

# Petri-háló alapú modellek készítése (Példa a modellezési folyamatra)

dr. Bartha Tamás

BME Méréstechnika és Információs Rendszerek Tanszék

# A modellezési feladat

## Alternating Bit Protocol

- Átviteli protokoll veszteséges csatornához
  - üzenet elveszhet (véges számú alkalommal), nyugta is
  - üzenet tartalma nem változhat
- Két résztvevő
  - Küldő folyamat: üzenetet küld, nyugtát fogad
  - Fogadó folyamat: üzenetet fogad, nyugtát küld
- Cél: a protokoll biztosítsa, hogy minden küldött üzenet véges időn belül eljusson a fogadóhoz

# 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 → 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 → 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

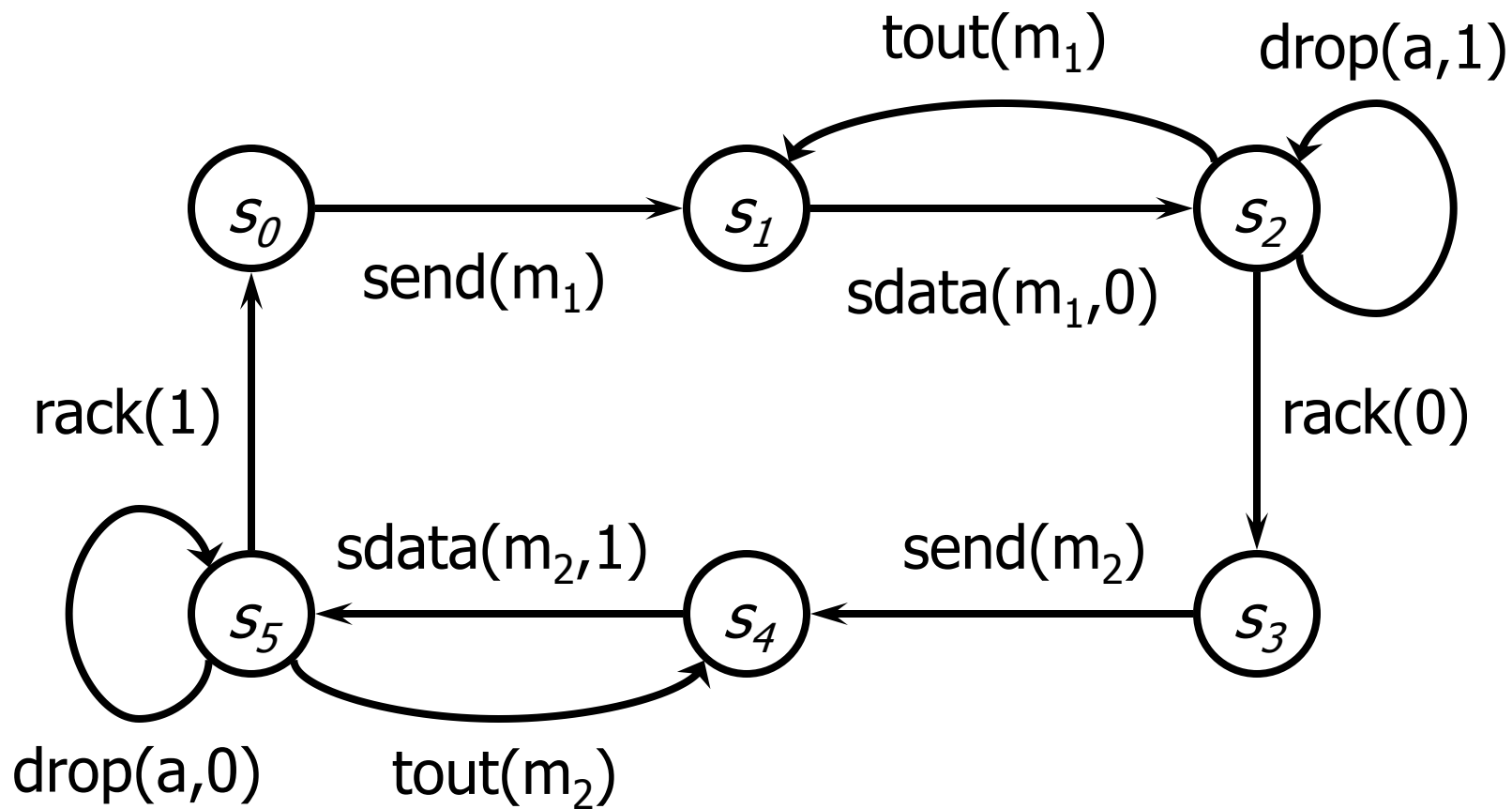
# A modellalkotás lépései

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

