Modellezés és szimuláció Petri hálókkal

dr. Bartha Tamás

Dr. Pataricza András

BME Méréstechnika és Információs Rendszerek Tanszék

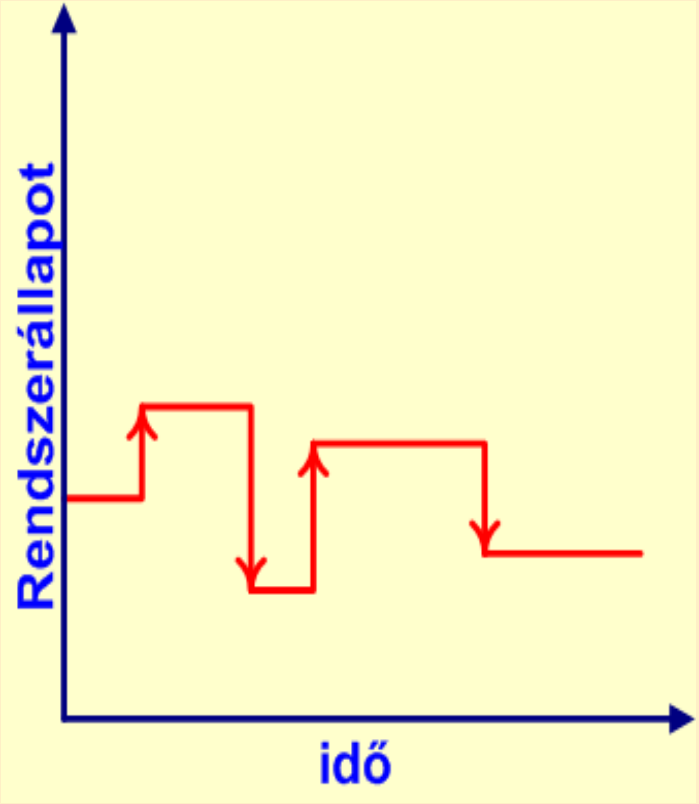
Diszkrét rendszermodellezés alapjai

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

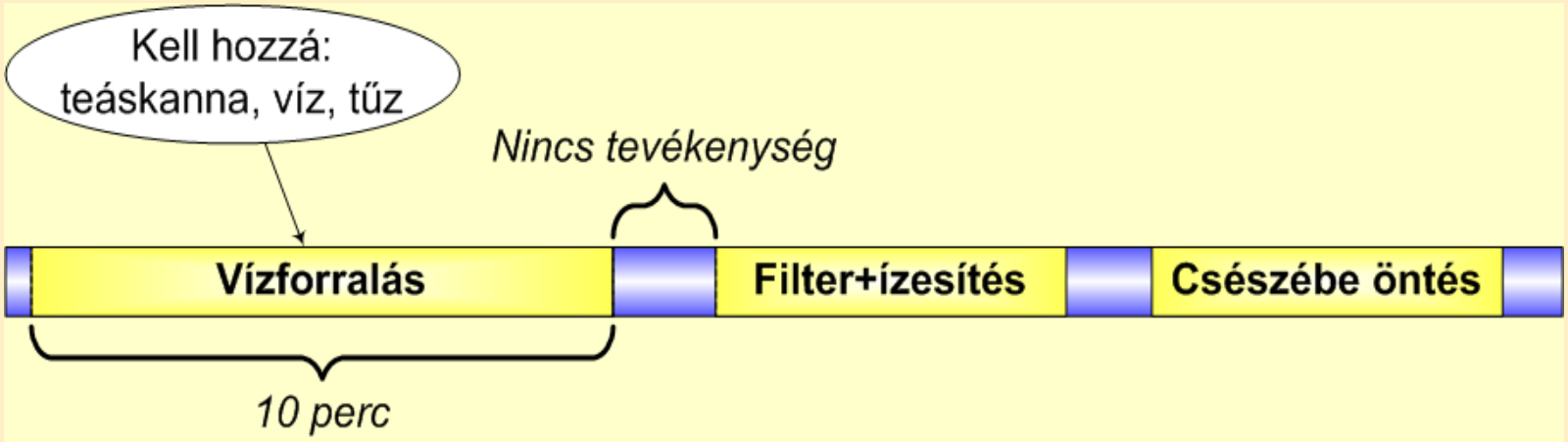
- Modellezés célja
  - Korreláció: modell ↔ „valóság”
  - Érthetőség, áttekinthetőség
    - Nem cél a minimális modell
  - A teljes rendszer dinamikája
    - Komponensek modellje egyszerű
    - Bonyolult kölcsönhatások
- Csoportosítás
  - Folytonos } • értékben
  - Diszkrét } • időben



Folyamatorientált modellezés

- Folyamat: tevékenységek logikailag rendezett sora
- Elemi tevékenységekből áll
- Jellemzői:
  - Erőforrást igényel
  - Mennyi ideig tart

+ járulékos információk:
- erőforrások
- kölcsönhatások, pl.
- erőforrás használat



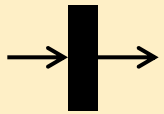
Tevékenységek

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

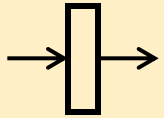
Tevékenységek modellezése Petri hálókban

Petri háló alapú modellek esetén a tevékenység:

- elemi lépés → tranzíció tüzelése
- felhasznált erőforrások → bemeneti / kimeneti helyek
- végrehajtási idő
  - determinisztikus
  - sztochasztikus



determinisztikus időzítésű tranzíció



exponenciális időzítésű tranzíció

Az engedélyezettséggel kapcsolatban felmerülő kérdések:

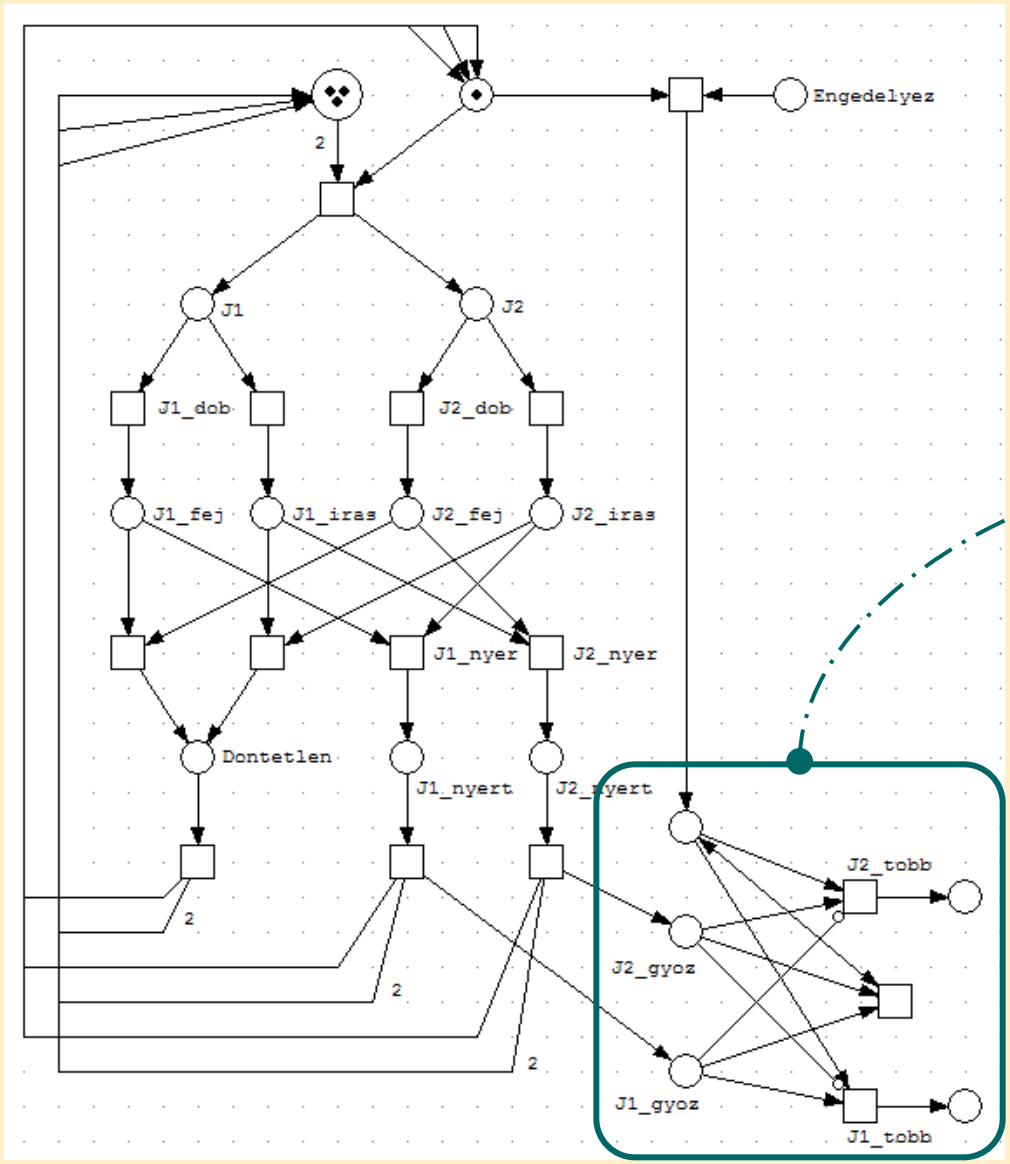
- Időzítetlen tranzíciók „prioritása” nagyobb: előbb tüzelnek
- Engedélyezettség megszűnése: mi lesz a gyújtási idővel?
  - újrainduló (újra sorsol)
  - megőrződő (korábban sorsoltból fennmaradó idő folytatódik)

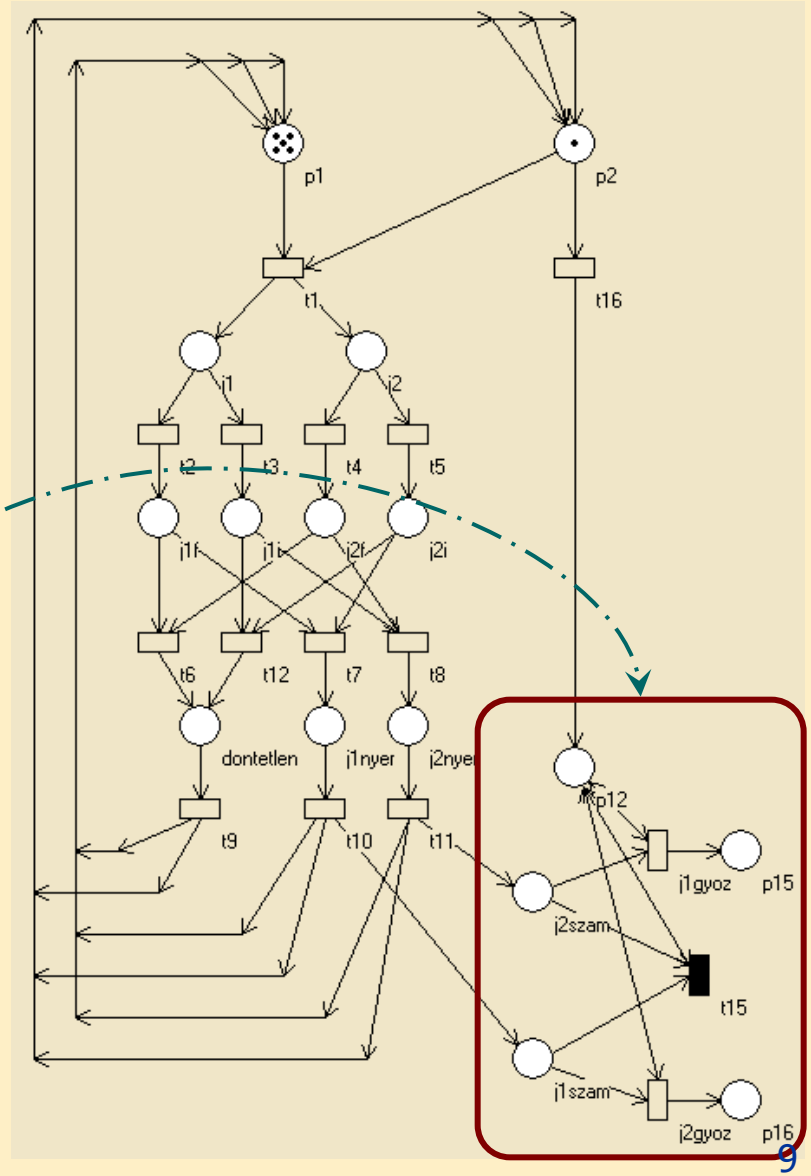
Sztochasztikus Petri hálók

SPN = (P, T, E, W, M_0, Λ)

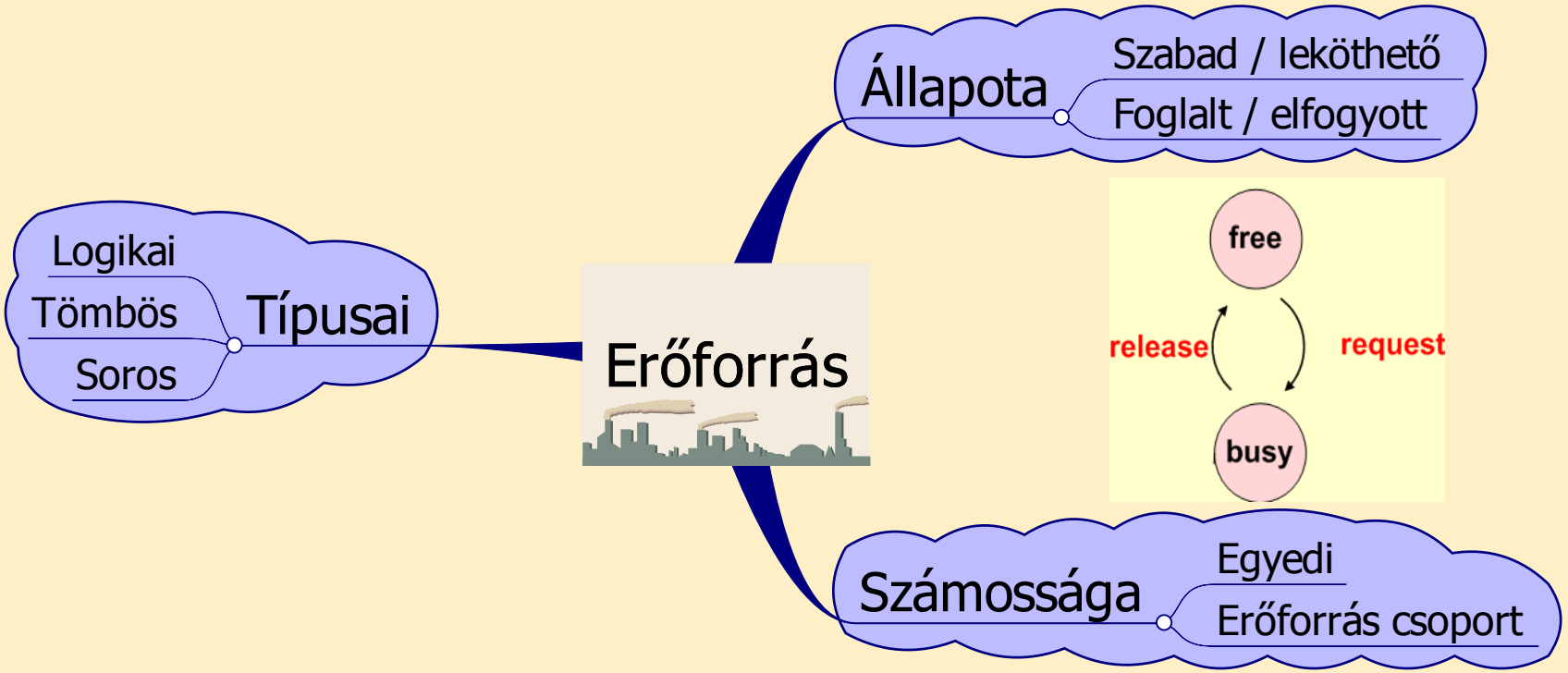
- tüzelési intenzitás $\Lambda : T \rightarrow R$
- időzítés: τ valószínűségi változó
  - eloszlásfüggvény: $P\{\tau < t\} = F(t) = 1 - e^{-\lambda t}$
  - sűrűségfüggvény: $f(t) = \lambda e^{-\lambda t}$
- tüzelési szabály változik: intenzitás ~ prioritás
- két exponenciális eloszlás minimuma is az: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

Időzítetlen tranzíciók prioritásának felhasználása





Erőforrások



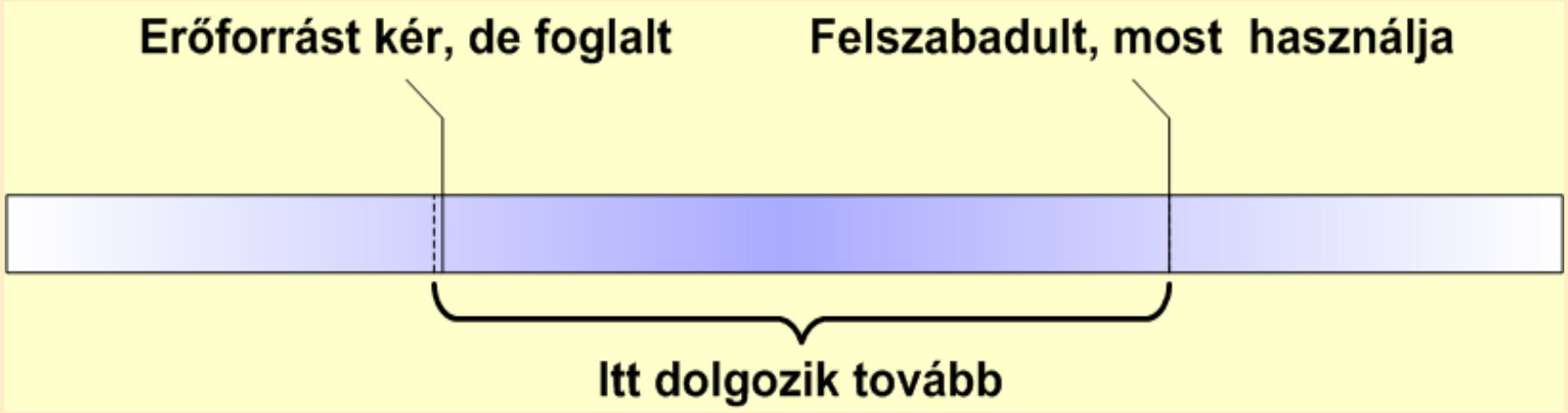
Erőforrás csoport

- funkcionálisan ekvivalensek
- szabad vagy foglalt a többitől függetlenül
- adott mennyiség lefoglalható

