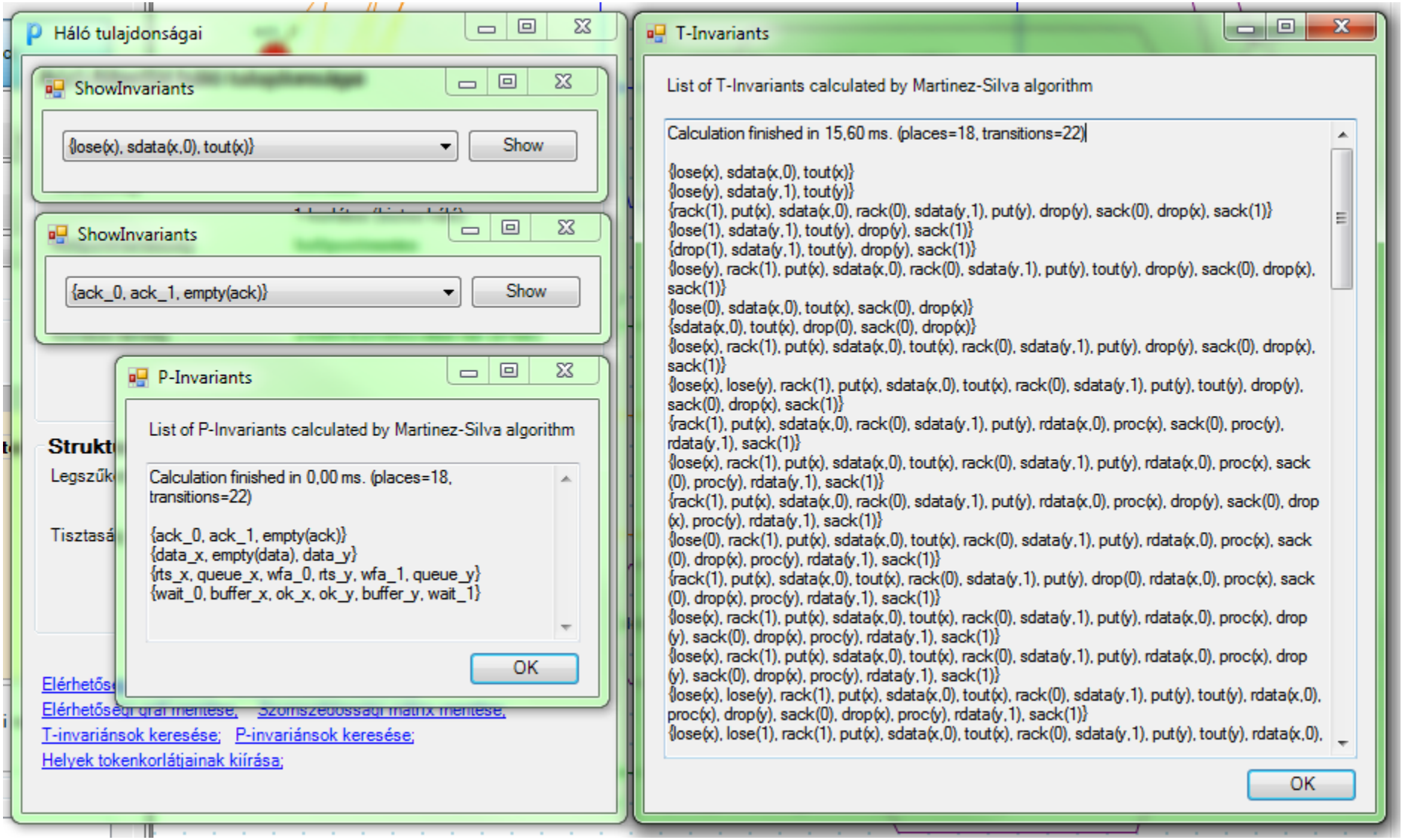
AG(**AF**(AlterBit.buffer_y>0)) ⇒ False

AF(**EG**(AlterBit.buffer_x=0)) ⇒ True

EF(AlterBit.wfa_0>0 & AlterBit.data_x=0) ⇒ True

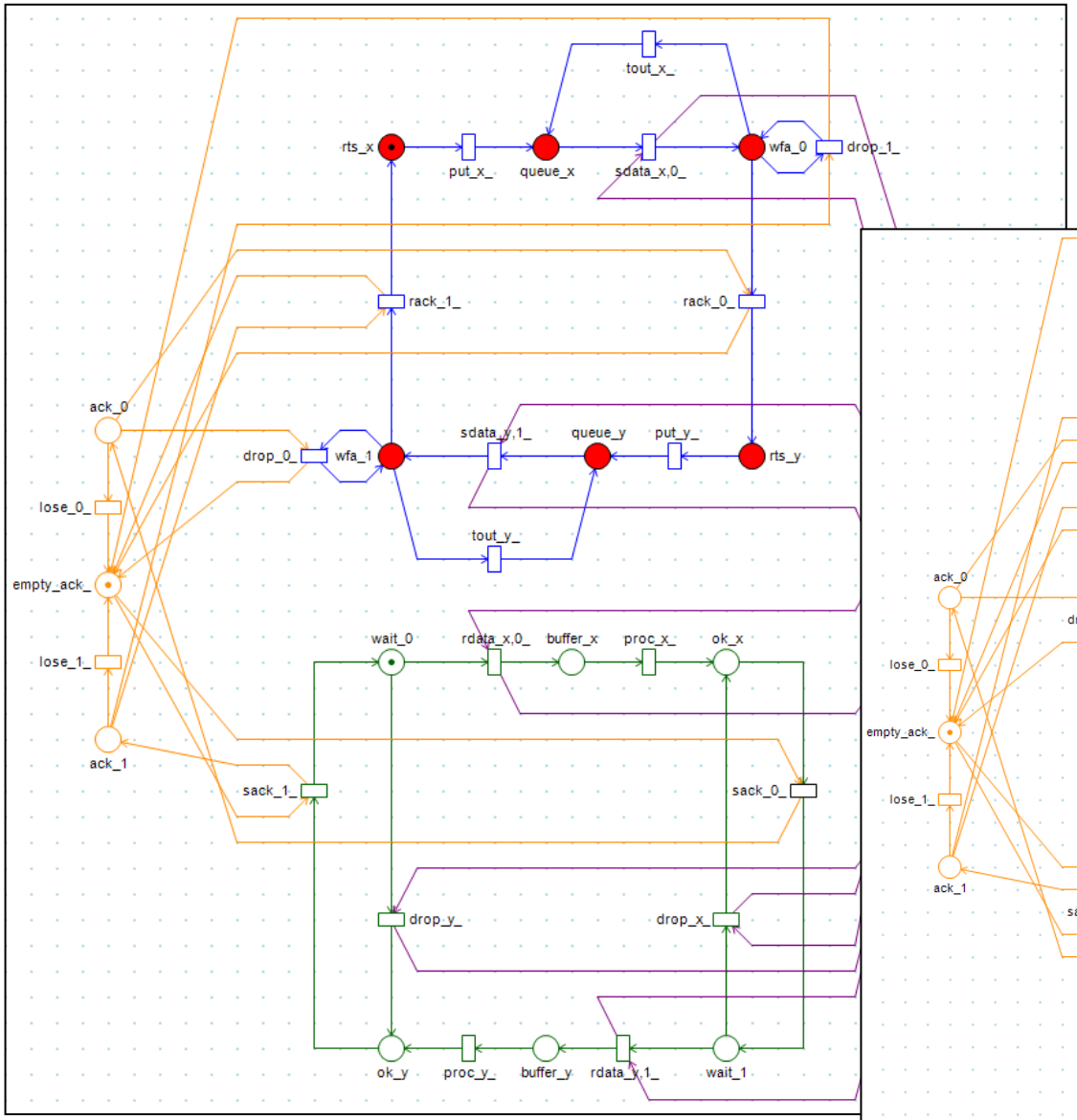
AF(AlterBit.queue_x>0 & **AX**(AlterBit.wfa_0>0 & AlterBit.data_x>0)) ⇒ True