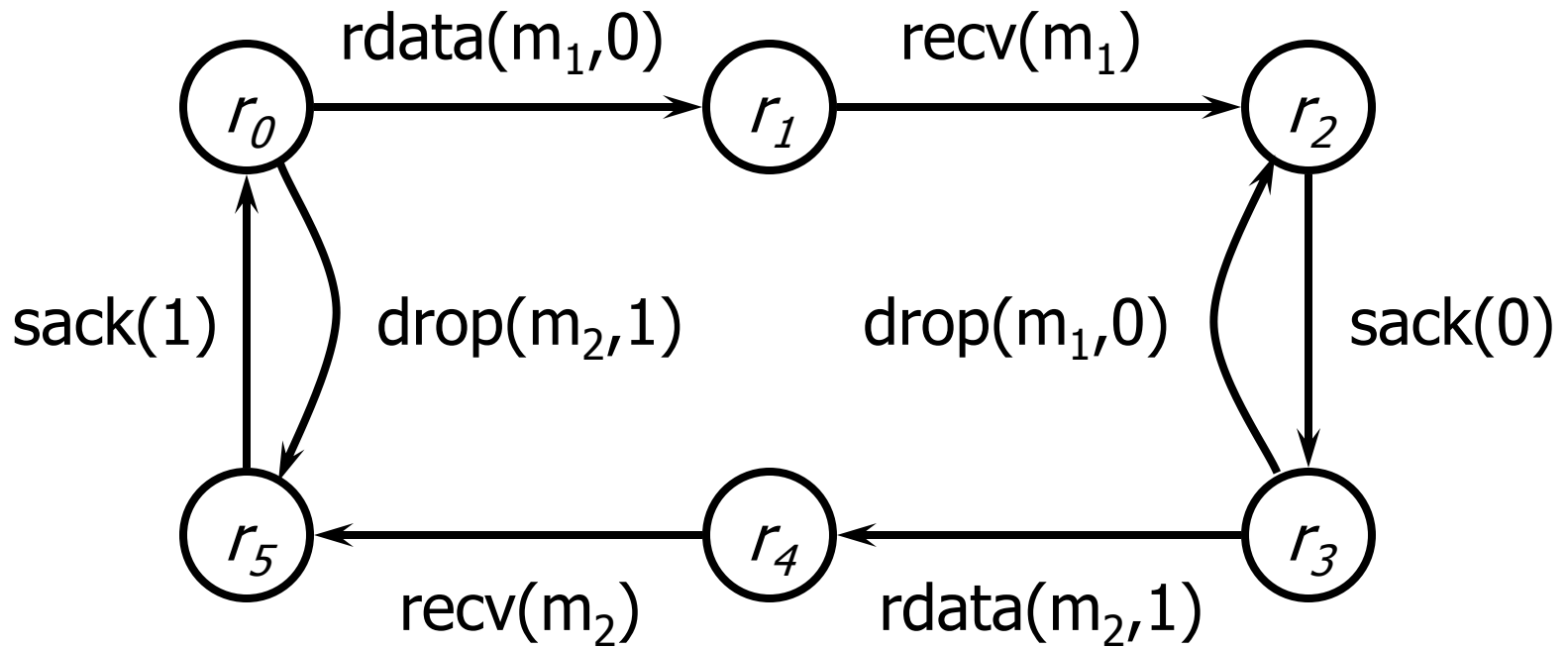
# Komponensek és állapotgráfjaik

- 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

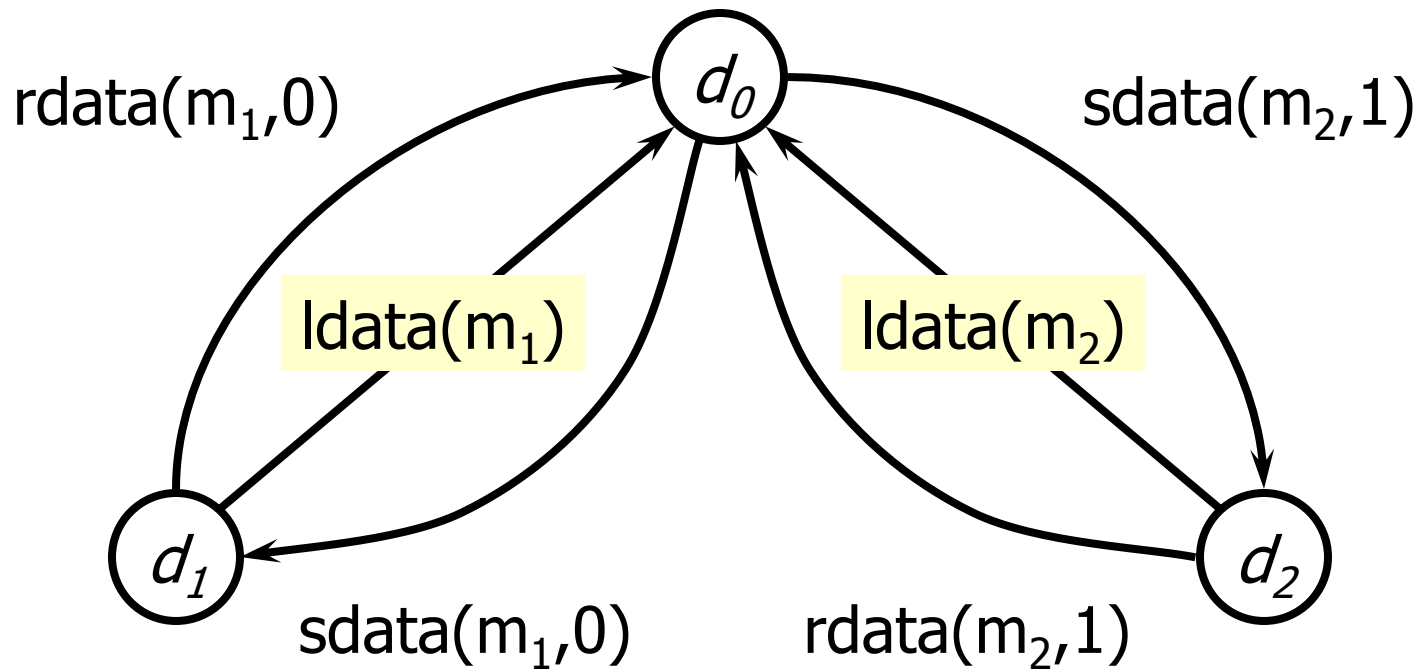


# Fogadó folyamat állapotgráfja

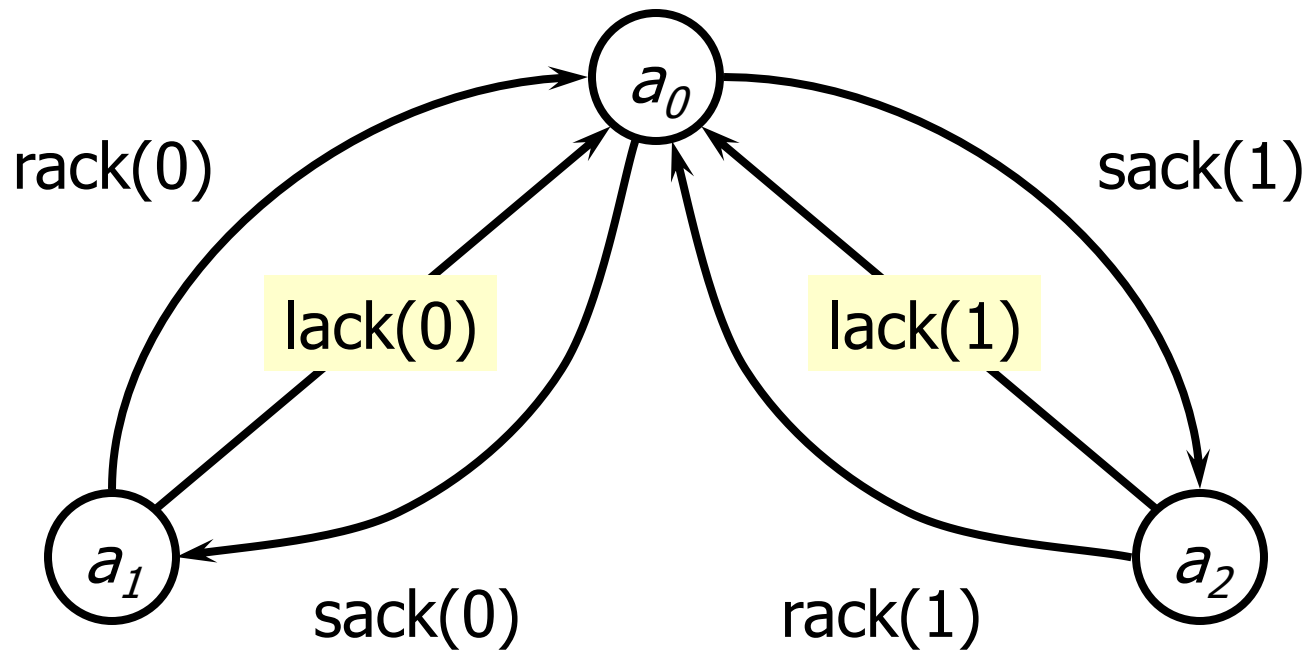




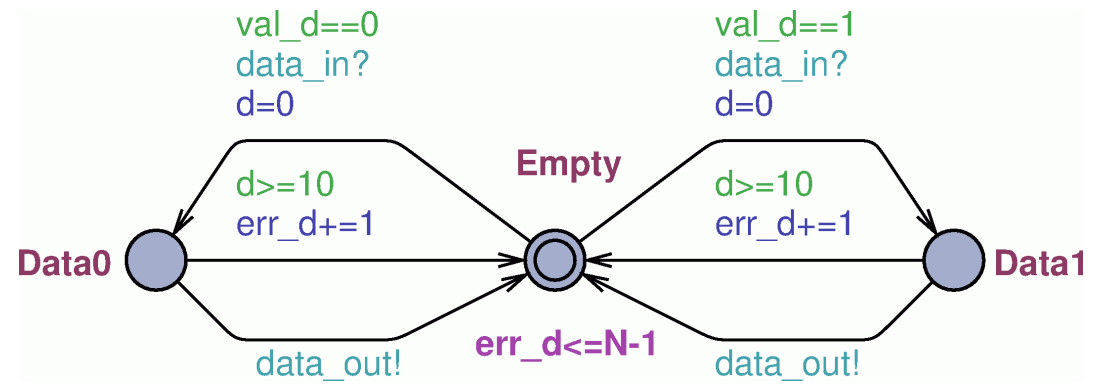
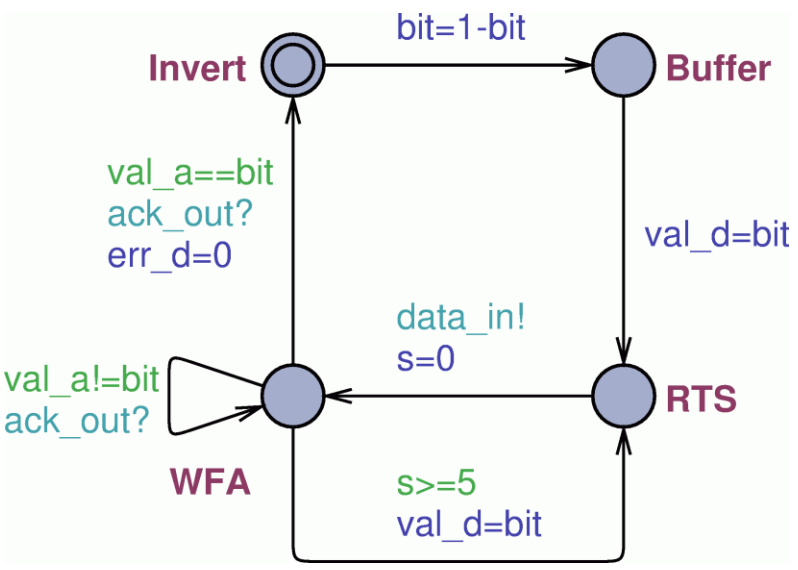
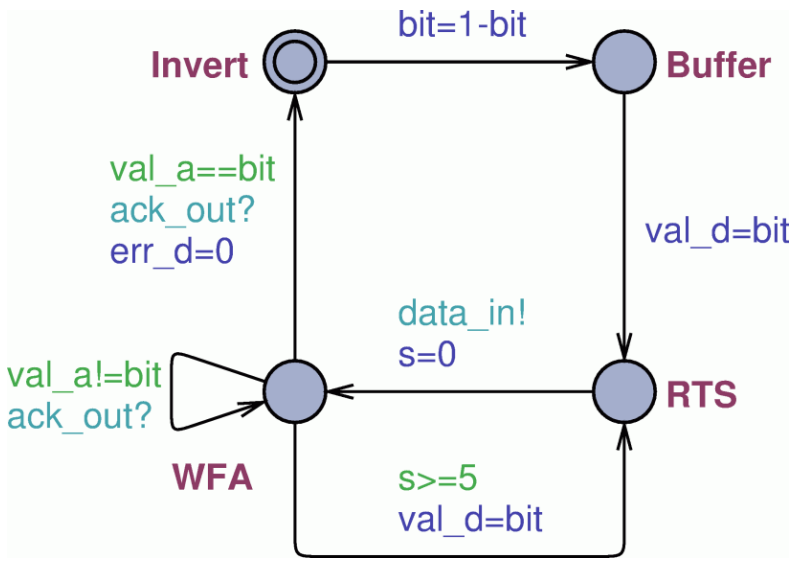
# Adat csatorna állapotgráfja