Blokkoló / nem blokkoló erőforrás

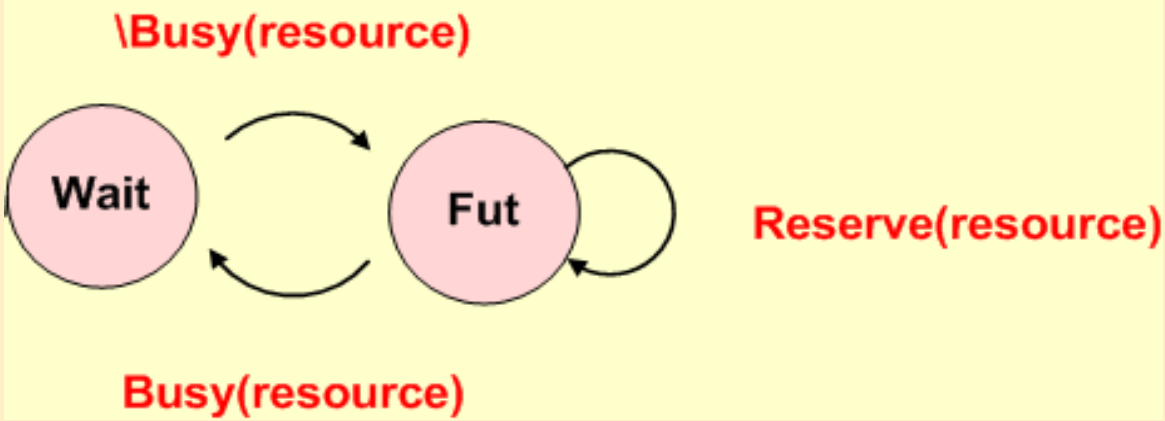
- Blokkoló

- Nem blokkoló



Események

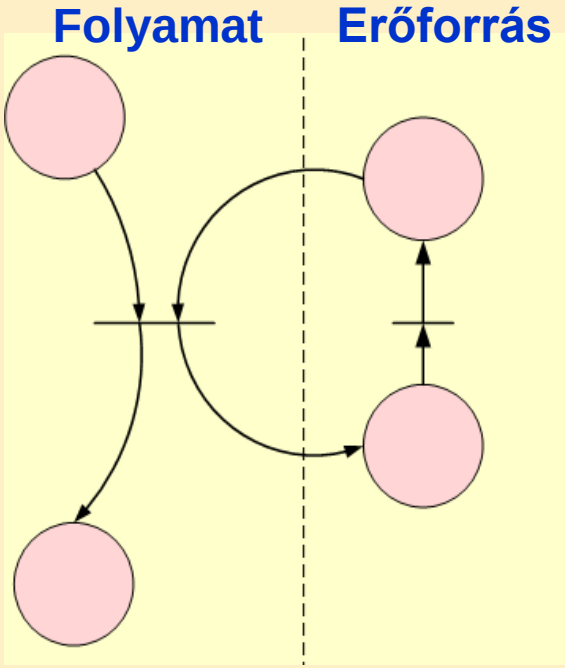
- Rendszerszintű modellezésben eseménynek tekinthető:
  - Valamely tevékenység kezdete, illetve vége
  - Az erőforrások állapotváltozása
- Tevékenység mit tud csinálni?
  - Erőforrást foglal: Reserve (resource_list)
  - Erőforrásra vár: Wait (usage_time)
  - Erőforrást elenged: Release (other_resources_list)



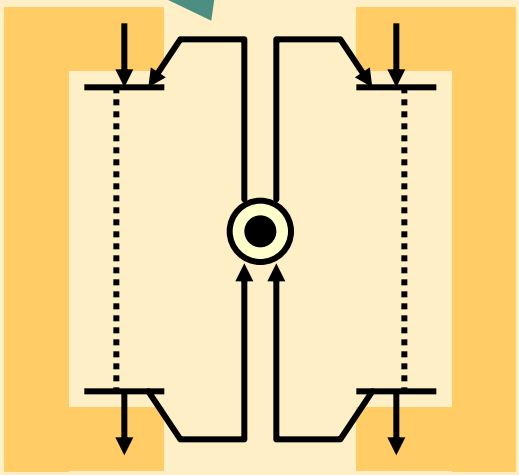
Erőforrás allokáció Petri hálókban

- Kölcsönös kizárás
- Több darab lefoglalása

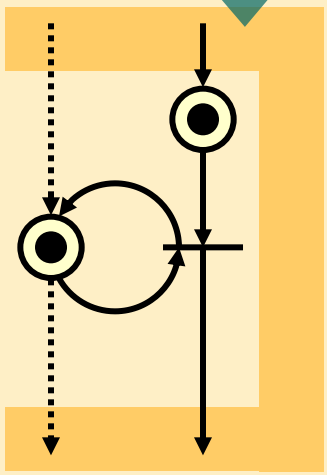
Cél: nem a minimális,
hanem az érthető Petri háló!



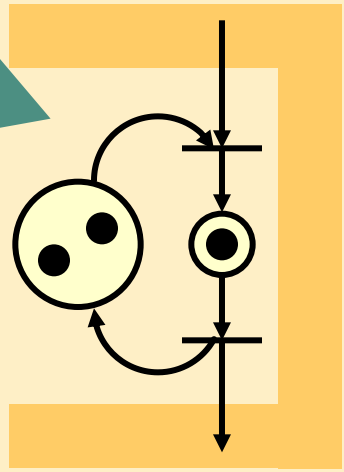
Kölcsönös kizárás megvalósítása



Állapotváltozó leolvasása

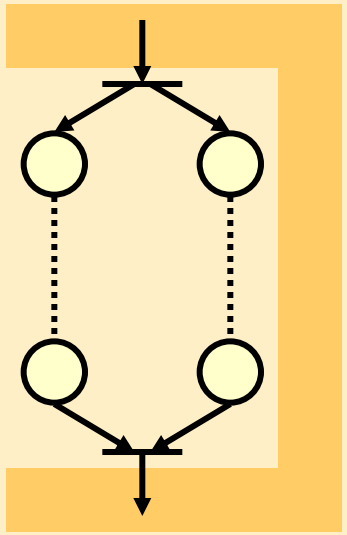


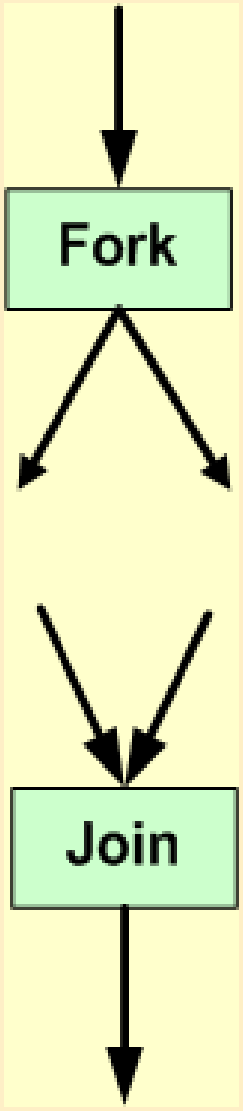
Korlátos erőforrás kapacitás modellezése

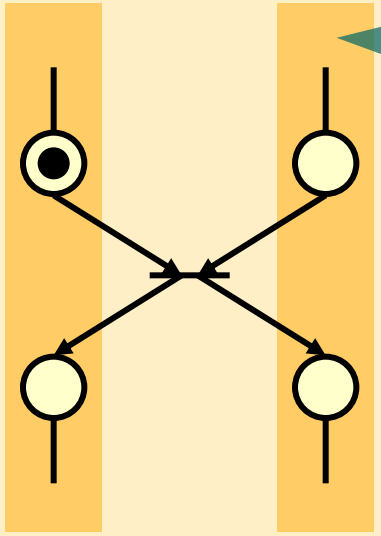


Üzenetek

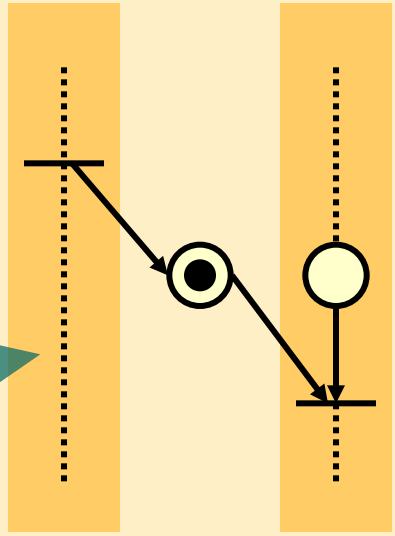
- Szoftverben párhuzamosság
  - FORK - elágazás
  - JOIN - visszatérés
- Kommunikáció biztosítása
  - üzenetekkel
- Wait – egymásra várás







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

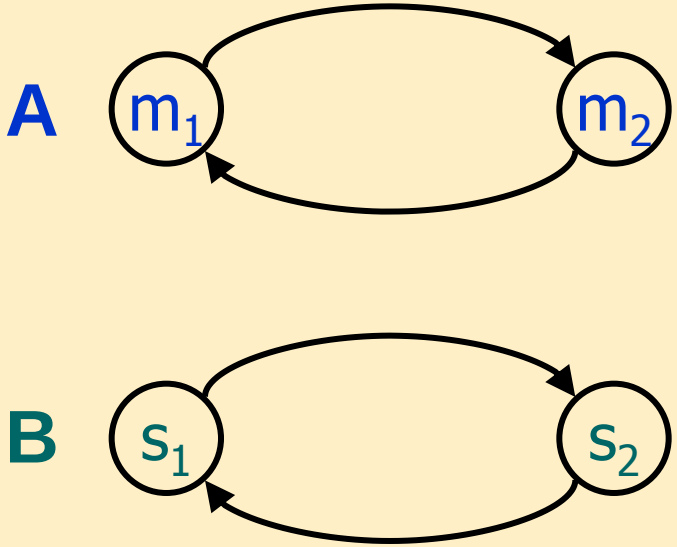
Rendszermodellezés

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

Automaták összevonása, direkt szorzat

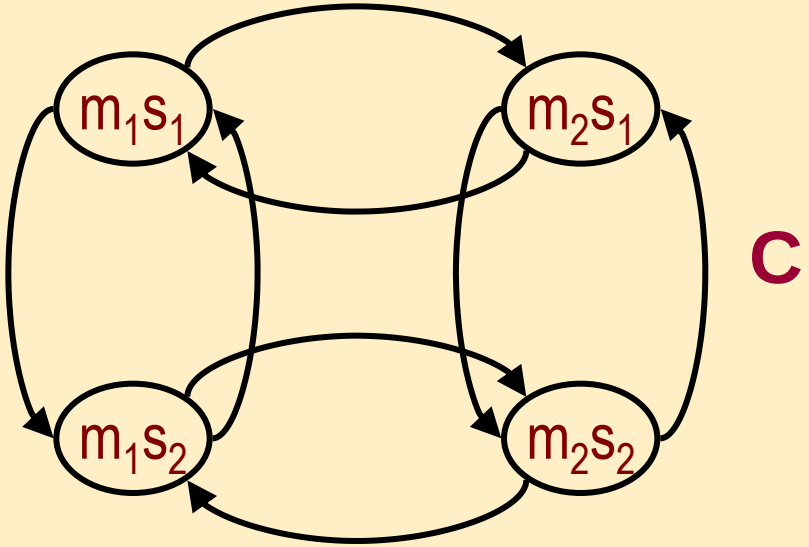
- Két (független) automatából álló rendszer



- Az automaták állapotai:

  $A = \{m_1, m_2\}$, $B = \{s_1, s_2\}$

- (Direkt) szorzatautomata: a rendszer állapottere



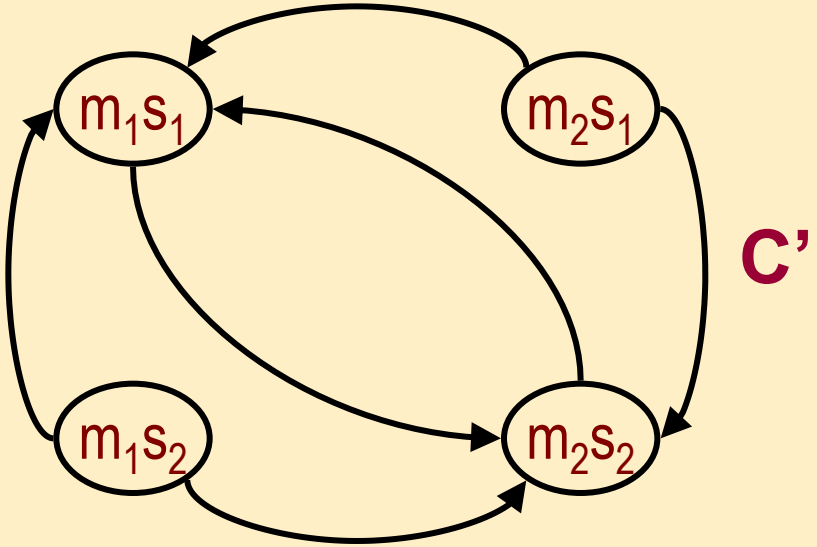
- Az állapotok halmaza:

  $C = A \times B$

  $C = \{m_1s_1, m_1s_2, m_2s_1, m_2s_2\}$

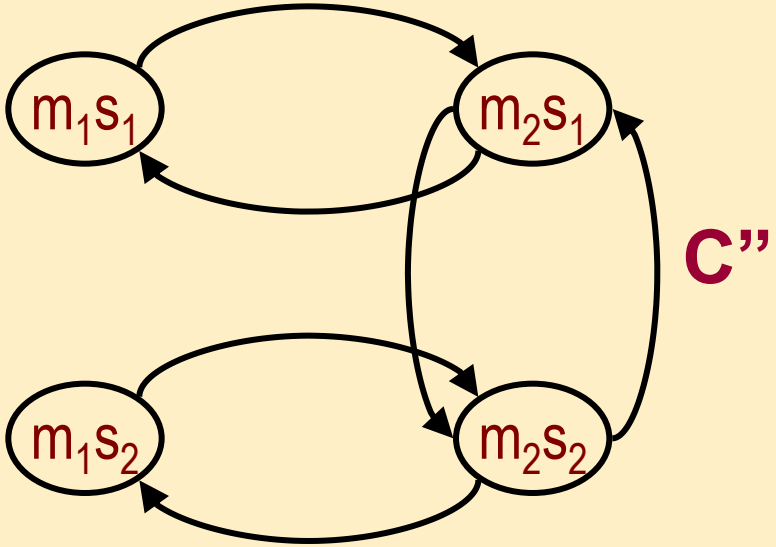
A rendszer finomítása: feltételek, szinkronizáció

- Egyidejű állapotváltás: szinkronizáció



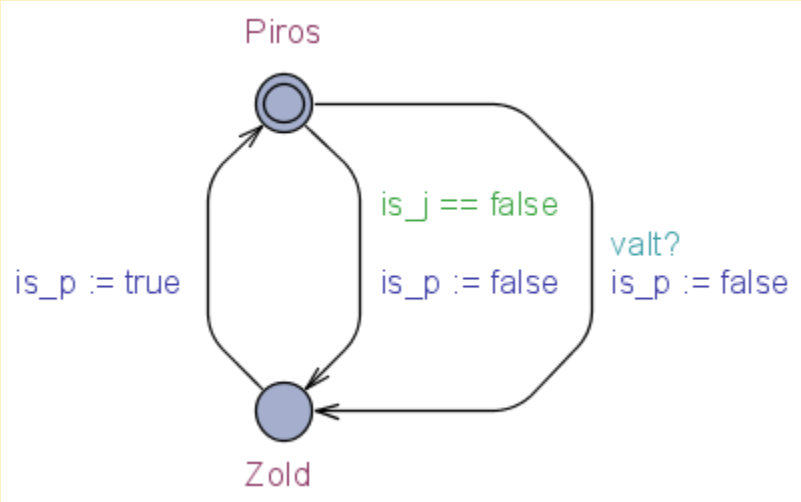
- pl. „A és B egyszerre vált állapotot, ha állapot indexük azonos”

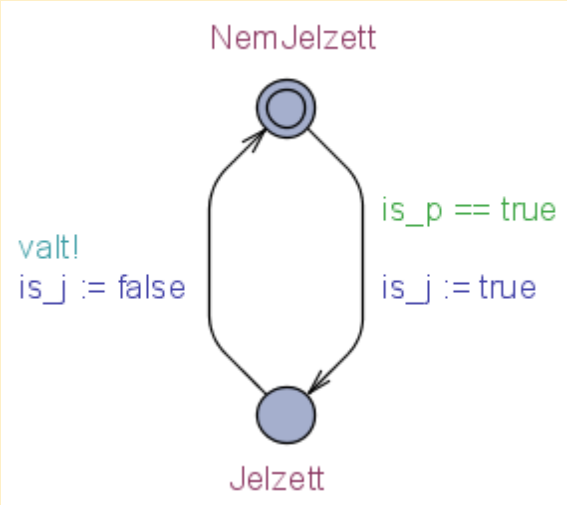
- Korlátozó feltételek: állapotátmenetek tiltása

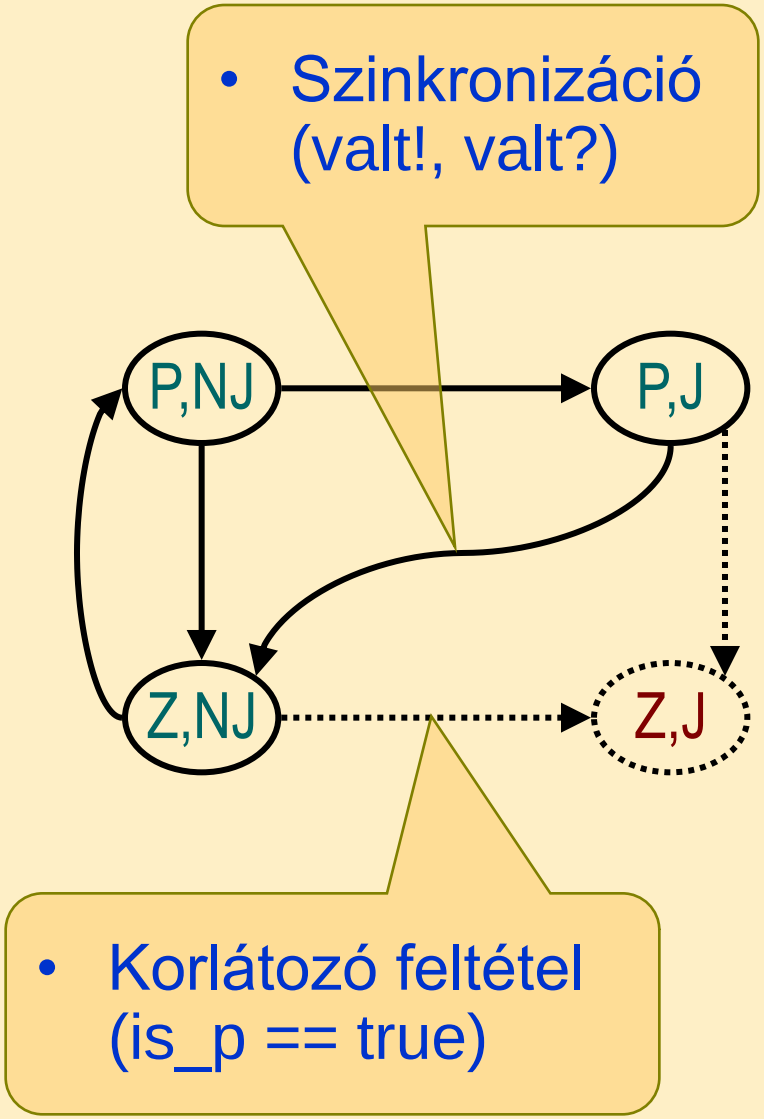


- pl. „B csak akkor vált állapotot, ha A m_2 állapotban van”