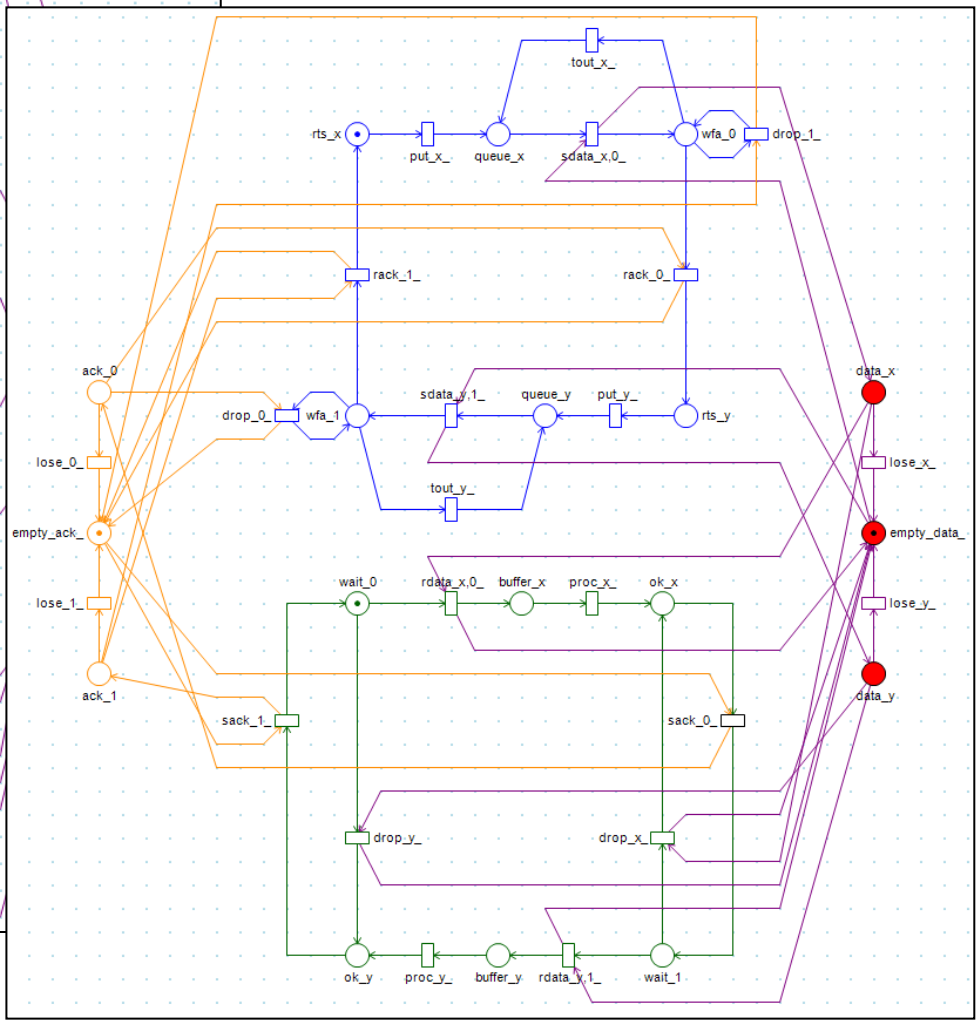
PetriDotNet: Invariáns analízis



PetriDotNet: P-invariánsok (példák)

Komponensek állapotgépei





PetriDotNet: T-invariánsok (példák)

Ciklikus működések (itt: hibátlan, adatvesztés)