# Nyugtázó csatorna állapotgráfja



# UPPAAL modell

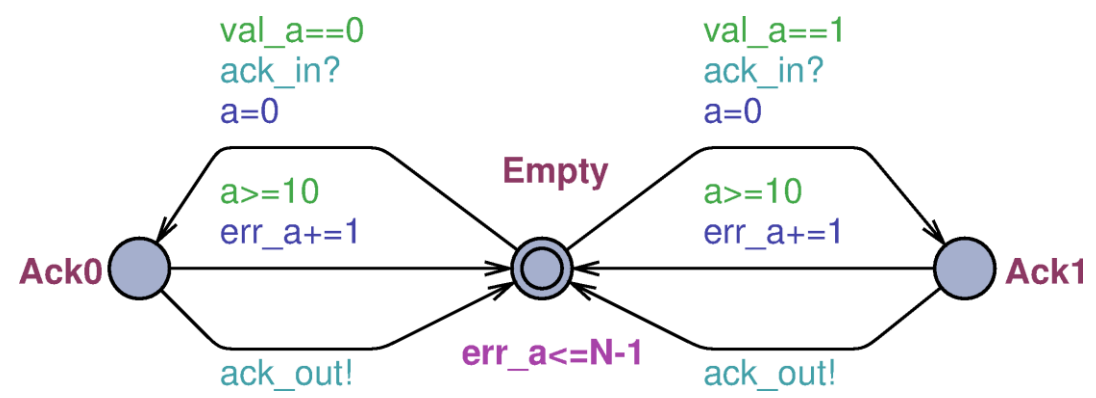


```
const int N=10;
```

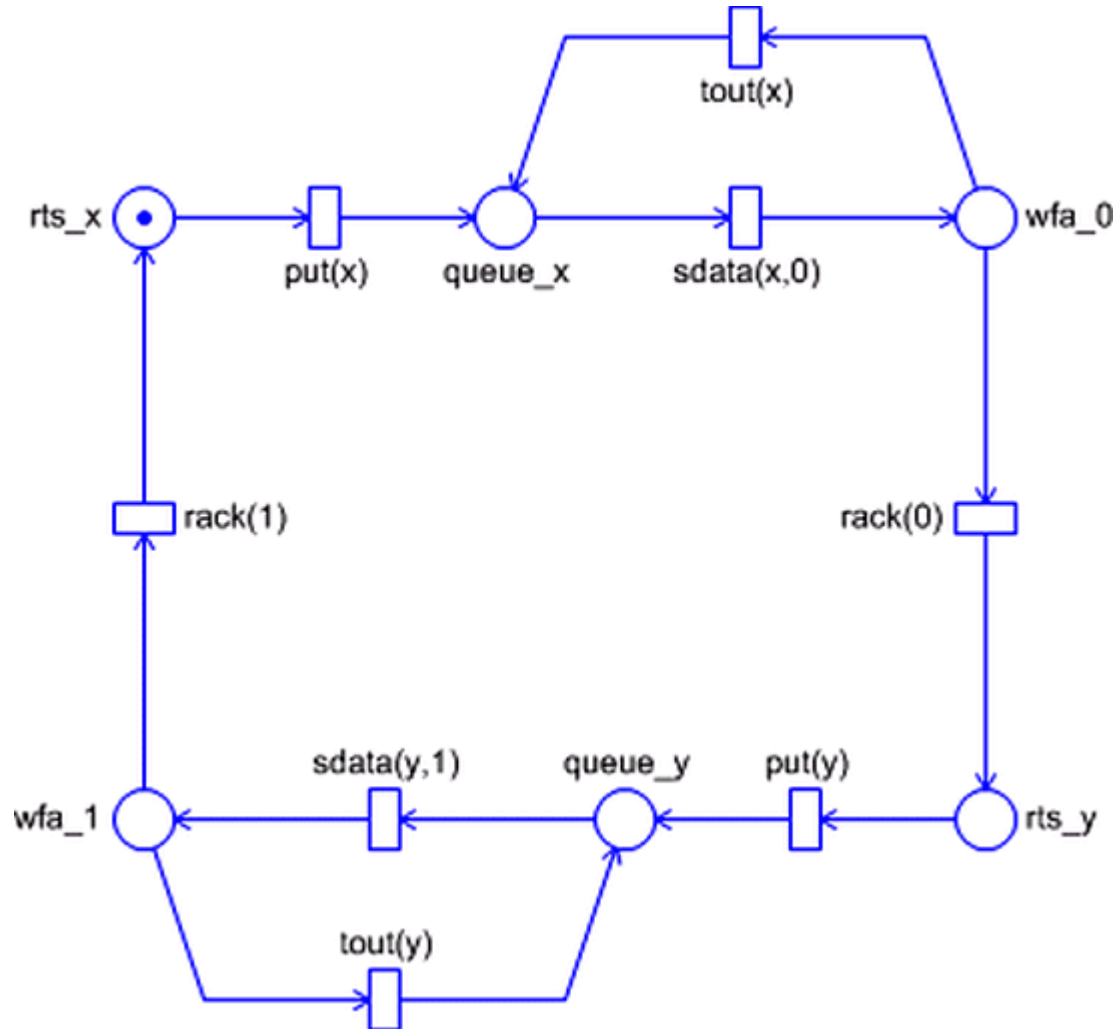
```
value_t val_d, val_a;  
error_t err_d, err_a;
```

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

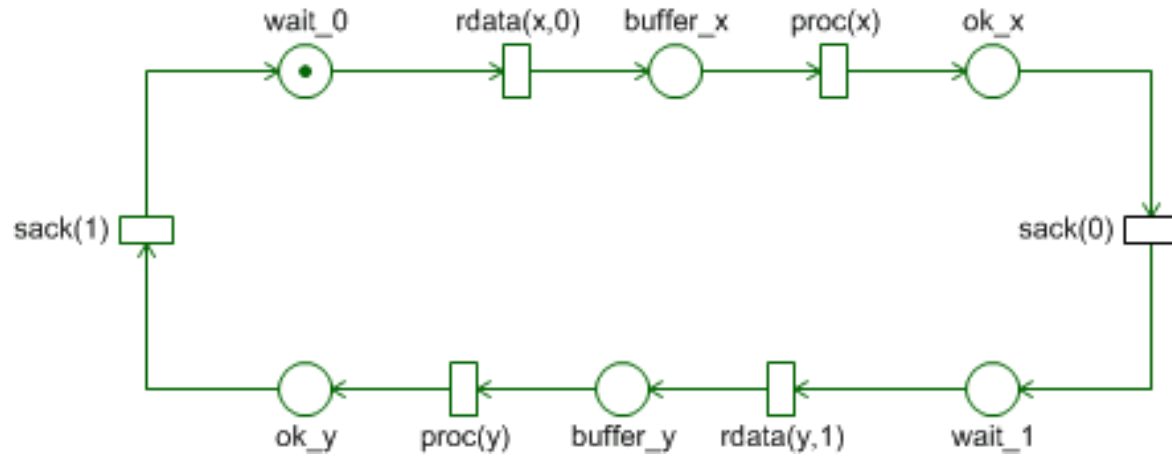
```
chan data_in, ack_in;  
urgent chan data_out, ack_out;
```