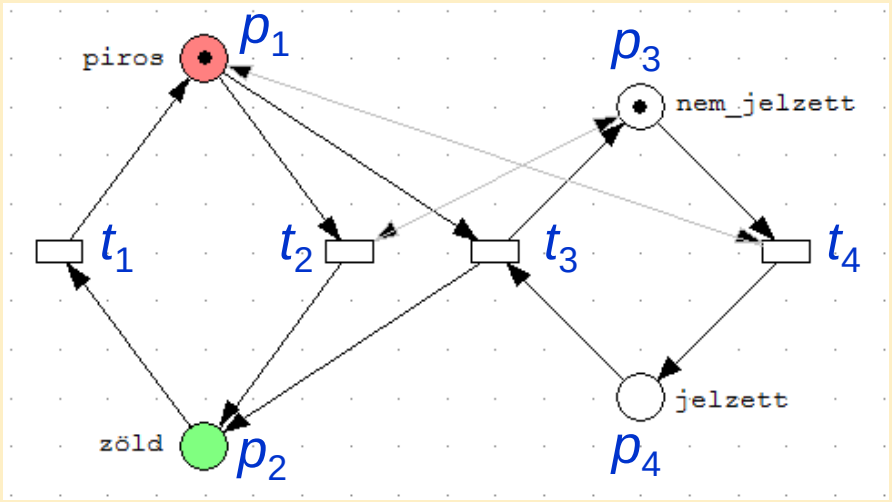
Példa: gyalogos lámpa jelzőgombbal

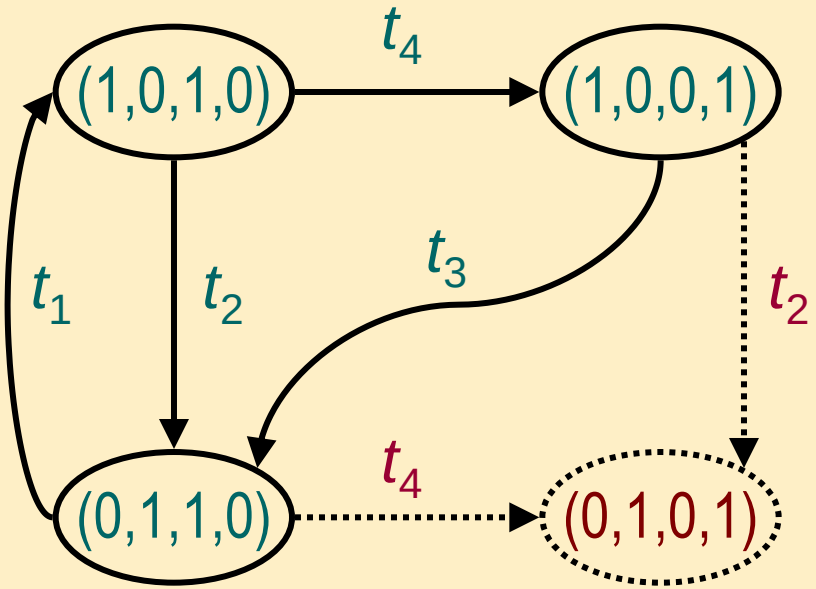






Ugyanez Petri hálóval





- Petri hálók esetén a feltételek és a szinkronizáció beépülnek a háló struktúrájába
  - feltételek az engedélyezettség feltételeiként jelennek meg
  - szinkronizáció egy „összevont” tranzíció formájában
  - nem kell szorzatautomatát építeni, csak elérhetőségi gráf építés

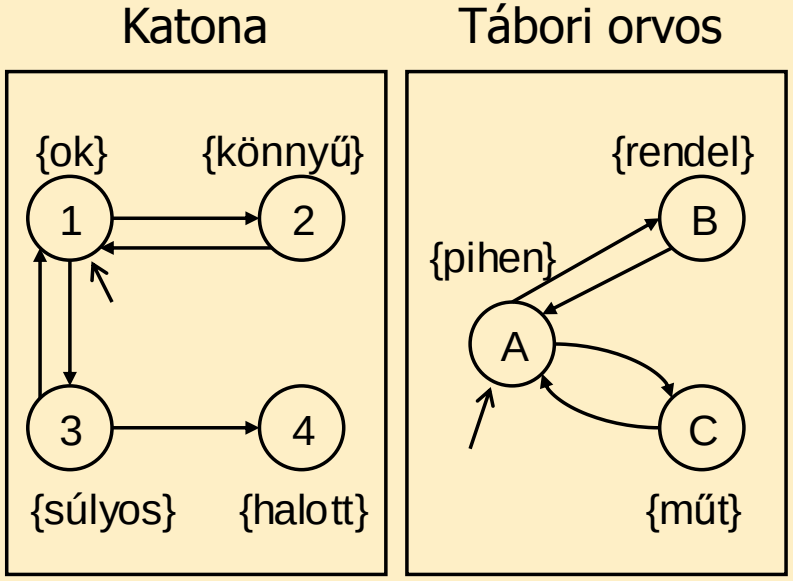
Egy összetettebb szöveges példa

Rendszer (két automata):

1. Katona
2. Tábori orvos

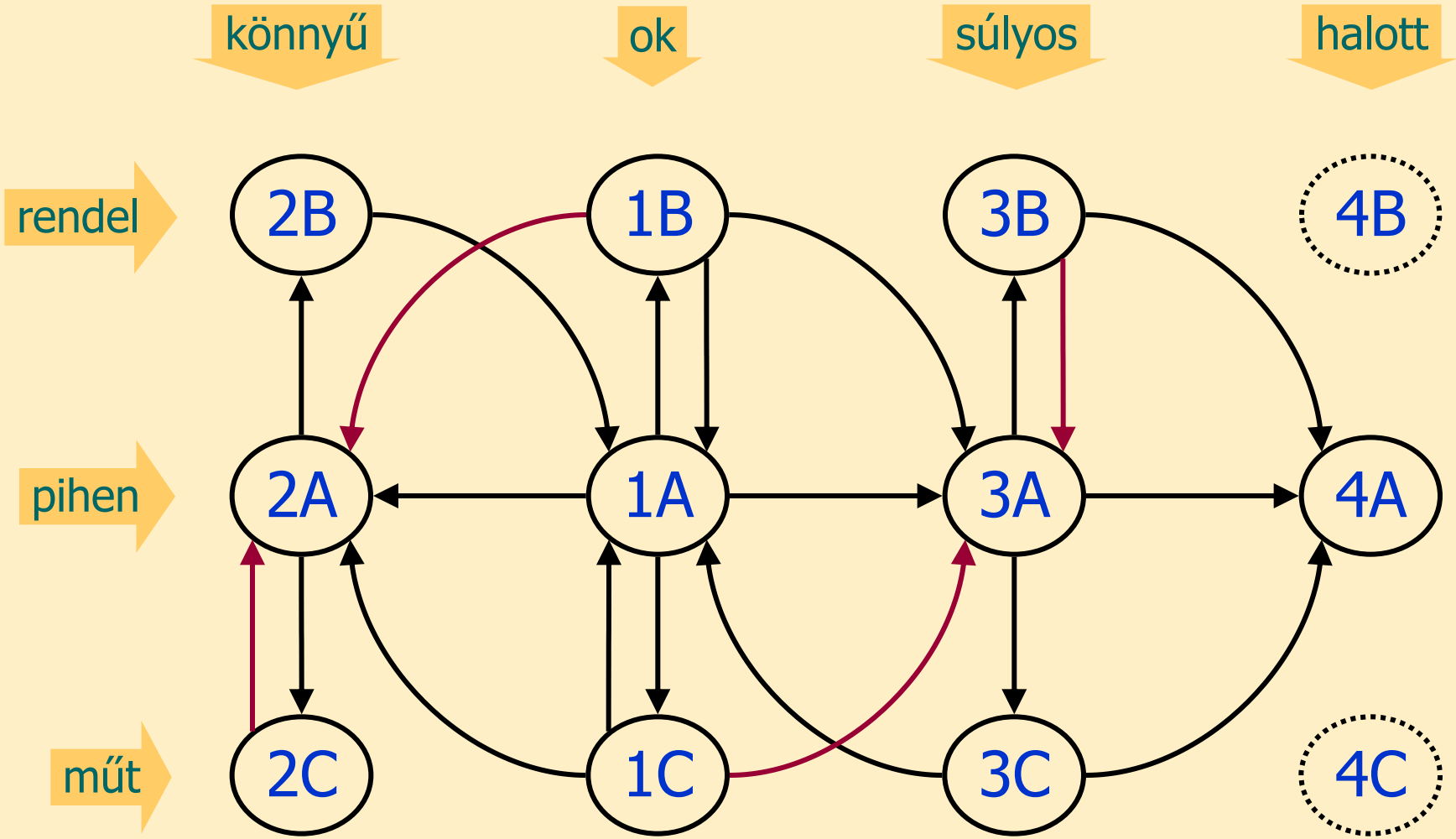
Korlátozások + szinkronizáció:

- Egy könnyebben sérült katona magától nem gyógyul meg, csak ha az orvos éppen rendel
- Ha a katona súlyosan sérült, akkor csak műtéttel lehet rajta segíteni
- Ha egy súlyos sérült katona nem kap időben kezelést, vagy a műtét sikertelen, akkor meghalhat



- Mindez egyszerre történik:
  - A rendelést követően a könnyű sérült biztosan meggyógyul
  - A műtét következtében a beteg vagy meggyógyul, vagy meghal
  - Az orvos a rendelés, illetve műtét után egyből pihenni kezd

A feltételek figyelembe vételével készített szorzat



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: $\mathbf{b^0}$
  - ha az üzenet elvész, a folyamat időtúllepéssel észleli a nyugta hiányát $\rightarrow$ újra küldi
  - ha a folyamat $\mathbf{b^0}$ bittel ellátott nyugtát kap (ilyet várt), akkor a következő üzenethez $\mathbf{b^1} = \neg\, \mathbf{b^0}$ bitet köt
  - ha a folyamat $\mathbf{b^X}$ bittel ellátott nyugtát vár és $\mathbf{b^Y}$ bittel ellátott nyugtát kap $\rightarrow$ egyszerűen eldobja