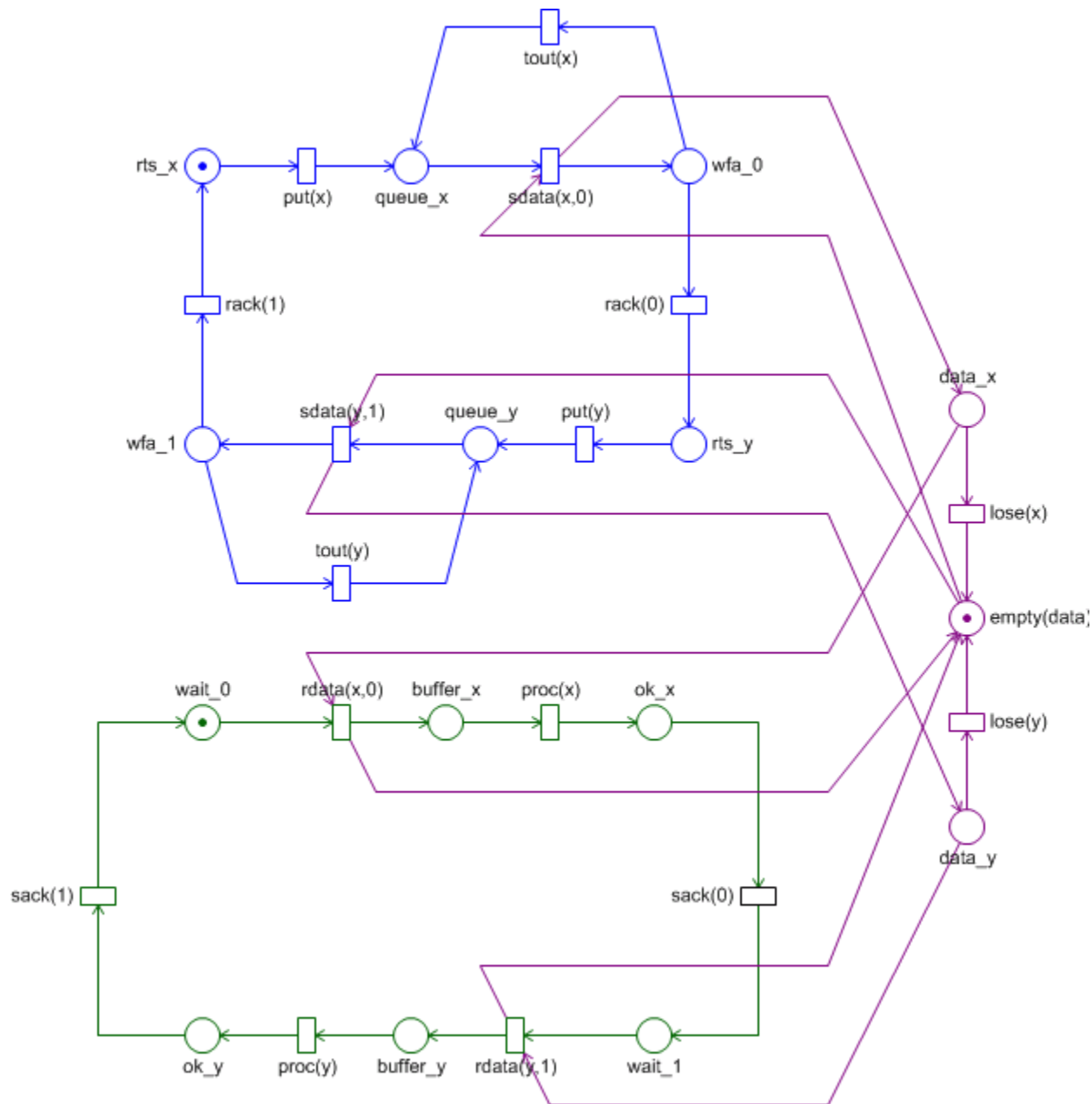
# Küldő folyamat Petri-háló modellje



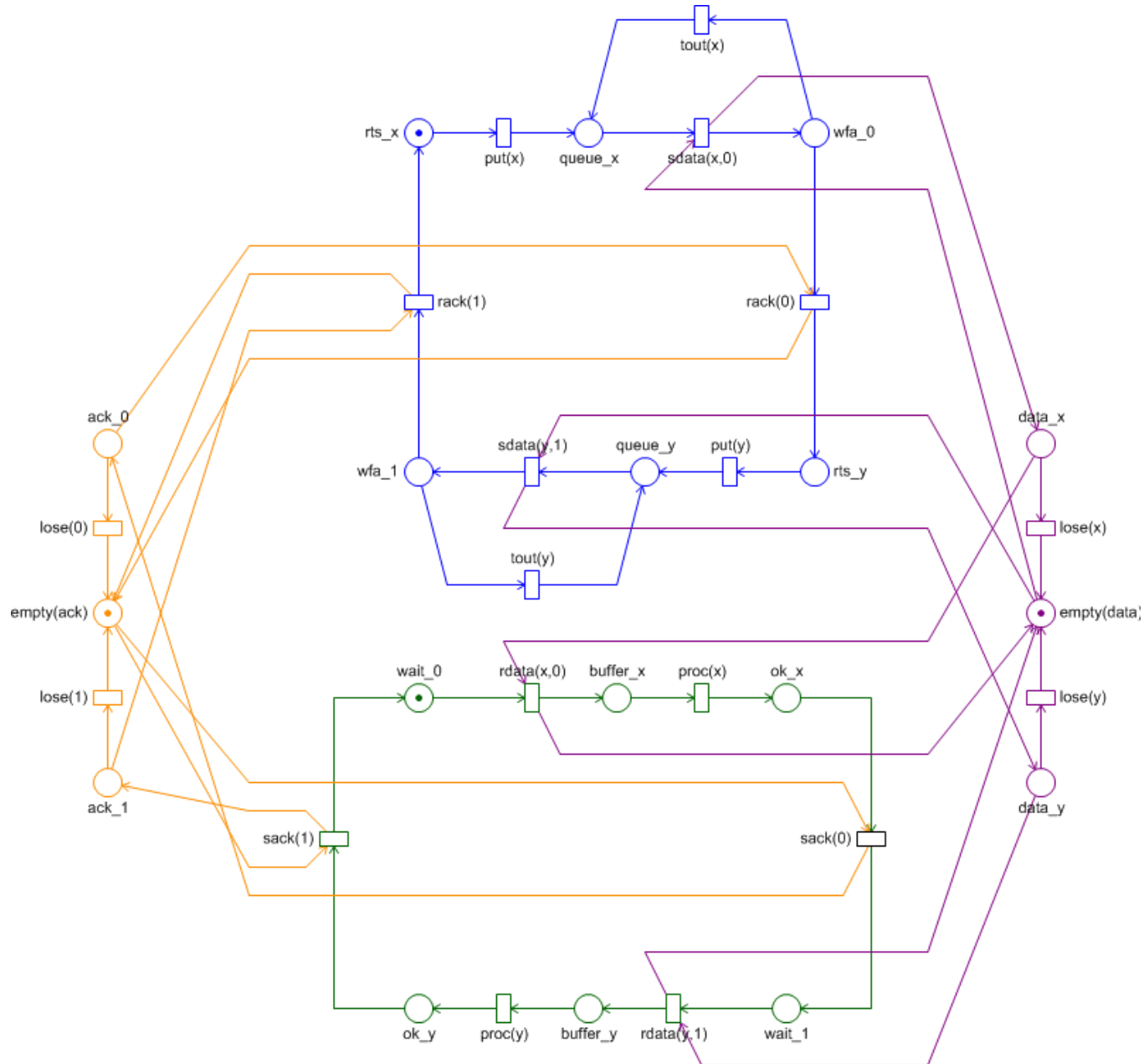
# Fogadó folyamat Petri-háló modellje



# Adat csatorna és az átvitel

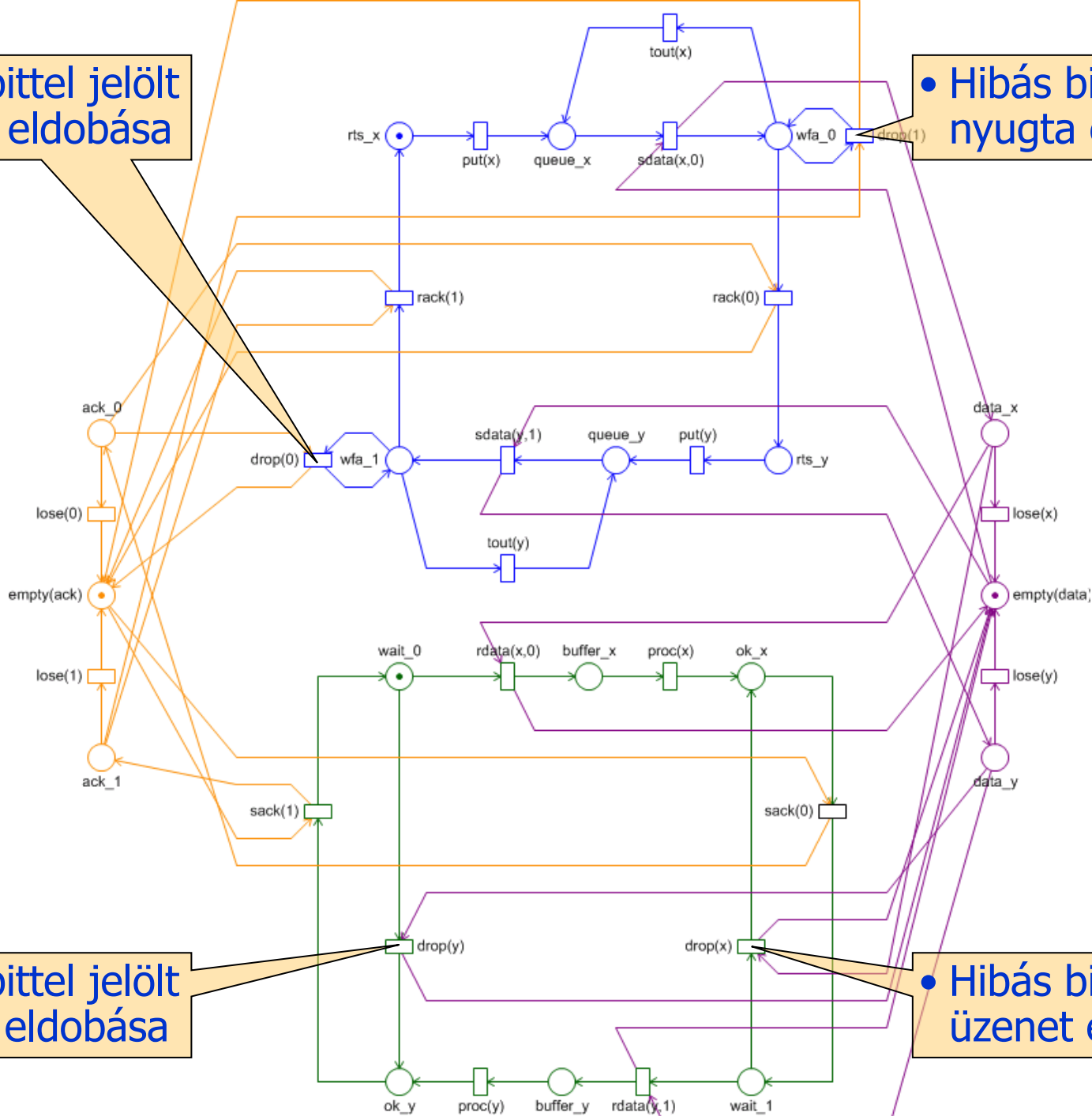


# Nyugtázó csatorna



• Hibás bittel jelölt nyugta eldobása

• Hibás bittel jelölt nyugta eldobása



• Hibás bittel jelölt üzenet eldobása

• Hibás bittel jelölt üzenet eldobása



# Modellező eszköz: PetriDotNet

petridotnet  
from BME-MIT



# 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 PetriDotNet 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), jatekban (bottom), indul (right).
- Transitions:** uzemben (center), veszit (bottom center).
- Edges:** zseton to nyer (weight 2), zseton to indul, nyer to uzemben, uzemben to veszit, veszit to jatekban, jatekban to indul, indul to zseton.
- Initial State:** 2 tokens in zseton, 1 token in 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 **Toolbox** (Select, Place, Transition, Edge, Token, Other elements) and a **Properties** panel. The Properties panel shows the **Identity** section with the following data:

| Identity |              |
|----------|--------------|
| Id       | n1           |
| Name     | <b>Net 1</b> |