Fogadó folyamat

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

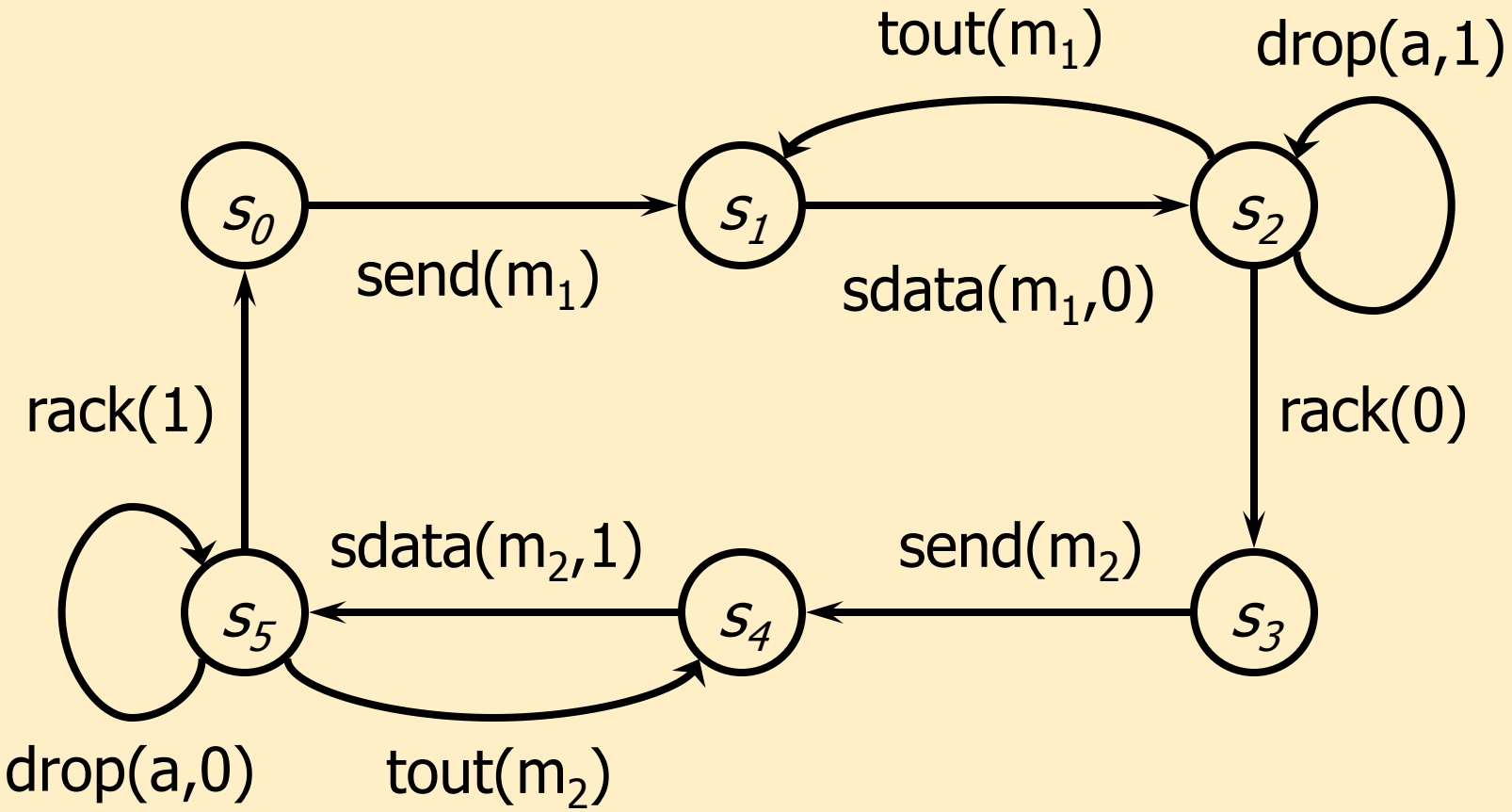
A modellalkotás lépései

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

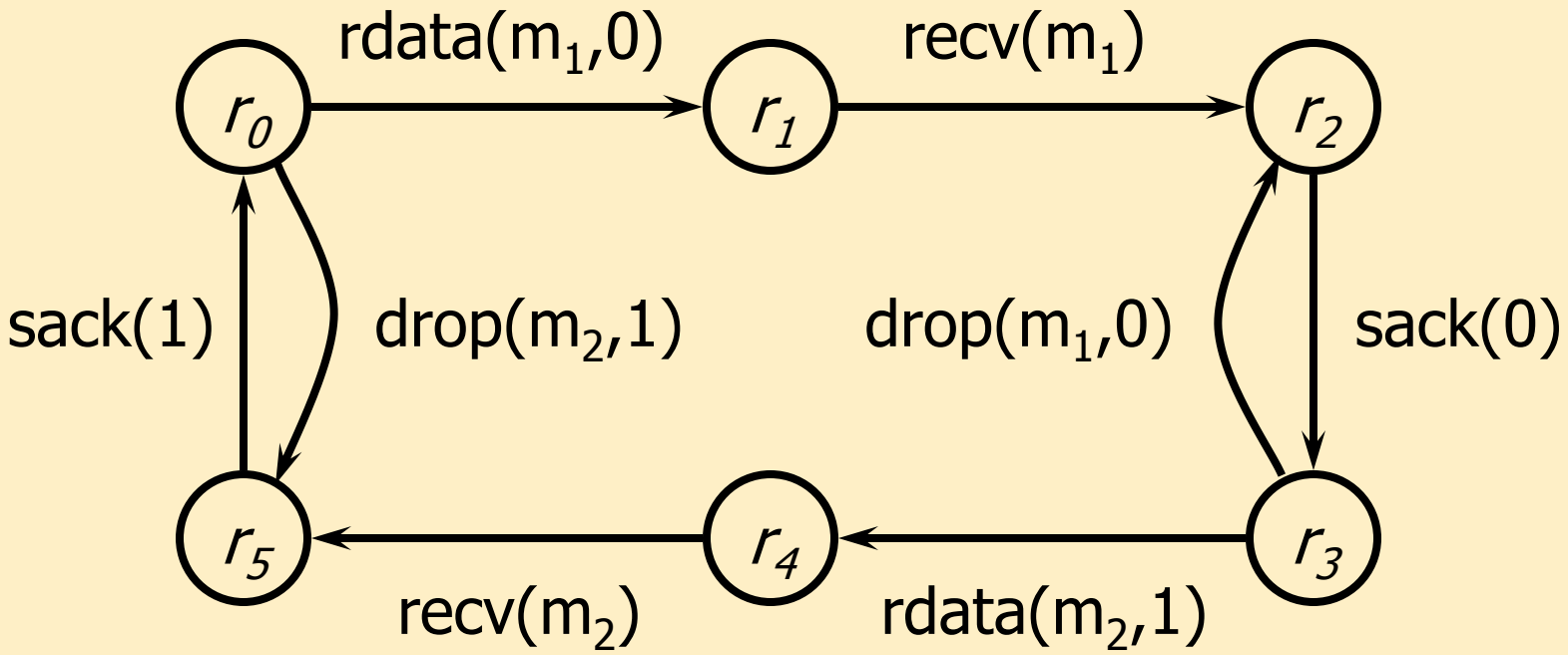
Komponensek és állapotgráfjaik

- 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

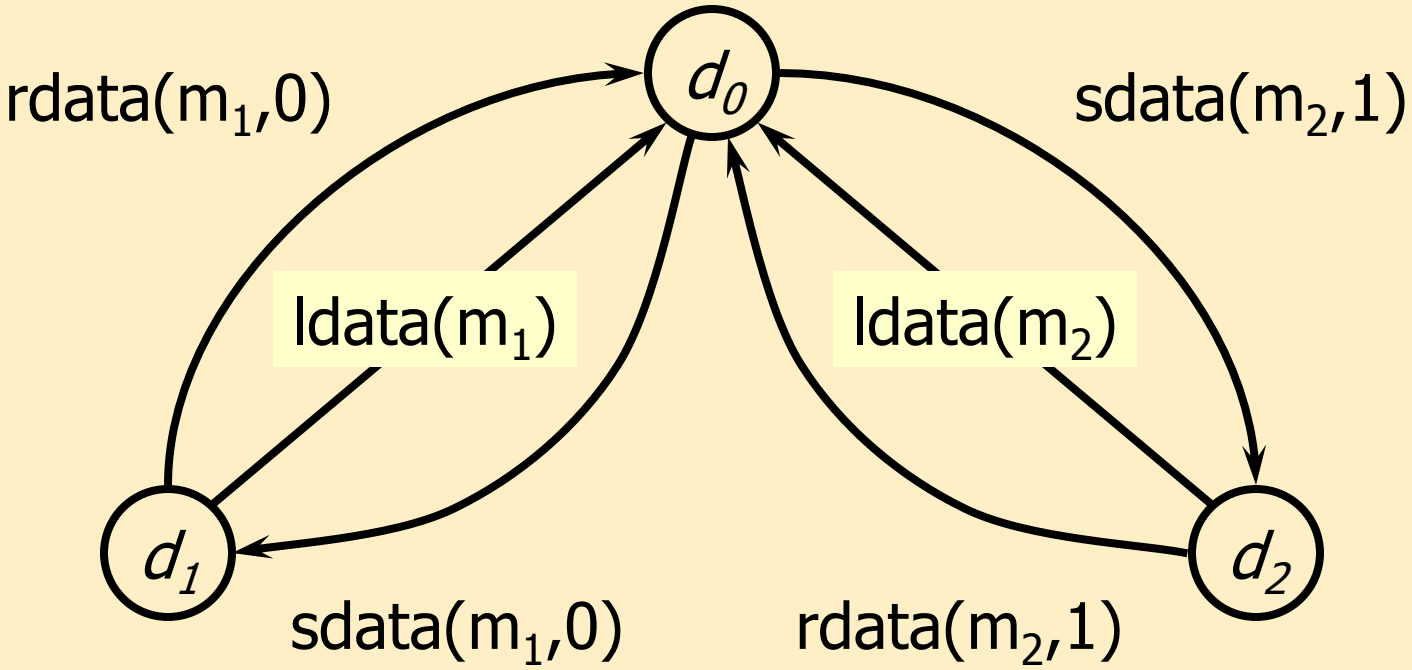
Küldő folyamat állapotgráfja



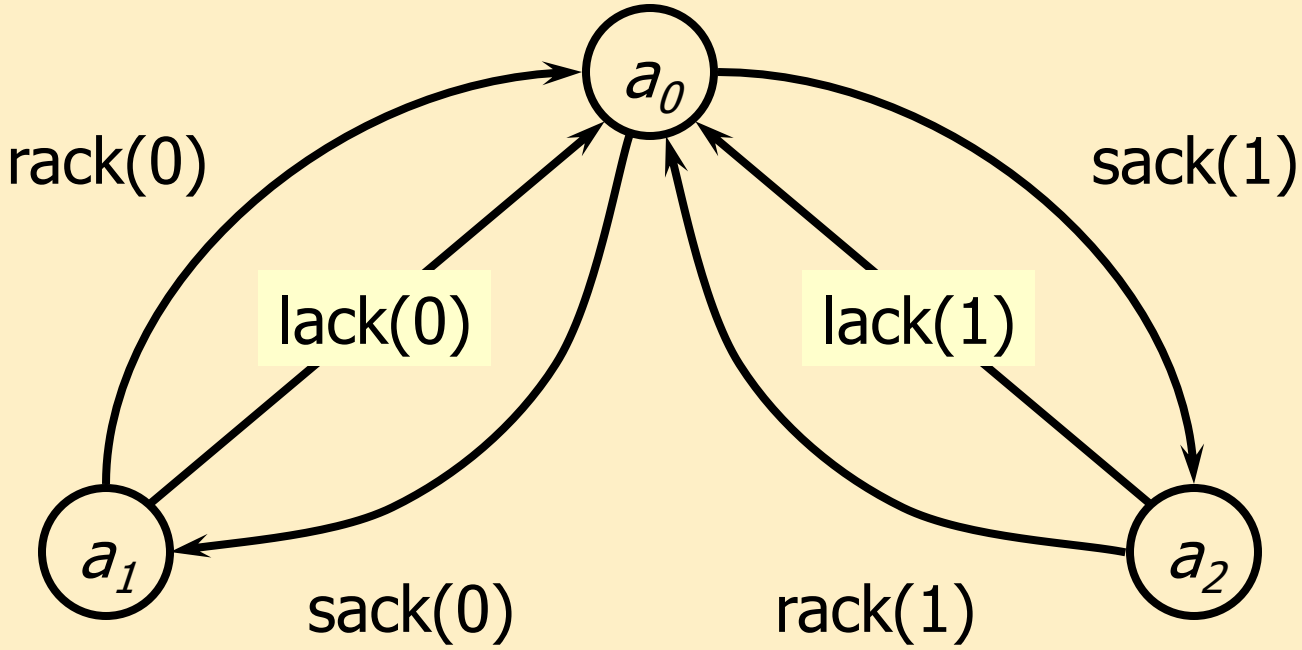
Fogadó folyamat állapotgráfja



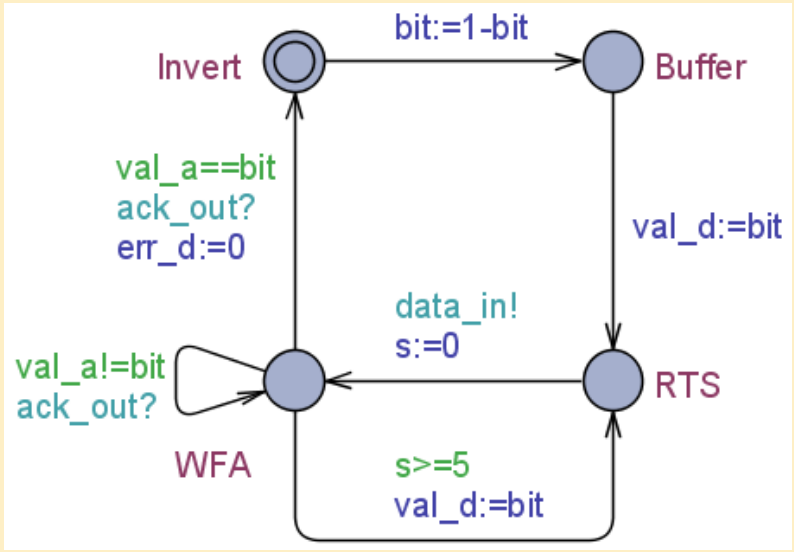
Adat csatorna állapotgráfja

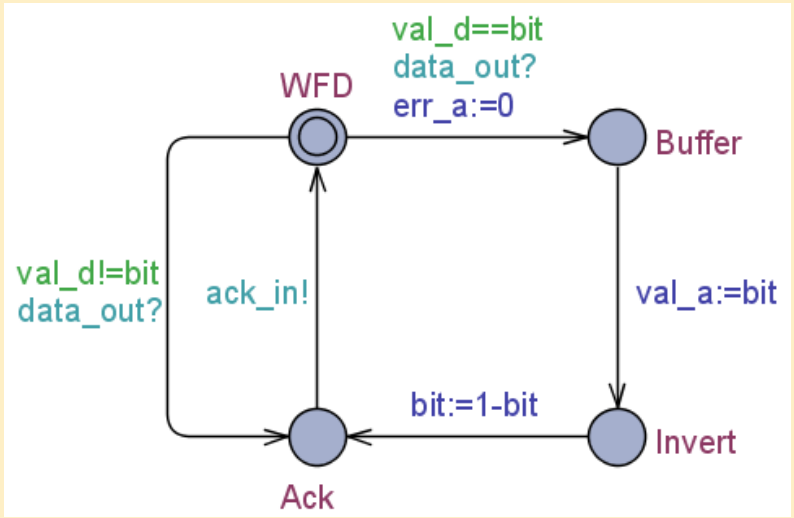


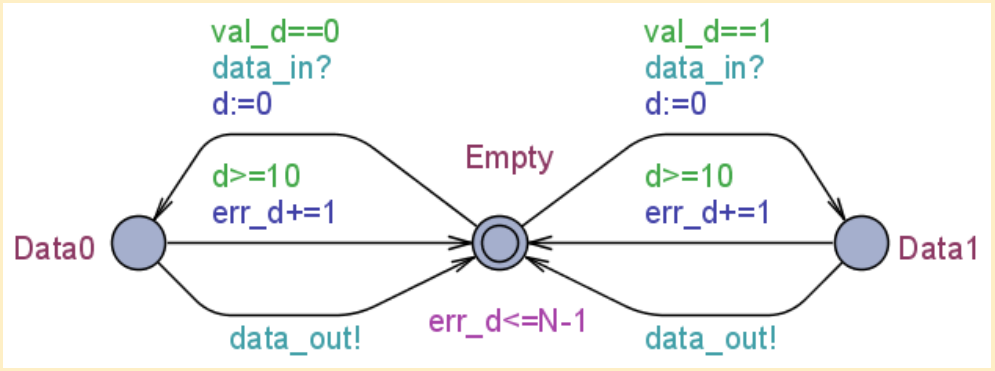
Nyugtázó csatorna állapotgráfja



UPPAAL modell

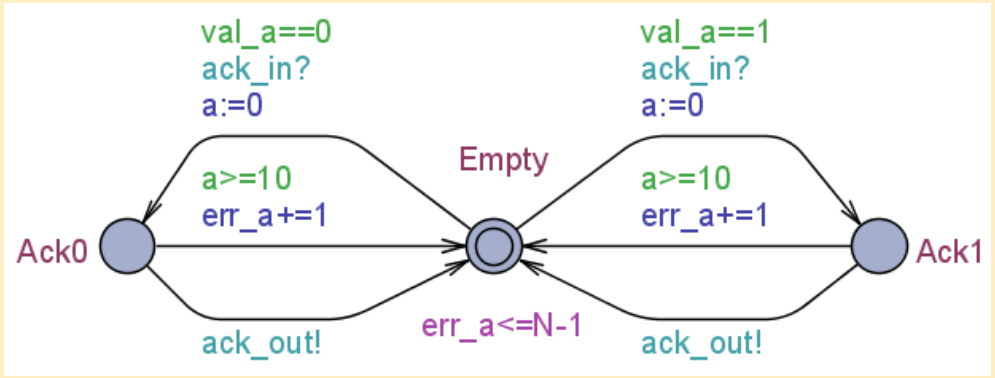




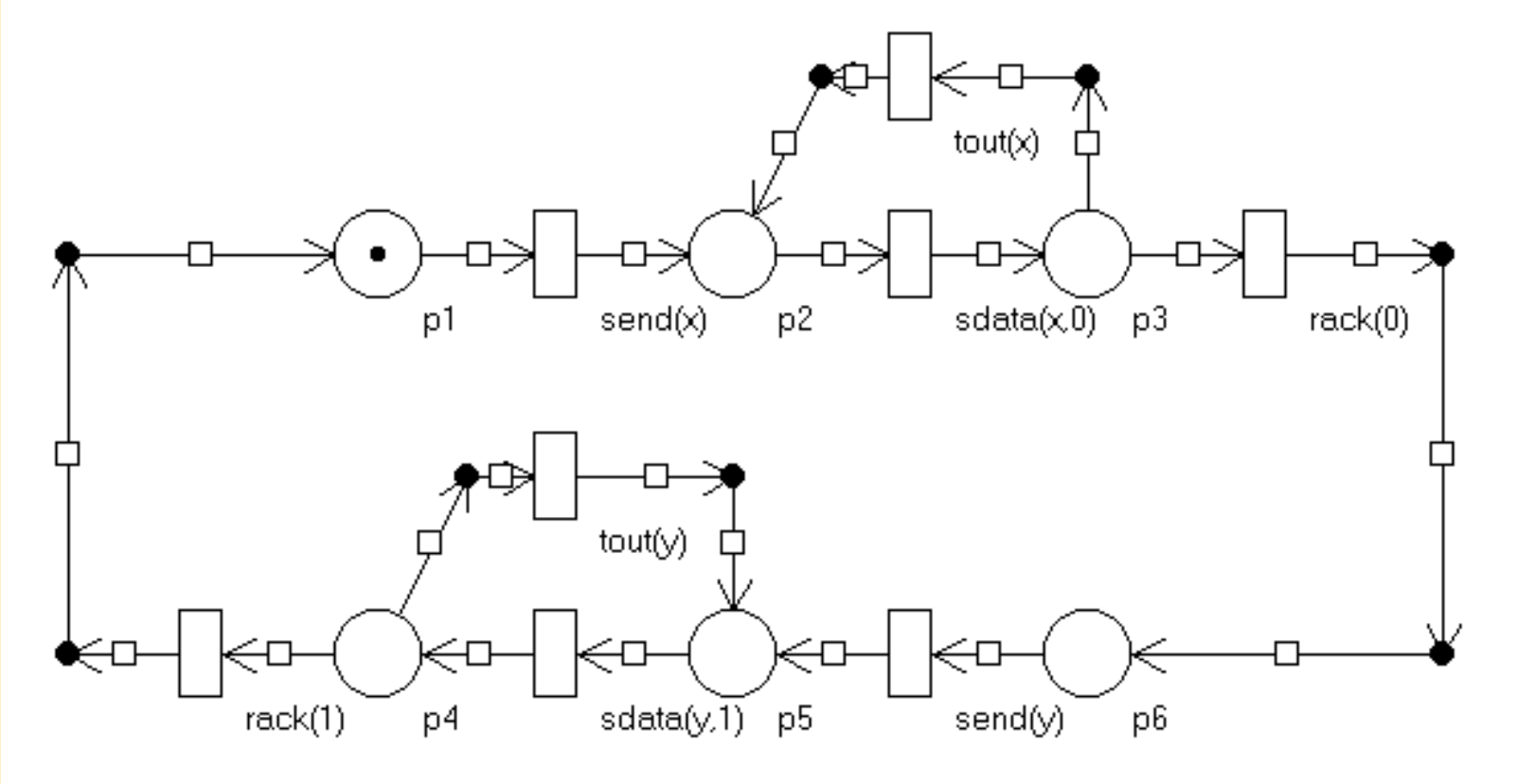



```
const int N=10;

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

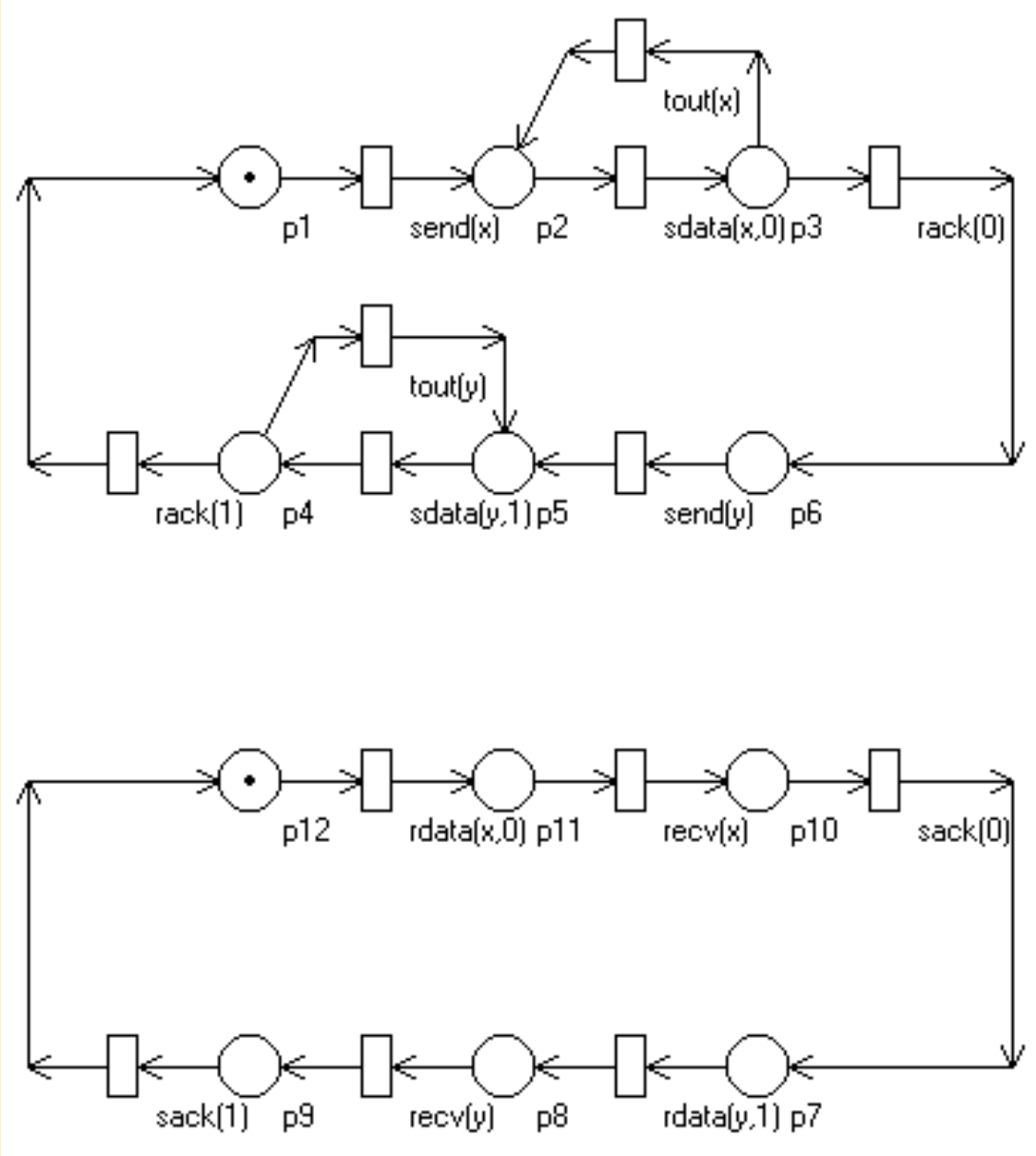
```
value_t val_d, val_a;
error_t err_d, err_a;

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



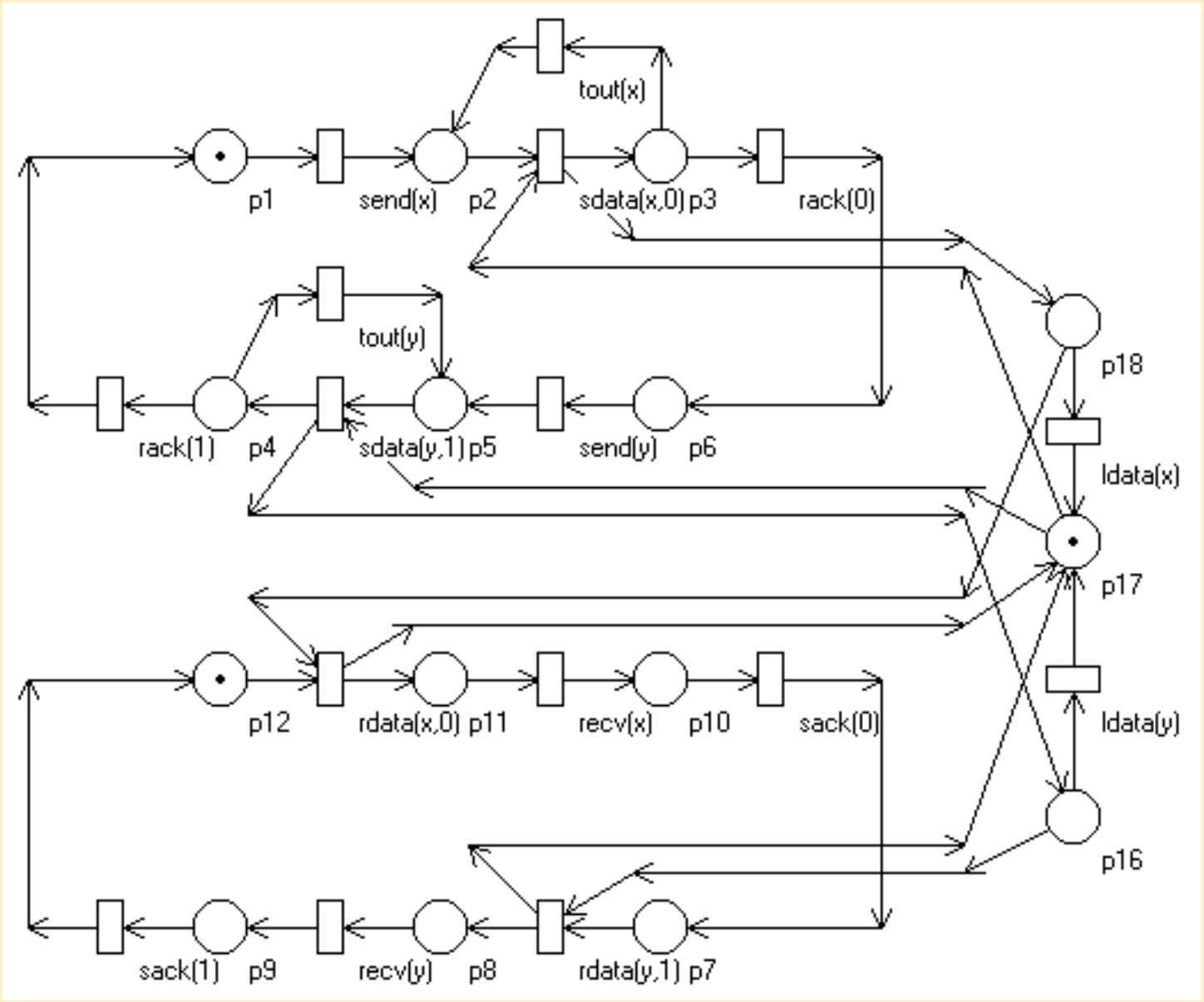
Küldő folyamat Petri háló modellje



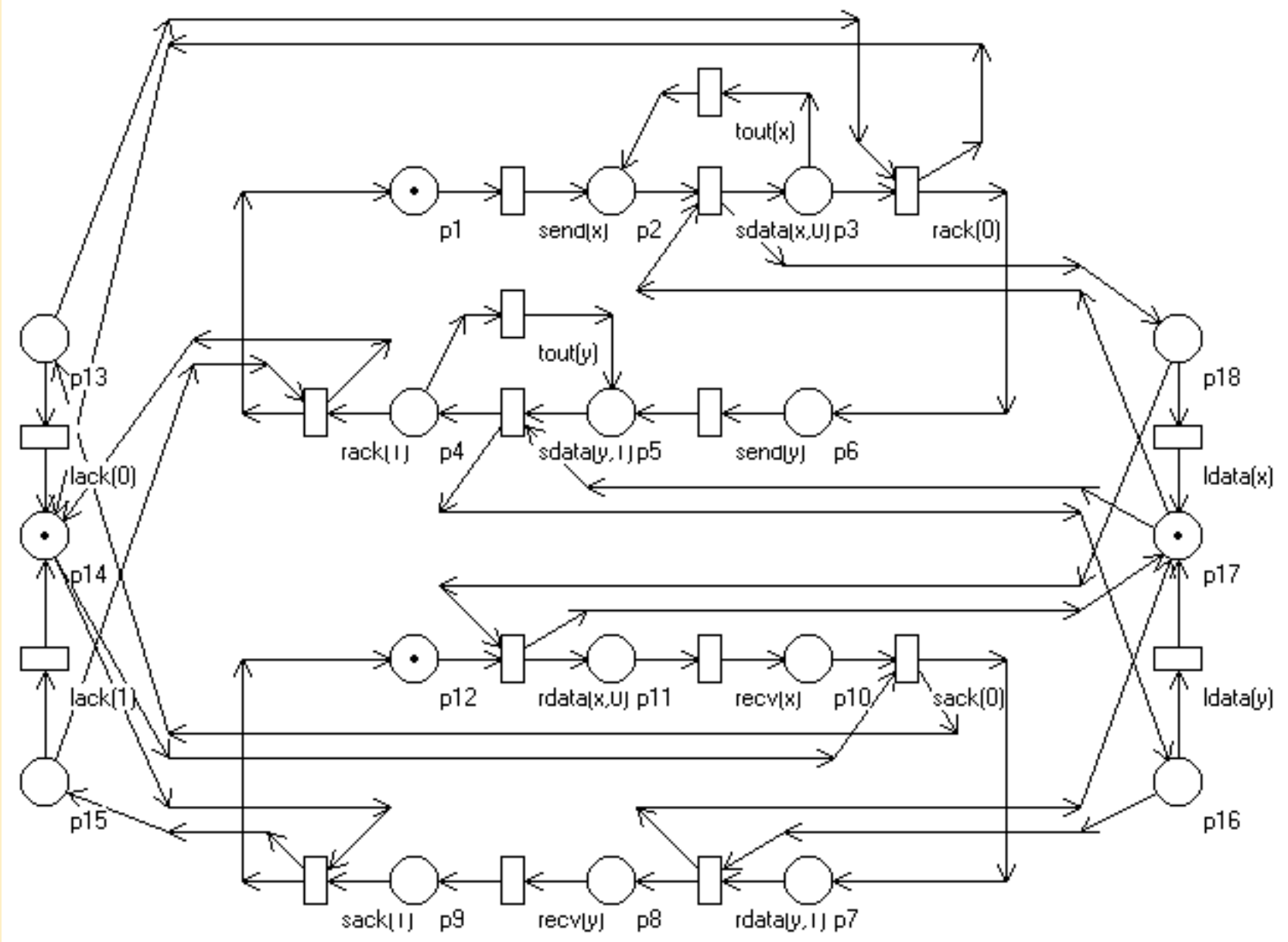
Fogadó folyamat Petri háló modellje



Adat csatorna és az átvitel

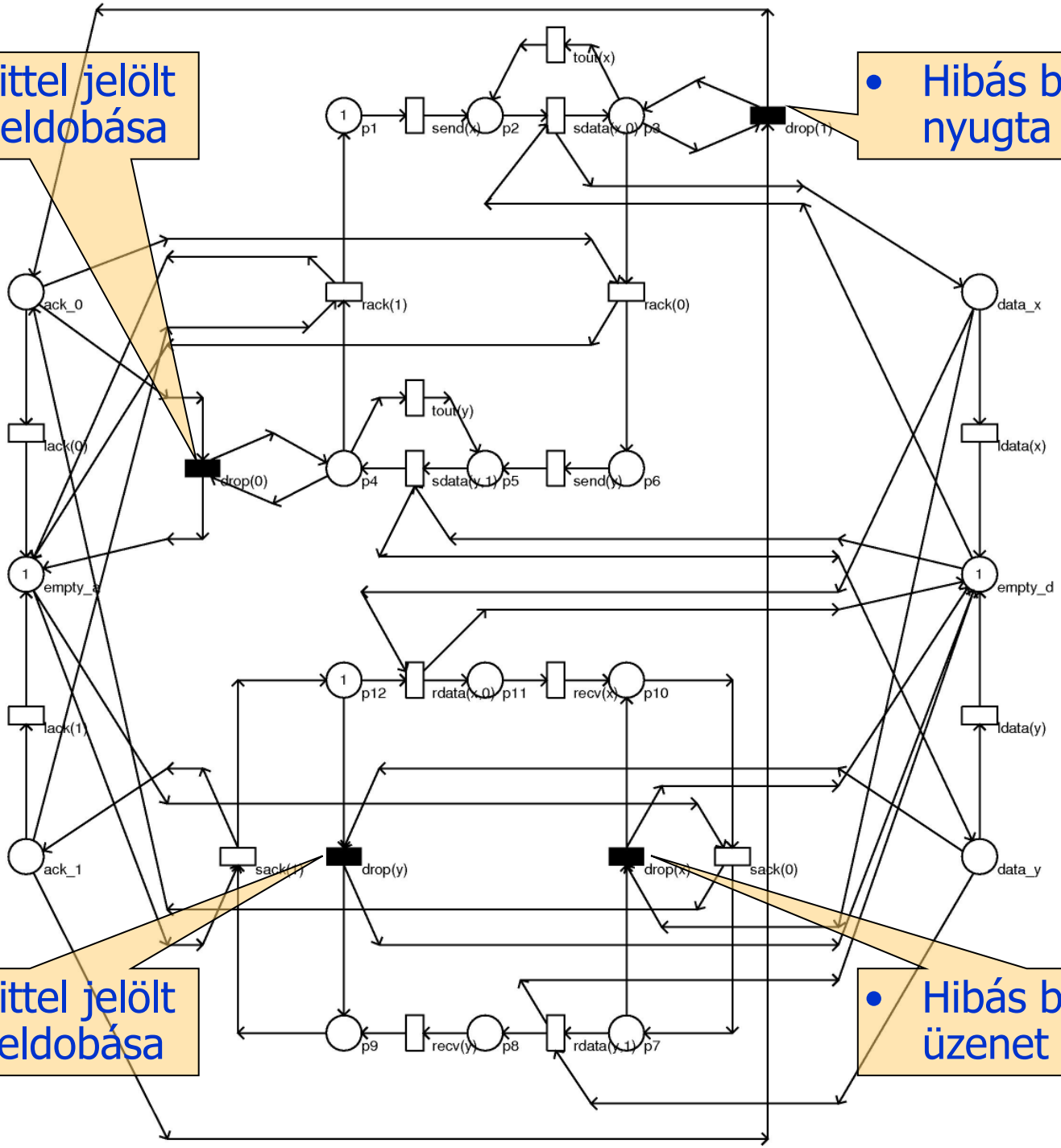


Nyugtázó csatorna



- Hibás bittel jelölt nyugta eldobása

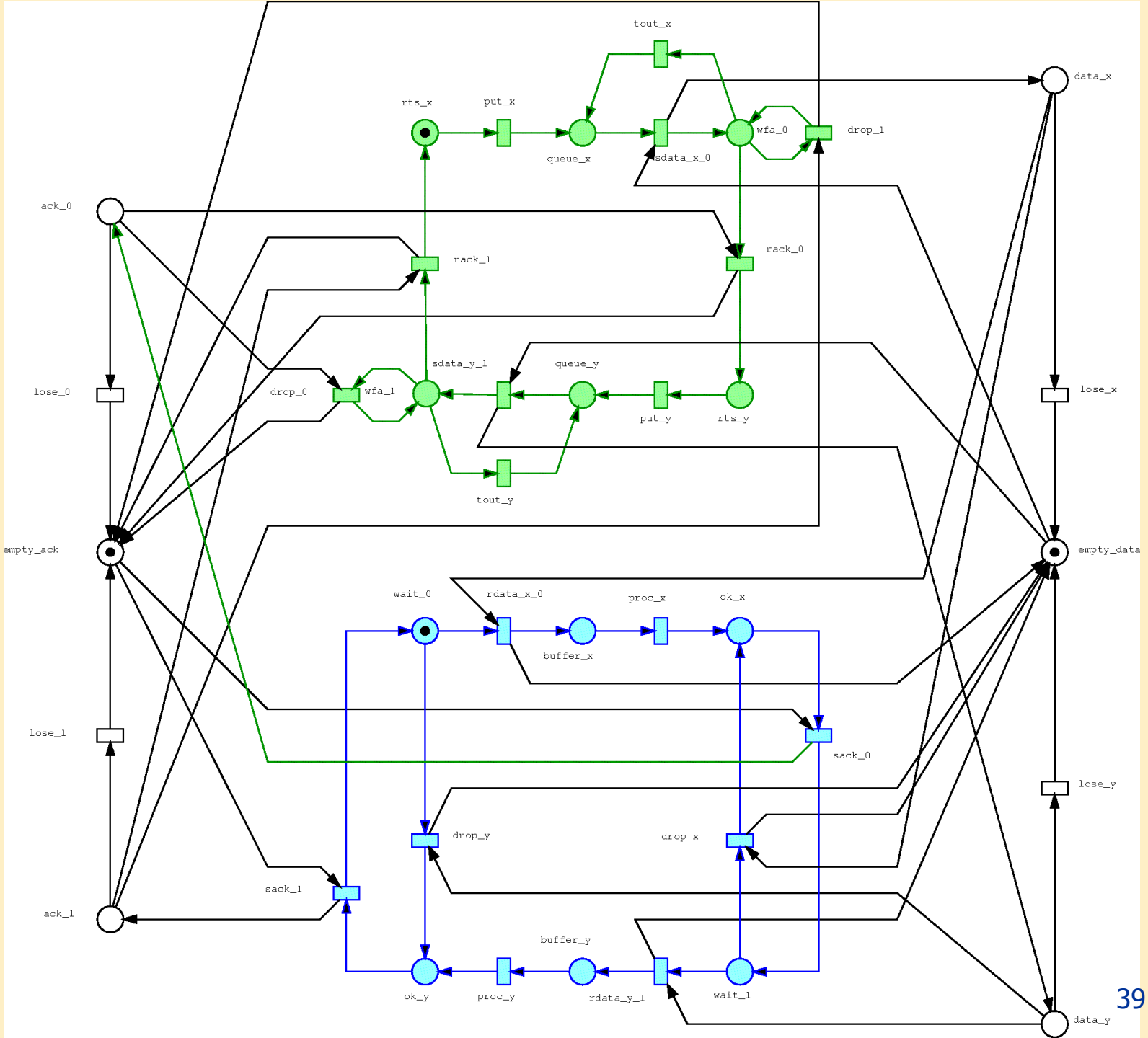
- Hibás bittel jelölt nyugta eldobása



- Hibás bittel jelölt üzenet eldobása

- Hibás bittel jelölt üzenet eldobása

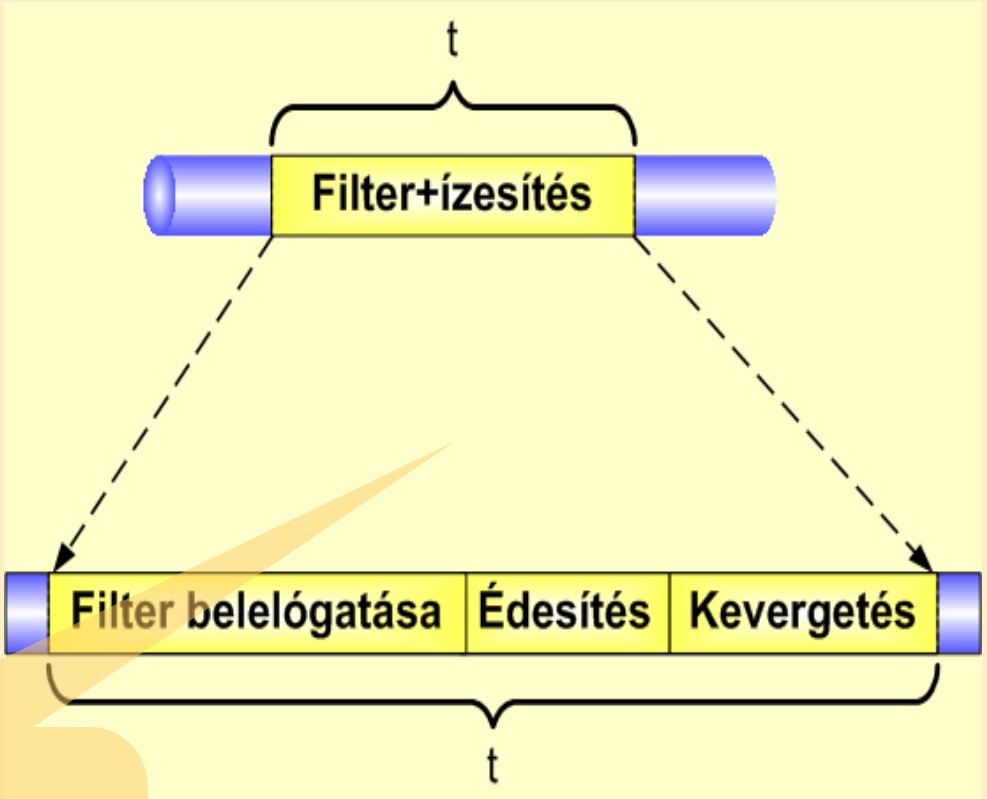
Snoopy: A teljes modell



Hierarchikus Petri hálók
Modellfinomítás

Hierarchikus modellfinomítás

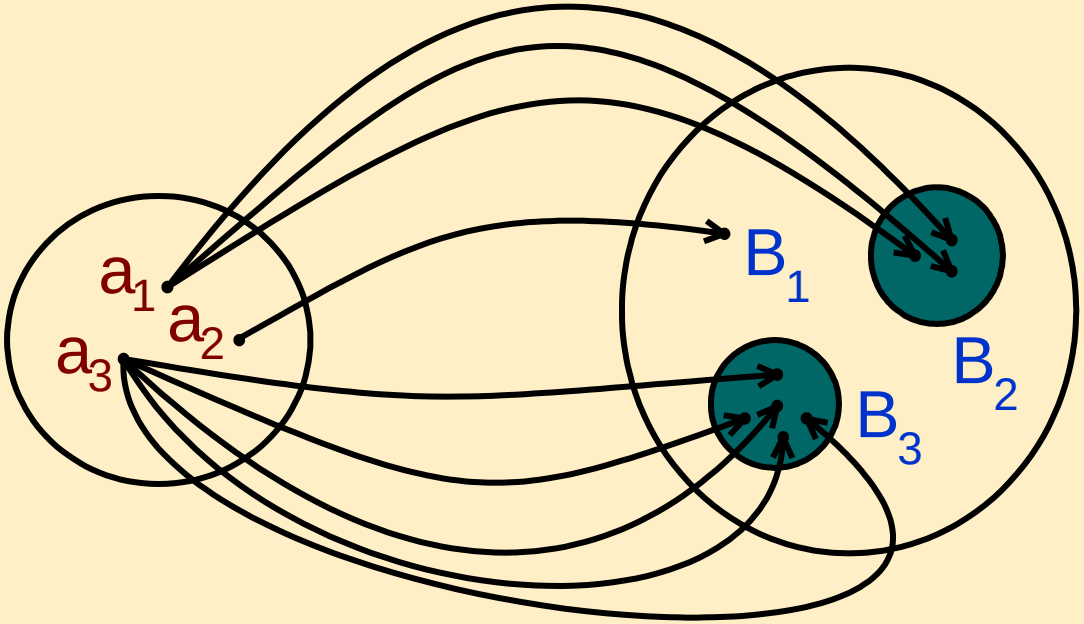
- Elemi eseményeket több részeseményre bontunk fel
- Az új események összideje = a régi esemény ideje
- Azonos bemenetekre azonos „lehetséges” válaszok