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

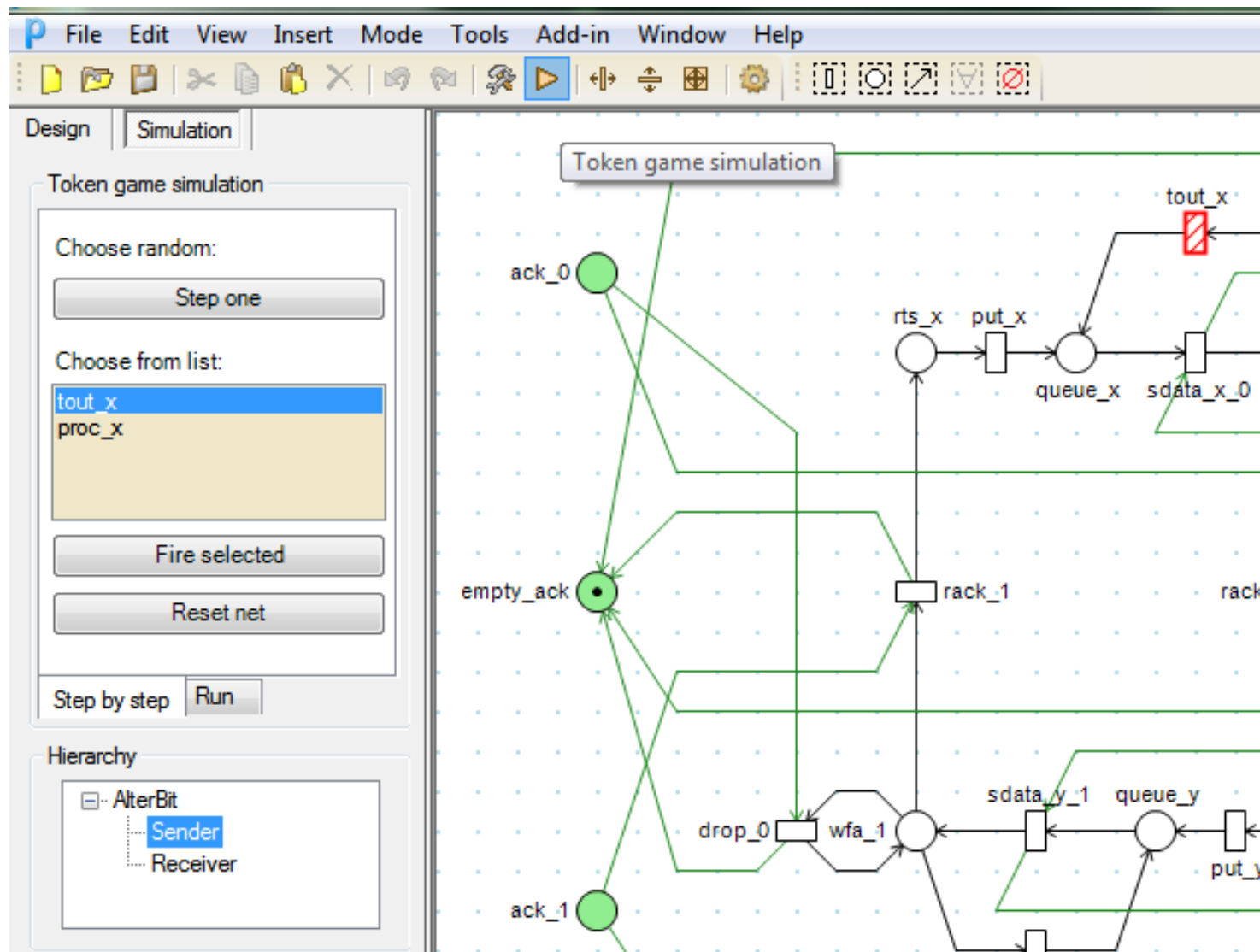
Advisors:  
András Vörös (BME-MIT)  
dr. Tamás Bartha (BME-MIT)

Contact us at <http://petridotnet.inf.mit.bme.hu/> or [petridotnet@inf.mit.bme.hu](mailto:petridotnet@inf.mit.bme.hu).