Kibontás
„egy az egyben”
behelyettesíthető
KOMPOZÍCIONALITÁS

Halmazfinomítás

Diszjunkt részhalmazok hozzárendelése elemekhez



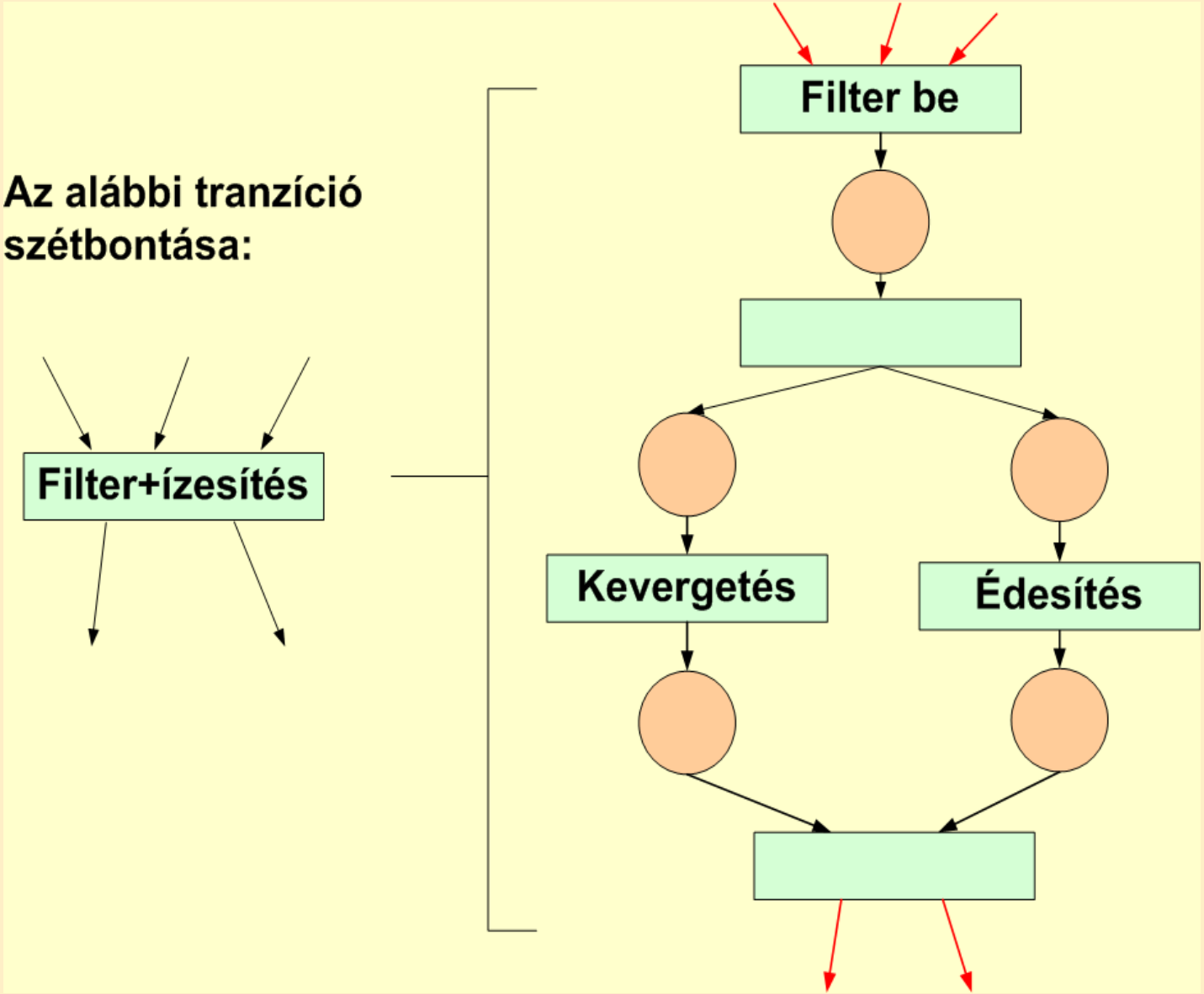
$\forall a_i, \in A$, $R(a_i) \subset B$ úgy, hogy $R(a_i) \cap R(a_j)=0$ $\forall i, j$

Kompozícionalitás (Petri hálók)

- Egy tranzíciót helyettesíthetünk
- A behelyettesítendő gráf
  - Tranzícióval kezdődjön és végződjön
  - Az eredeti tranzíció be/kimenő élei ezekbe menjenek
- Csak akkor használjuk, ha szükséges
  - Kanalat csak a kevergetésre foglalom
    - eredeti modellben nem részletezett tevékenység
  - Komplexitás megnő

Finomítás Petri hálónál

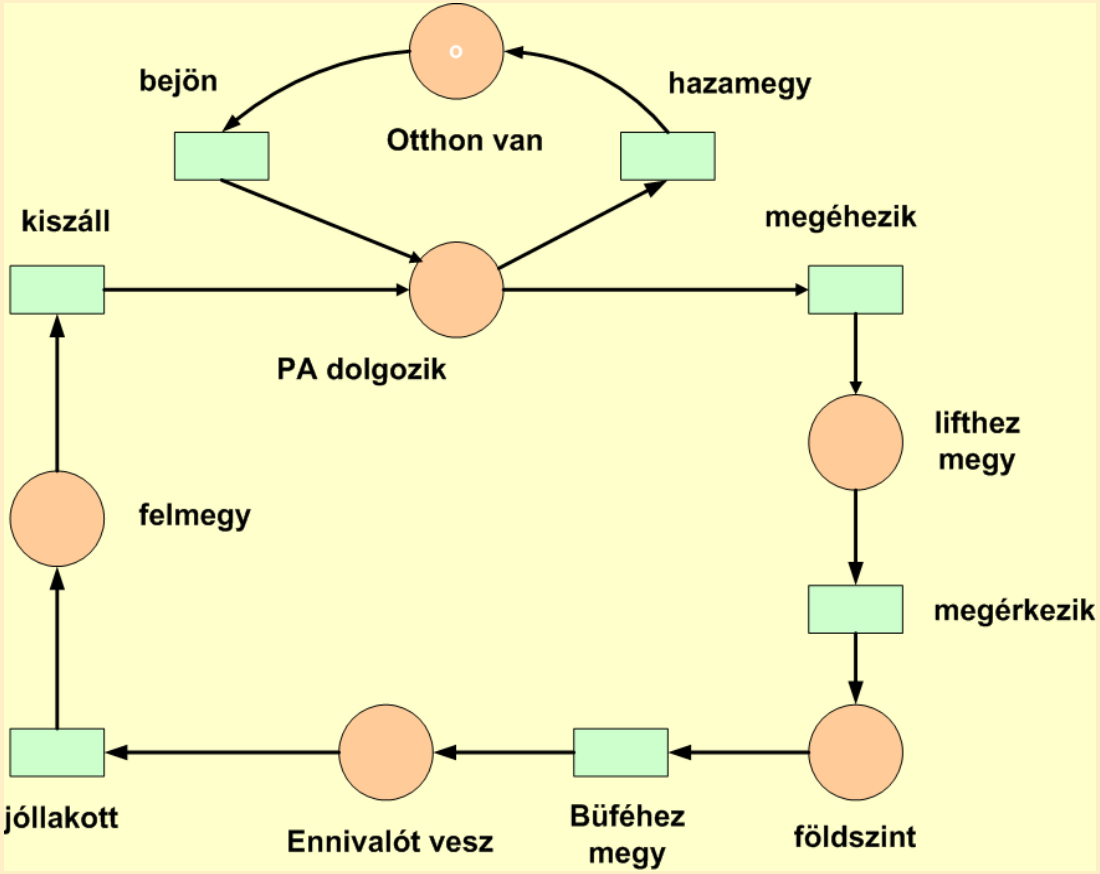
Az alábbi tranzíció szétbontása:



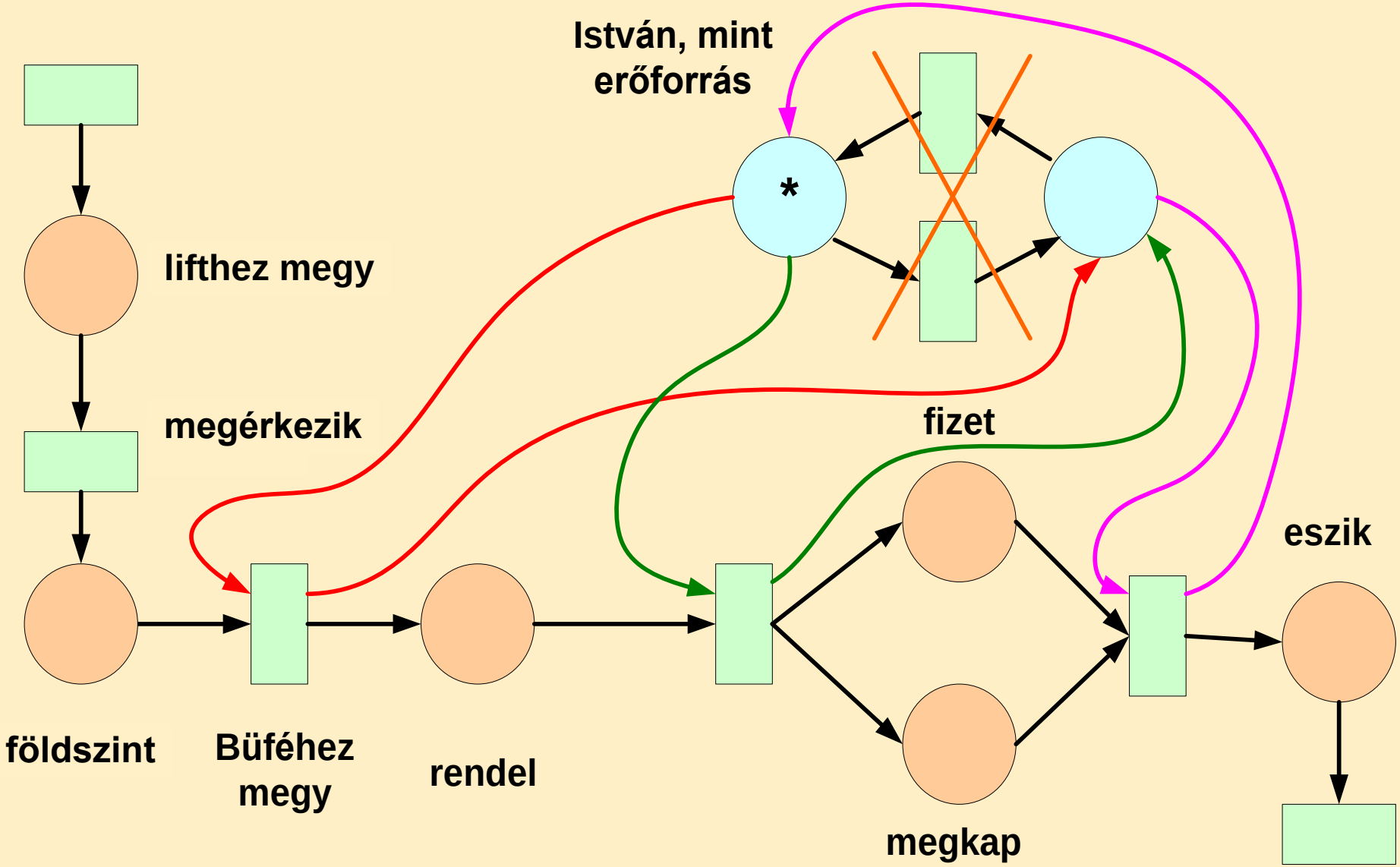
Példa: PA professzor egy napja

Uzsonnázás

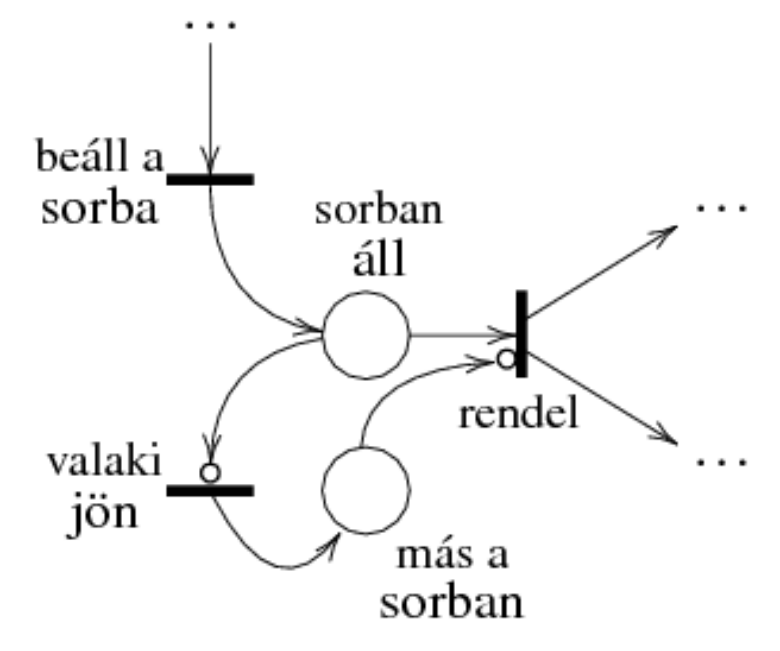
- Erőforrás modellezés
  - professzor (PA)
  - büfés (István)
- Kompozícionalitás
  - étkezés
  - fizetés
- Petri hálóvá transzformálás
  - erőforrások kezelése

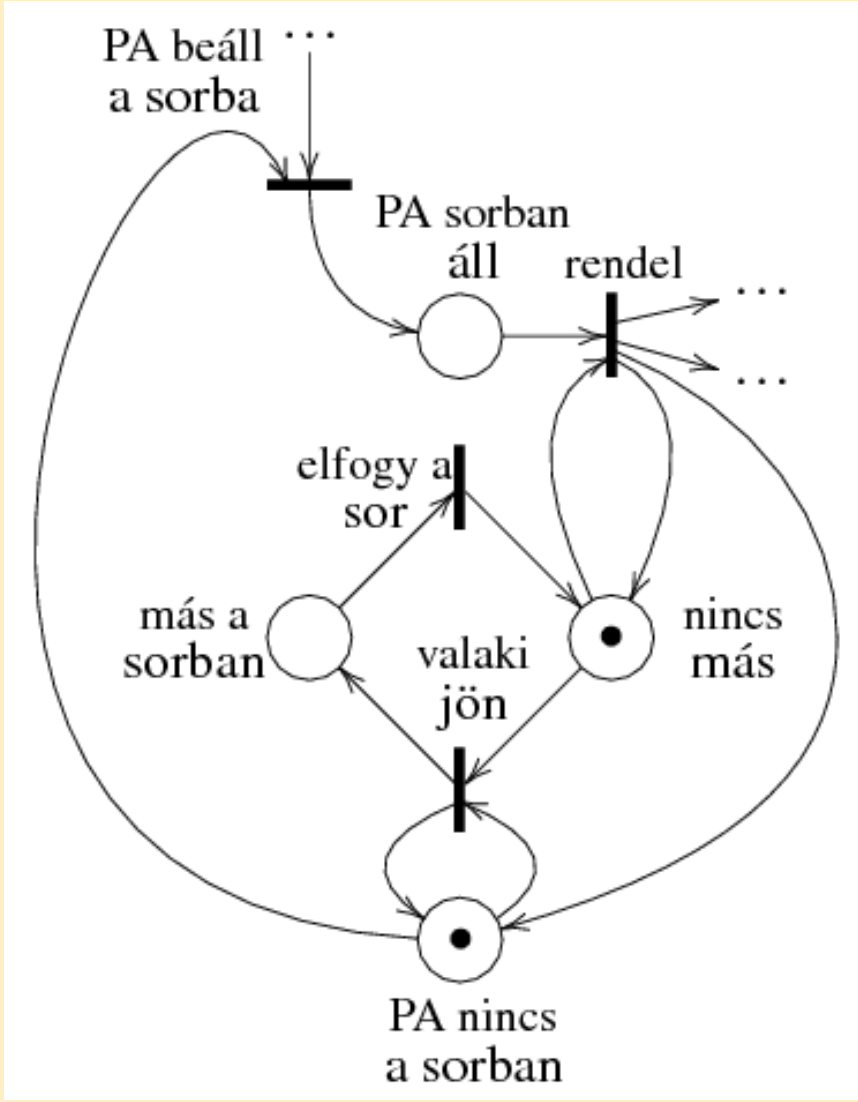


Finomítás, erőforrás modellezés



Tiltó él kiváltása





Hierarchikus modellezés

- Hierarchia:
  - A modell elemei több hierarchia szinten helyezkednek el
    - rendszer > alrendszer > komponens típusú megközelítés
    - modellelemek újrafelhasználása, tipizálás, hibázás csökkentése
- Modellfinomítás (top-down módszer)
  - A modell elemeinek szisztematikus bővítése
  - A bizonytalanságok (nemdeterminizmus) megszüntetése
    - tervezés során a komponenseket részrendszerekké bővítjük
    - fekete dobozból „átlátszó” doboz lesz
- Modellhierarchia kialakítása („bottom-up” jellegű)
  - Bizonyos háló részleteket alhálókkal helyettesítünk
  - Cél a lokális komplexitás csökkentése $\rightarrow$ áttekinthetőség

Hierarchikus modellek

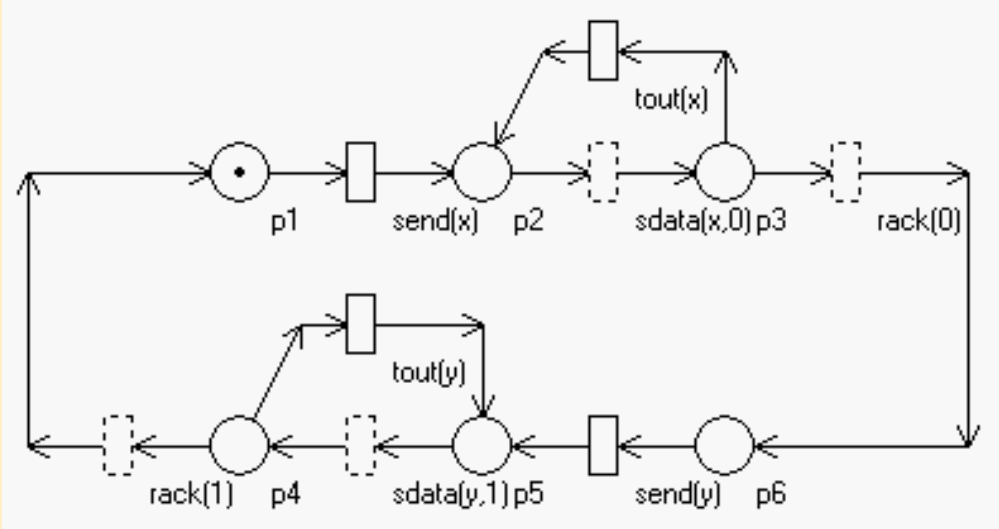
- A hierarchikus felépítés előnyei
  - a részfeladatok megoldására koncentrál
  - egyszerűsíti a leírást az egyes szinteken
  - áttekinthető a részrendszerek kapcsolata
- A hierarchia eszközei
  - alhálók, „fő” háló
  - interakciós pontok: helyek és átmenetek
  - subnet-to-subnet élek

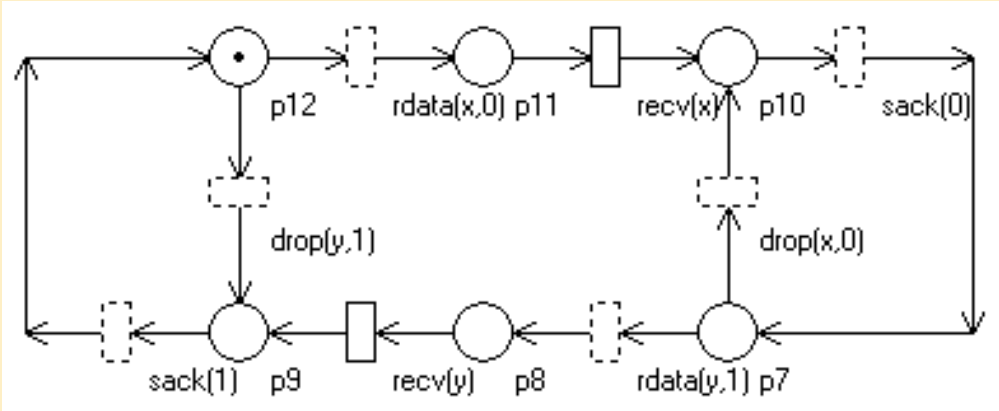
Hierarchikus felépítés eszközei

- PetriDotNet
  - Modellfinomítás
    - Helyettesítő tranzíció (Coarse transition): tranzícióból alháló
- Snoopy
  - További elemek
    - Helyettesítő hely (Coarse place): helyből alháló
    - Környezettel való kapcsolat szerint tranzíció / hely jellegűek
    - Viszont nem lehetnek egymással közvetlen kapcsolatban!
  - Áttekinthetőség növelése
    - Logikai tranzíció (Logic / Fusion transition): globális tranzíció
    - Logikai hely (Logic / Fusion place): globális hely
    - A keresztül-kasul haladó élek számát csökkentik

DNAnet: A küldő és a fogadó folyamat

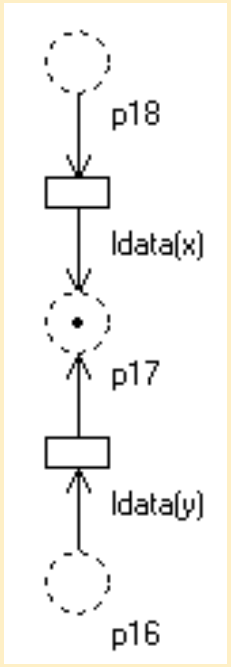
- Nem strukturált eszköz: alháló
  - Bármit helyettesíthet, de emiatt kevésbé áttekinthető
  - Nem támogatja a szisztematikus modellbővítést, modellfinomítást
  - Támogatja a modellelemek újrafelhasználását

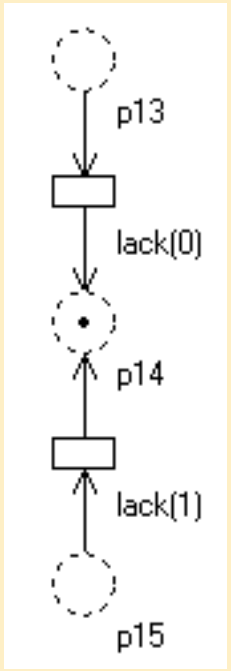


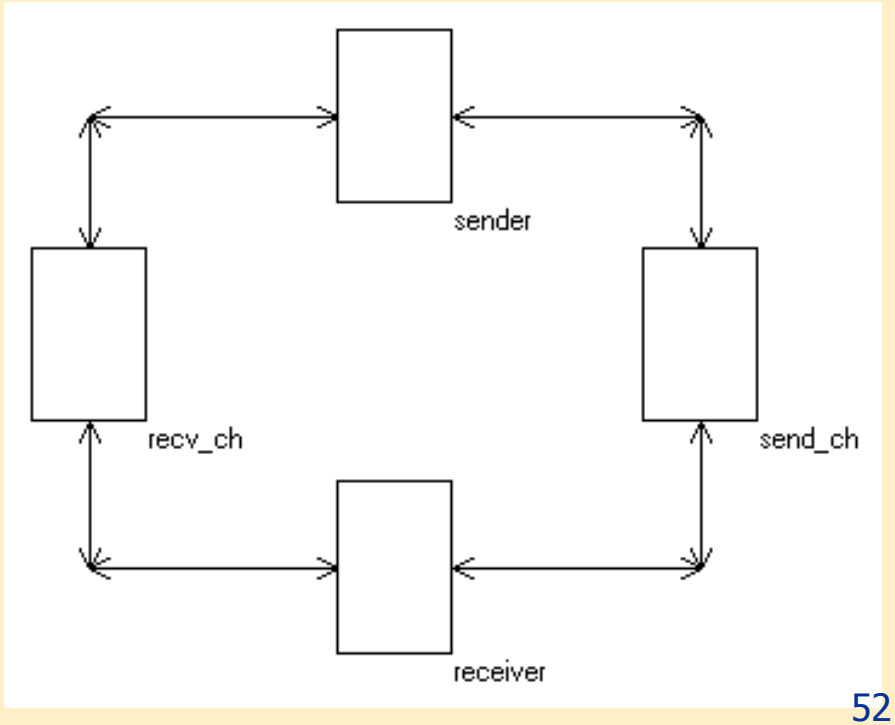


DNAnet: Az alrendszerek egyesítése

- A csatornák azonos felépítése jól látható
  - Interakció: helyekkel
  - A komponensek kapcsolata a fő hálóban
- Subnet komponens
- Alhálók közti élek
  - forrás interakciós pontok
  - cél interakciós pontok

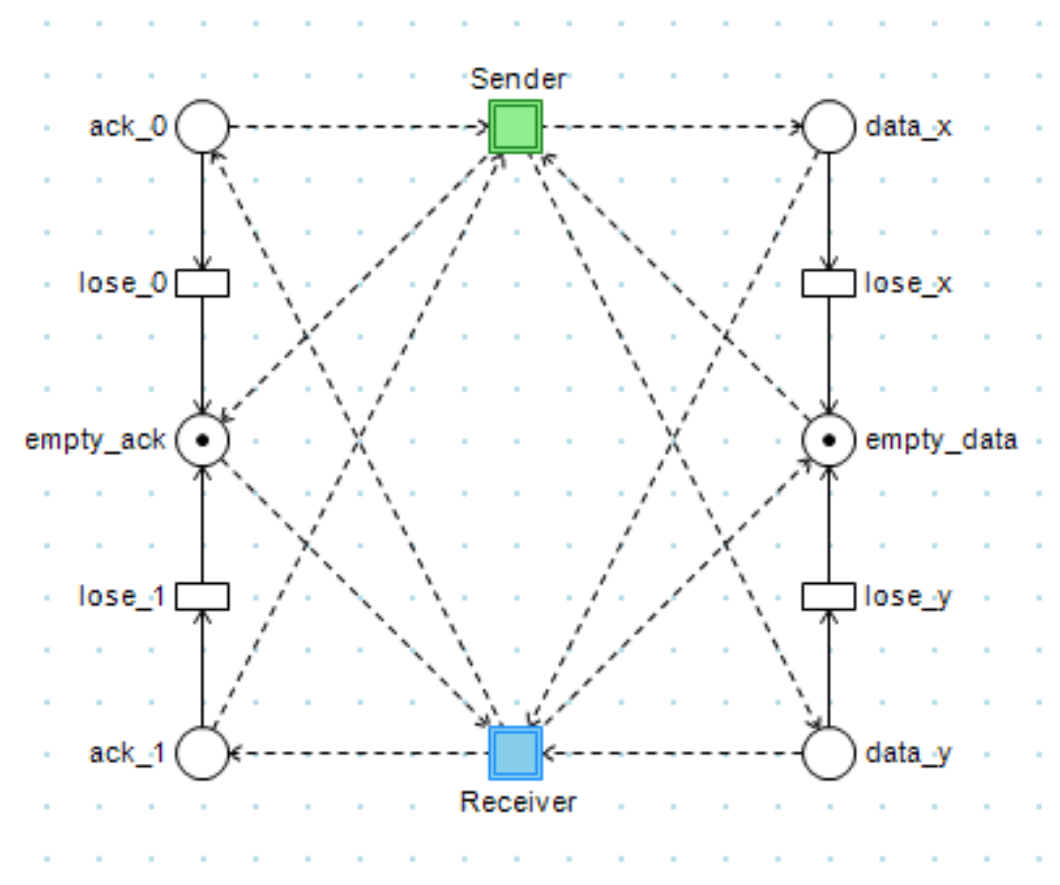




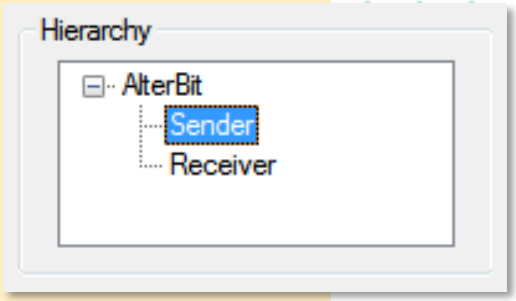


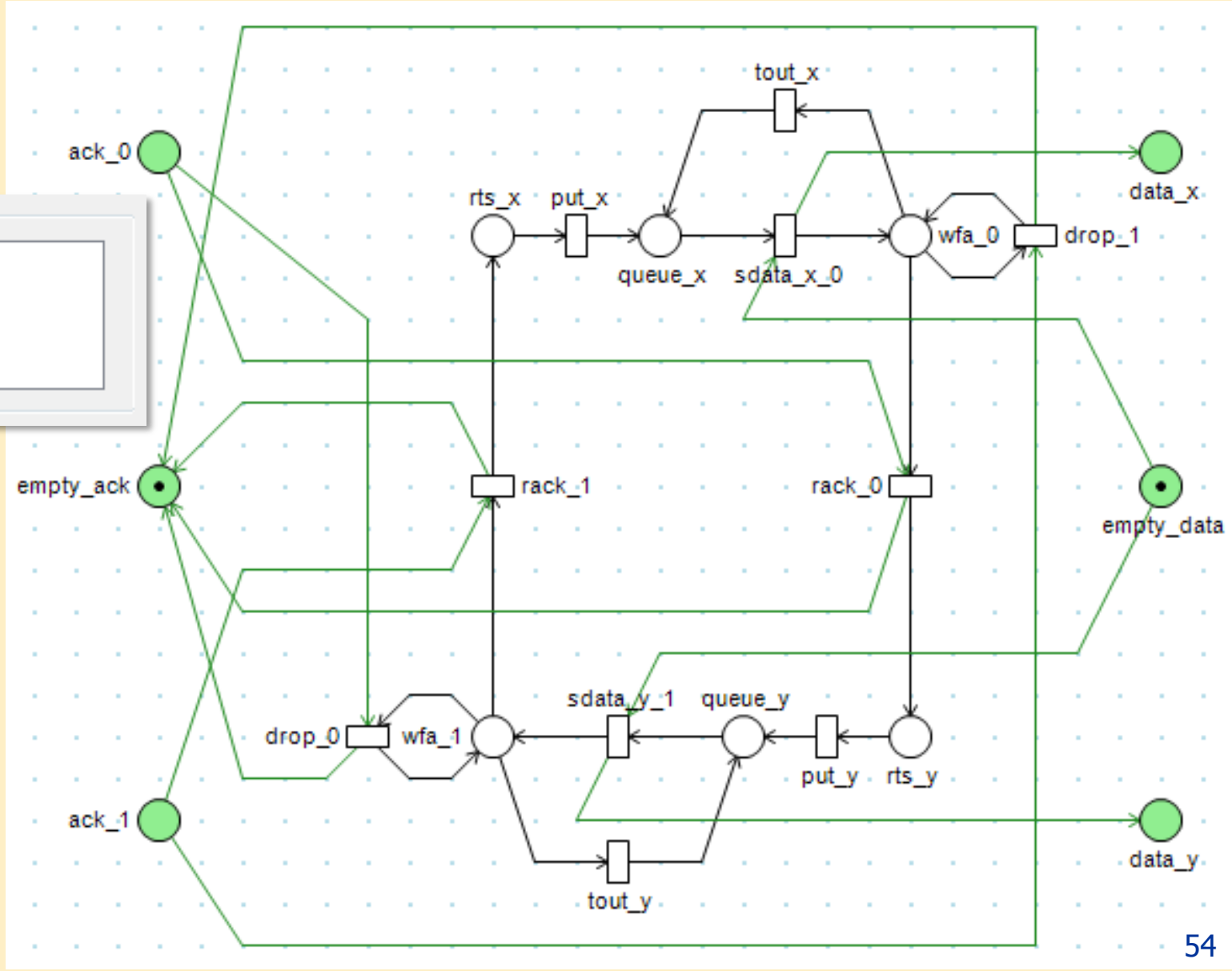
PetriDotNet, Snoopy: Alrendszerek egyesítése

- Más megközelítés
- Kapcsolt helyek, átmenetek
  - alháló szinten láthatóvá válnak
  - átszerkeszthetők
- Alháló: coarse node
  - hely vagy átmenet
  - kevésbé rugalmas
  - kapcsolódó éleket ajánlott előbb