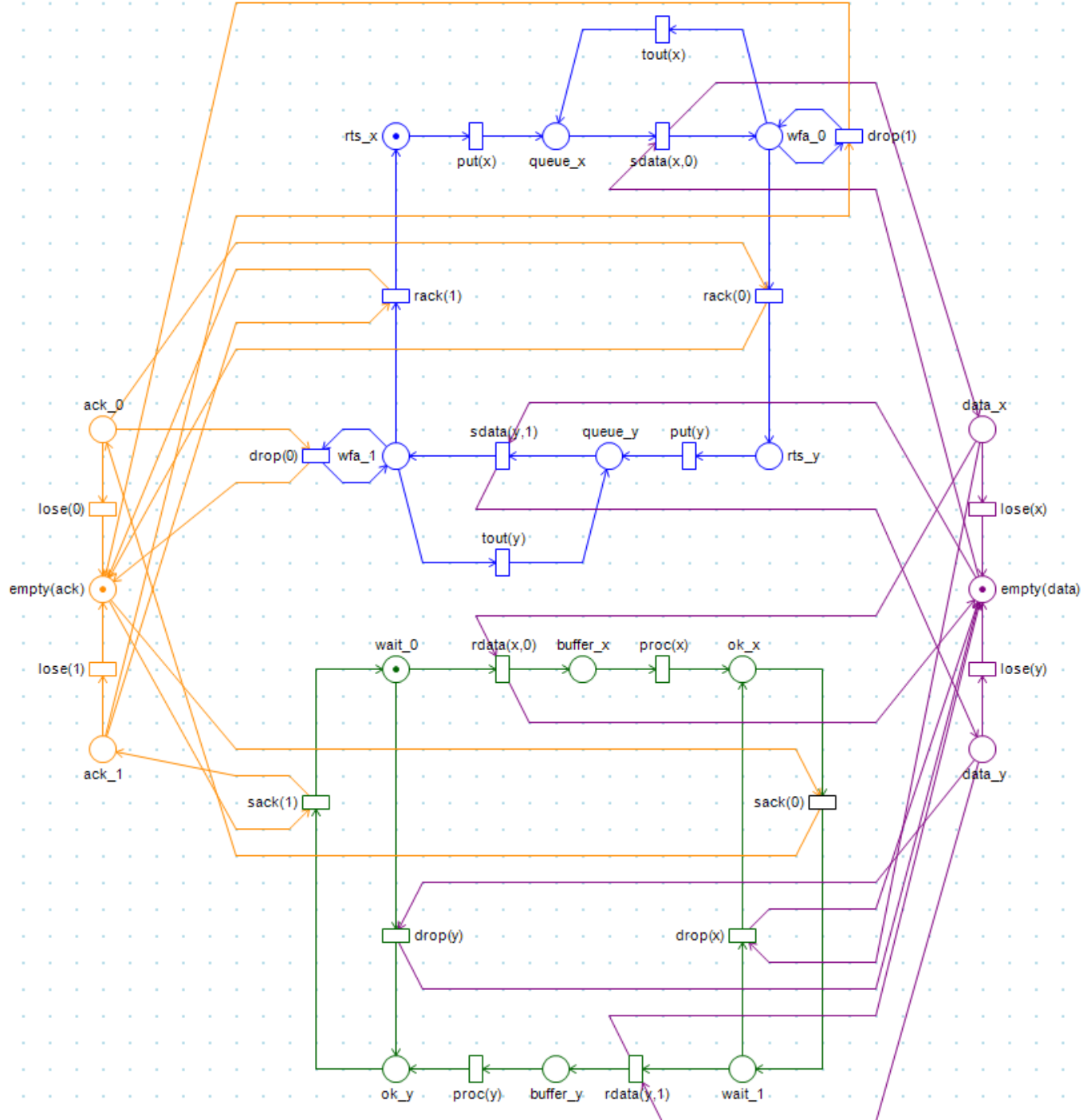
Buttons: Send error feedback, OK

Logo at the bottom: MŰEGYETEM 1782

# Animációs képernyő



# PetriDotNet: A teljes modell



# PetriDotNet analízis eredmények

The image displays the PetriDotNet software interface. On the left, a Petri net diagram is shown with various places and transitions. Places include `ack_0`, `lose(0)`, `empty(ack)`, `ack_1`, `sack(1)`, `wait_0`, `rddata(x,0)`, `buffer_x`, `proc(x)`, `ok_x`, `sack(0)`, `ack_0`, `lose(0)`, `empty(ack)`, `ack_1`, `sack(1)`, `drop(0)`, `wfa_1`, `sdata(y,1)`, `queue_y`, `put(y)`, `rts_y`, `tout(y)`, `rack(1)`, `put(x)`, `queue_x`, `sdata(x,0)`, `wfa_0`, `tout(x)`, and `rack(0)`. Transitions are represented by rectangles and places by circles. Colored lines connect the analysis results to specific parts of the Petri net.

**CTL MODEL CHECKING**  
Expression:  $AG(EF(AlterBit.ok\_y > 0))$   
Model: AlterBit  
Result: True  
Runtime: 0,02 s

**Háló tulajdonságai**

### A(z) AlterBit háló tulajdonságai

**Dinamikus tulajdonságok**

|                    |   |
|--------------------|---|
| Állapotok száma:   | 108   |
| Korlátosság:       | <b>korlátos</b><br>1-korlátos (biztos háló) |
| Holtpontmentesség: | <b>holtpontmentes</b>                       |
| Megfordíthatóság:  | <b>megfordítható</b>                        |
| Persisztencia:     | <b>nem perzisztens</b>                      |
| Korlátos fairség:  | <b>a háló korlátozottan fair (B-fair)</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: Invariáns analízis

The screenshot displays the PetriDotNet application interface. On the left, the 'Háló tulajdonságai' (Network Properties) window contains two 'ShowInvariants' sub-windows. The first shows the expression `{lose(x), sdata(x,0), tout(x)}` and the second shows `{ack_0, ack_1, empty(ack)}`. Below these is a 'P-Invariants' window showing a list of 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}`. On the right, the 'T-Invariants' window displays a list of 22 T-invariants calculated by the Martinez-Silva algorithm, with a calculation time of 15.60 ms. The invariants are listed in a scrollable area, starting with `{lose(x), sdata(x,0), tout(x)}` and ending with `{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))}`. The interface also includes a sidebar with 'Struktúra' (Structure) and 'Legszűkebb' (Narrowest) sections, and a bottom section with links for 'Elérhetőségi grafikon mentése', 'Szomszédsági mátrix mentése', 'T-invariánsok keresése', 'P-invariánsok keresése', and 'Helyek tokenkorlátjainak kiírása'.

**Háló tulajdonságai**

ShowInvariants  
`{lose(x), sdata(x,0), tout(x)}` Show

ShowInvariants  
`{ack_0, ack_1, empty(ack)}` Show

**P-Invariants**  
List of P-Invariants calculated by Martinez-Silva algorithm  
Calculation finished in 0,00 ms. (places=18, transitions=22)  
`{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}`  
OK

**T-Invariants**  
List of T-Invariants calculated by Martinez-Silva algorithm  
Calculation finished in 15,60 ms. (places=18, transitions=22)  
`{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),`  
OK

**Struktúra**  
Legszűkebb  
Tisztaság

[Elérhetőségi grafikon mentése](#), [Szomszédsági mátrix mentése](#),  
[T-invariánsok keresése](#), [P-invariánsok keresése](#),  
[Helyek tokenkorlátjainak kiírása](#)

# PetriDotNet: Elérhetőségi gráf rajzolása (GraphViz)