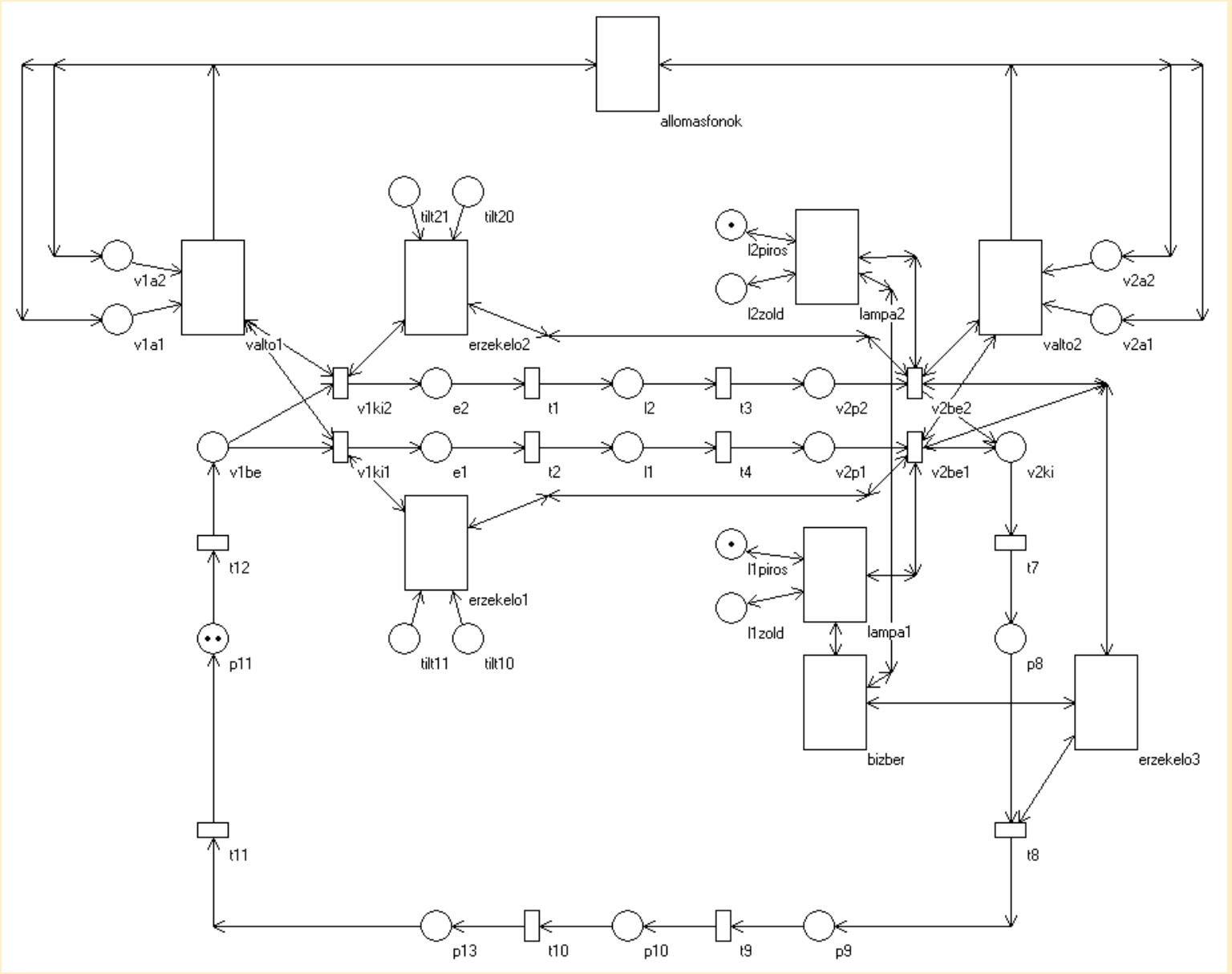


PetriDotNet: Küldő folyamat alhálója

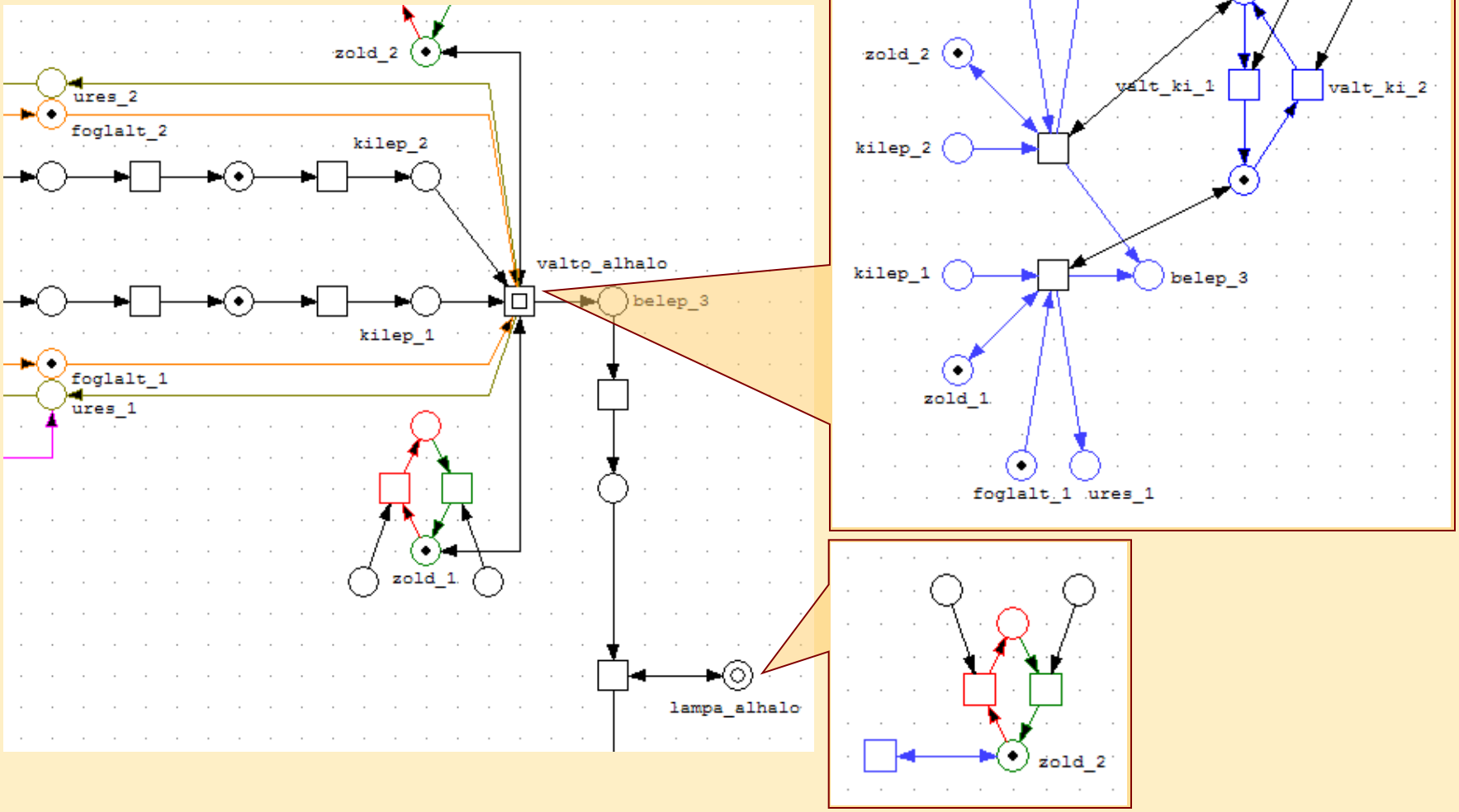




Modellvasút hierarchikus modellje DNAnet-ben



Váltó és szemafor megvalósítása alhálóval

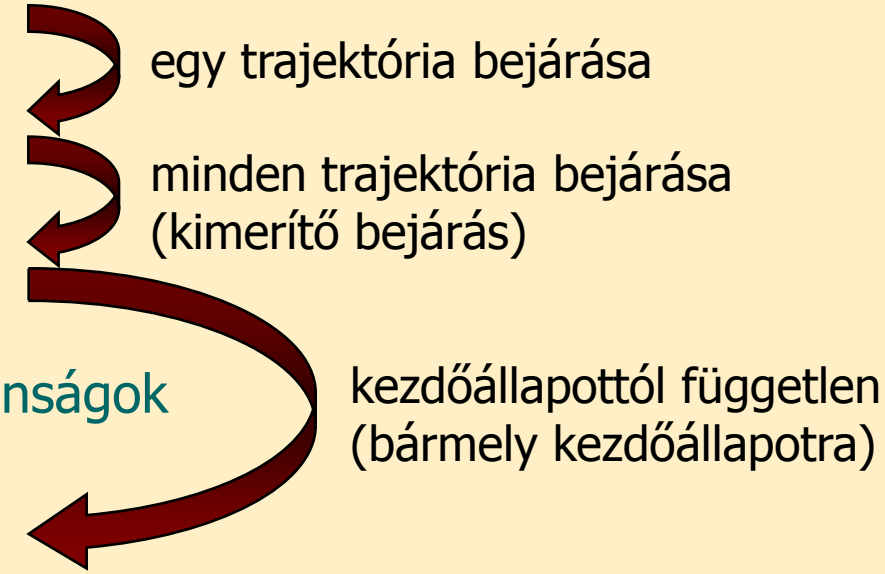


Petri háló modellek „tesztelése”
Szimuláció, token játék algoritmusai

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

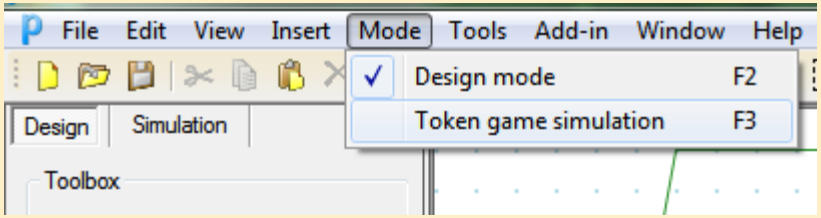
Diszkrét rendszerek szimulációja

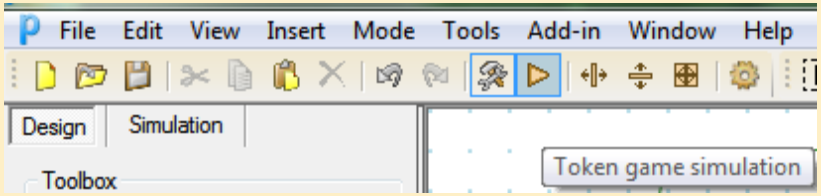
- Cél: a vizsgált rendszer „valósághű” modellezése
  - valósághű ~ validáció (modell megfelelőség)
- Szimuláció folyamatmodellek esetén
  - tevékenységorientált (eseményorientált)
  - csak az események időpontjait tartjuk nyilván
    - tevékenységek kezdete és vége (vagy időtartama)
    - erőforrások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

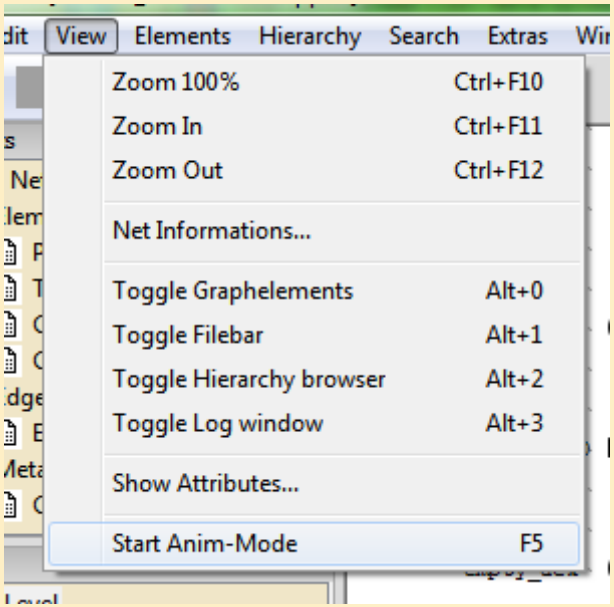
Petri hálók szimulációja

- 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)

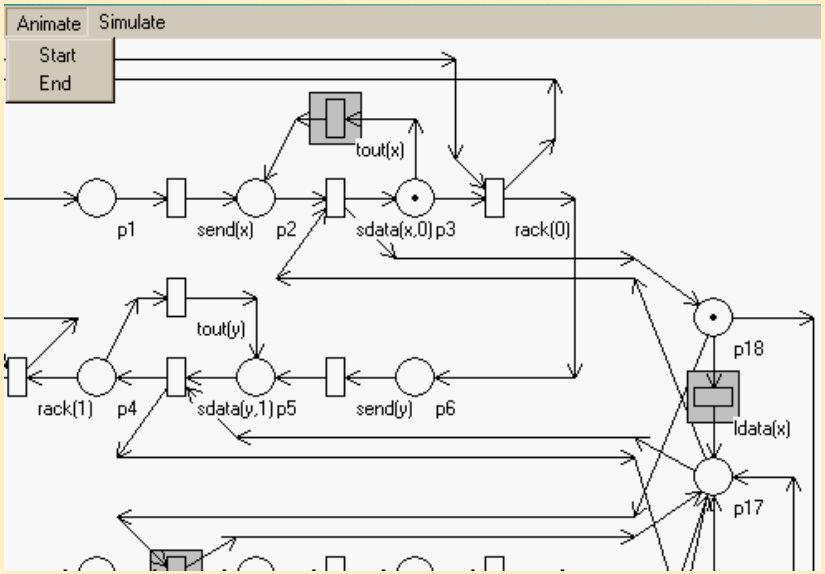


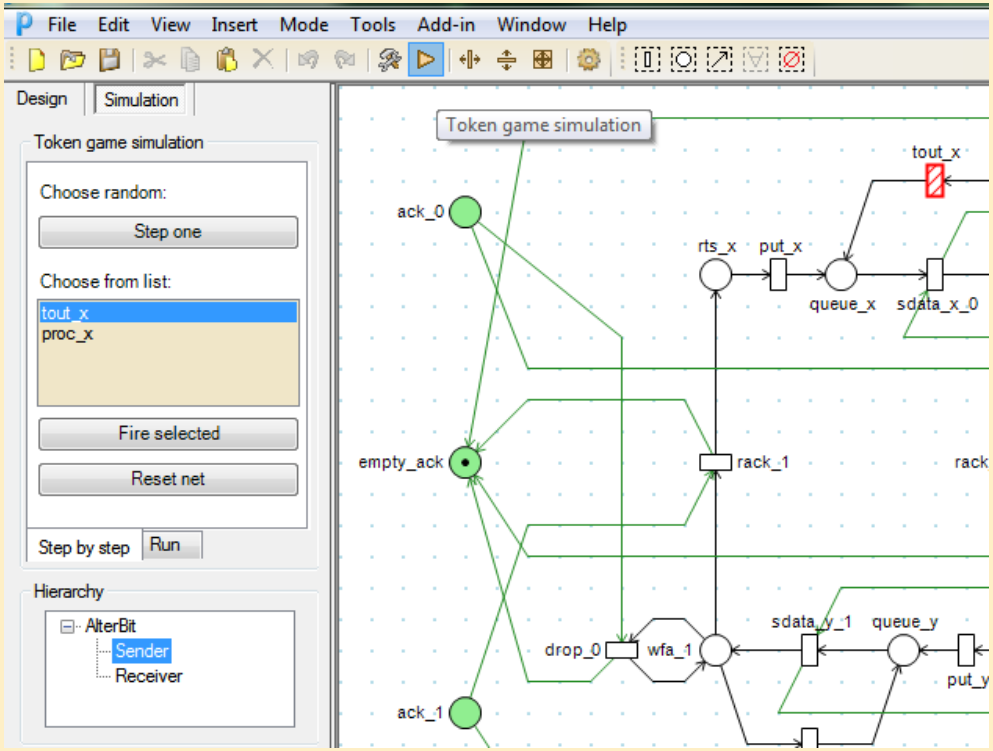


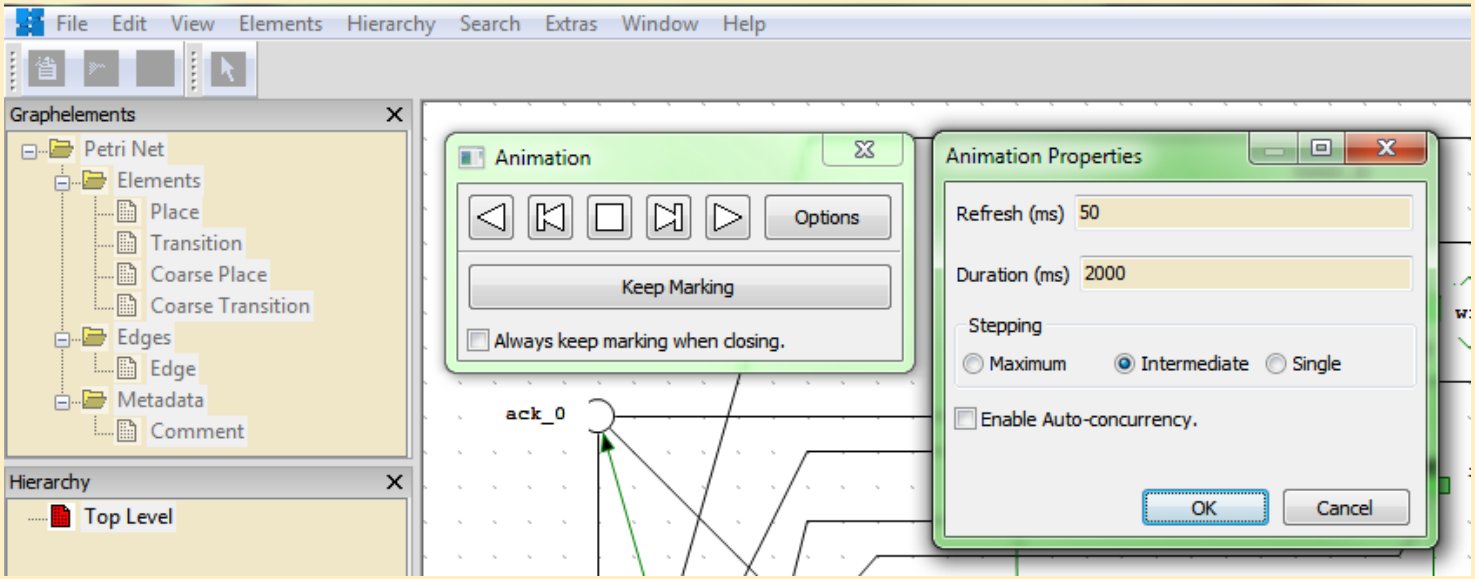


- A modell interaktív ellenőrzése
- Engedélyezett átmenetek jelölve (DNAnet)
- Jelölt átmenetre kattintva tüzel
- Előállítja az új tokeneloszlást
- Konkurens átmenetek:
  - manuálisan
  - PetriDotNet, Snoopy: véletlen választással is
- Befejezéskor visszaállítja a kezdeti tokeneloszlást

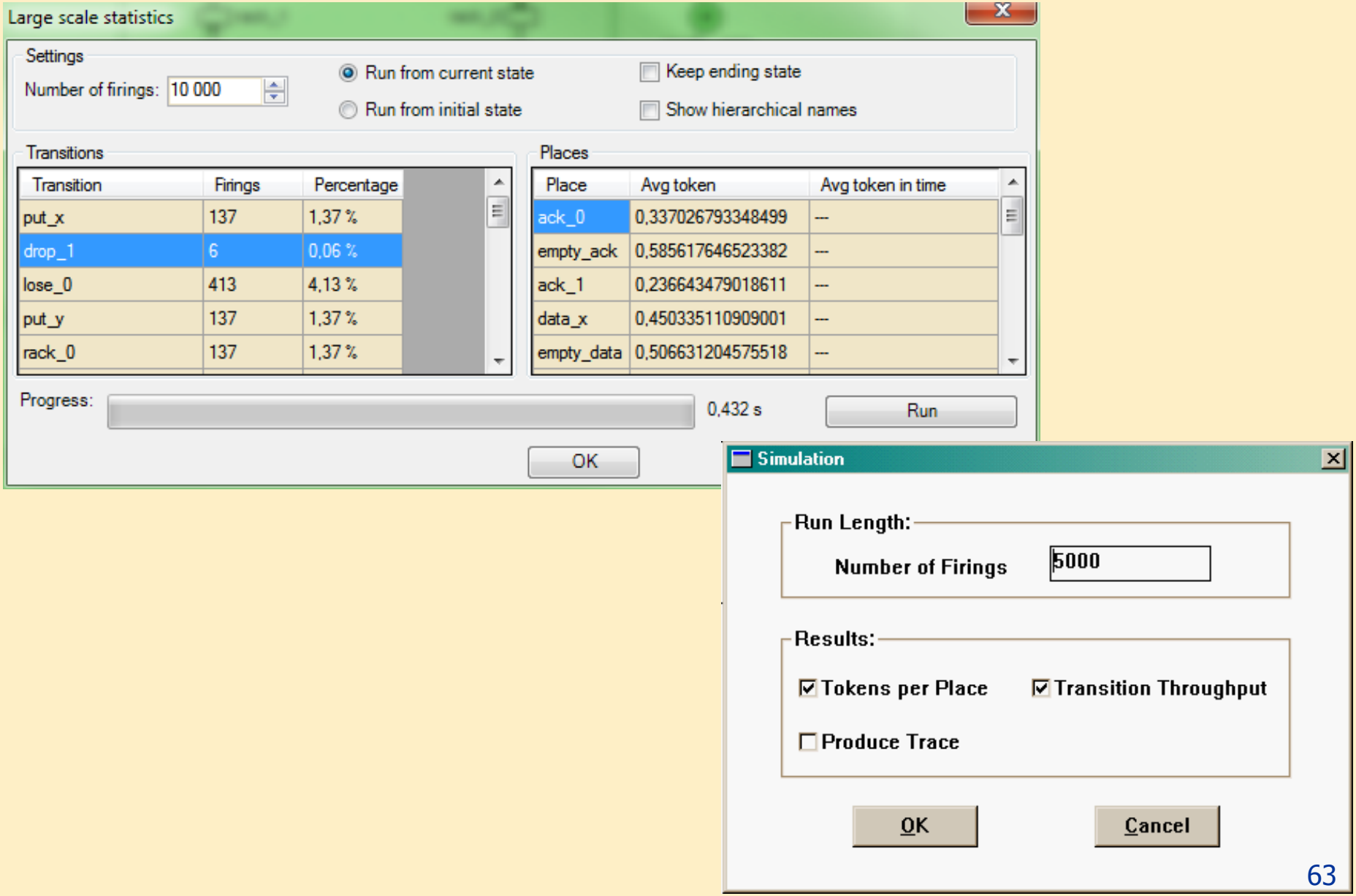
Animációs képernyő







Szimuláció


Large scale statistics
Settings
Number of firings: 10 000
Run from current state
Run from initial state
Keep ending 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

    $L_{fireable} \leftarrow \emptyset$

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

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

    return $L_{fireable}$

end function

Az állapotváltozás nagysága

Ha t tüzel M állapotban

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

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

Tüzelés

// Inicializálás

$M \leftarrow \mathrm{M}_0$

$L_{fireable} \leftarrow collect\_fireable\_transitions(M)$

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

    // Tüzelés

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

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

    $L_{fireable} \leftarrow collect\_fireable\_transitions(M')$

    $M \leftarrow M'$

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 hozzátett tokenek száma: $M^+ = \mathbf{W}^{+\mathbf{T}} \cdot \underline{e}_t$
  - 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

$M \leftarrow \mathrm{M}_0$

$L_{fireable} \leftarrow \emptyset$

// Tüzelhető tranzíciók kezdeti halmaza

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

    if $enabled(t, \mathrm{M}_0)$ **then** $L_{fireable} \leftarrow L_{fireable} \cup \{t\}$

Hatékony algoritmus: tüzelési ciklus

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

// Tüzelés

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

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

// Letiltottak eltávolítása

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

if $not(enabled(t', M'))$ **then** $L_{fireable} \leftarrow L_{fireable} \setminus \{t'\}$

// Engedélyezettek bevonása

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

if $enabled(t'', M')$ **then** $L_{fireable} \leftarrow L_{fireable} \cup \{t''\}$

$M \leftarrow M'$

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

$M \leftarrow \mathrm{M}_0$

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

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

// Tüzelhető tranzíciók kezdeti halmaza

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

    if $enabled(t, \mathrm{M}_0)$ **then** $L_{fireable}[\pi(t)] \leftarrow L_{fireable}[\pi(t)] \cup \{t\}$

Prioritásos algoritmus: tüzelési ciklus

while $\bigcup_{\pi\in\Pi} L_{fireable}[\pi] \neq \emptyset$ **do**

  for $\pi = \pi_{max}$ **to** π_{min} **step** -1 **do** // Tüzelés

    if $L_{fireable}[\pi] \neq \emptyset$ **then**

      $t \leftarrow rnd(L_{fireable}[\pi])$

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

      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** $L_{fireable}[\pi(t')] \leftarrow L_{fireable}[\pi(t')] \setminus \{t'\}$

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

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

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

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

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

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

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 *not*(*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**

$\tau'' \leftarrow time(t'')$

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

end for

end procedure

Főprogram: inicializálás

// Inicializálás

$M \leftarrow \mathrm{M}_0$

$\tau_{simulation} \leftarrow 0$

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

$L_{immadiate}[\pi] \leftarrow \varnothing$

// 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** $\bigcup_{\pi \in \Pi} L_{immediate}[\pi] \neq \emptyset$ **do**

  for $\pi = \pi_{max}$ **to** π_{min} **step** -1 **do**

    if $L_{immediate}[\pi] \neq \emptyset$ **then**

      // Azonnali tranzíció tüzelése

      $t \leftarrow rnd(L_{immediate}[\pi])$

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

      $maintenance(t, M')$

      $M \leftarrow M'$

      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

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

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

$maintenance(t, M')$

$M \leftarrow M'$

goto MAIN

end if

end.