Adatfolyam hálók
Finomítás: ellenőrzött modellváltoztatások
Hierarchikus Petri hálók

dr. Bartha Tamás
Dr. Pataricza András

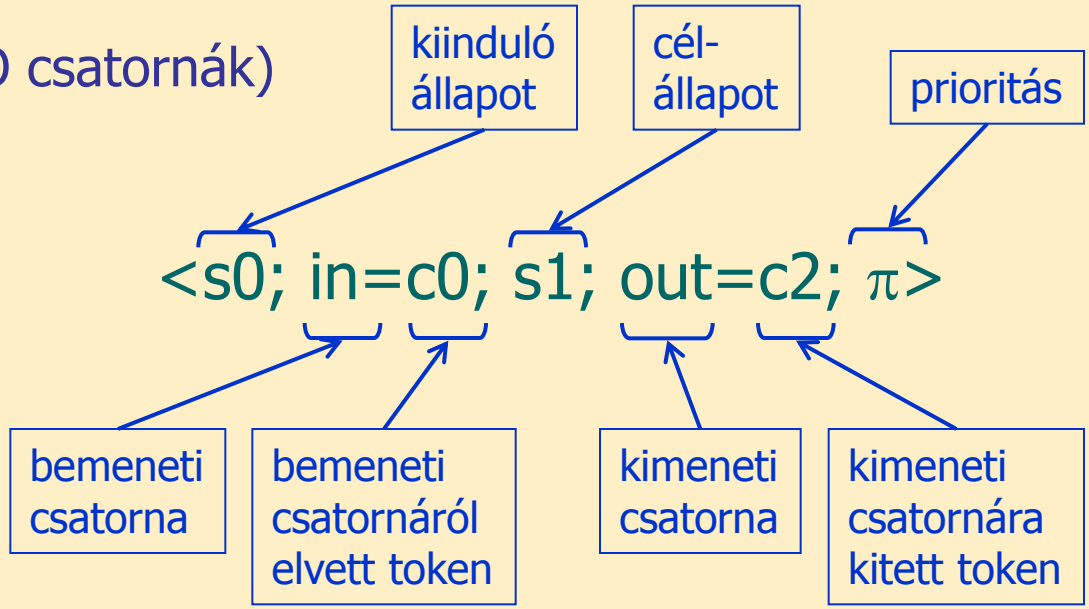
BME Méréstechnika és Információs Rendszerek Tanszék

Adatfolyam modellezés

Nem determinisztikus DFN formalizmus

- [Jonsson, Cannata]

- Struktúra
  - Adatfolyam gráf (DFG)
    - csomópontok
    - irányított élek (FIFO csatornák)
- Viselkedés
  - Tüzelési szabályok:
- Adatok
  - (Színes) tokenek



Adatfolyam háló formalizmus

Adatfolyam hálózat formális leírása

- **Adatfolyam hálózat**: egy hármas (*N*, *C*, *S*)
  - *N* : csomópontok halmaza
  - *C* : csatornák halmaza
    - I: bemenő csatornák
    - O: kimenő csatornák
    - IN: belső (csomópontok közötti) csatornák

    } kapcsolat a külvilággal
  - *S* : állapotok halmaza
- **Adatfolyam csatorna**:
  - (alapértelmezés: végtelen kapacitású) FIFO csatorna,
  - egy bemeneti és egy kimeneti csomóponthoz kötve
  - állapota: $S_c = \times^{\infty} M_c$ tokenszekvencia

Adatfolyam csomópont formális leírása

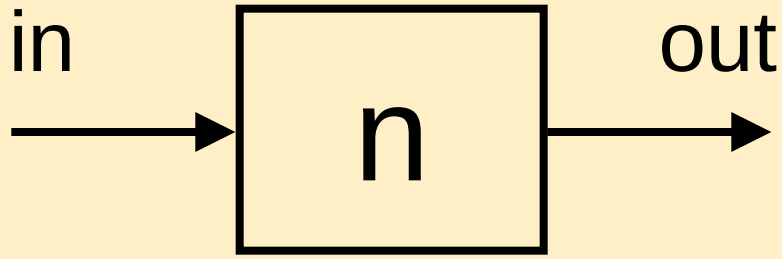
Adatfolyam csomópont: $n = (I_n, O_n, S_n, s_n^0, R_n, M_n)$, ahol

- I_n – bemenő csatornák halmaza
- O_n – kimenő csatornák halmaza
- S_n – csomópont állapotok halmaza
- s_n^0 – csomópont kezdőállapota, $s_n^0 \in S_n$
- M_n – tokenek halmaza
- R_n – tüzelések halmaza, $r_n \in R_n$ egy ötös $(s_n, X_{in}, s'_n, X_{out}, \pi)$
  - s_n – tüzelés előtti és utáni állapotok, $s'_n \in S$
  - X_{in} – bemenő leképzés, $X_{in} : I_n \to M_n$
  - X_{out} – kimenő leképzés, $X_{out} : O_n \to M_n$
  - π – tüzelés prioritása, $\pi \in N$

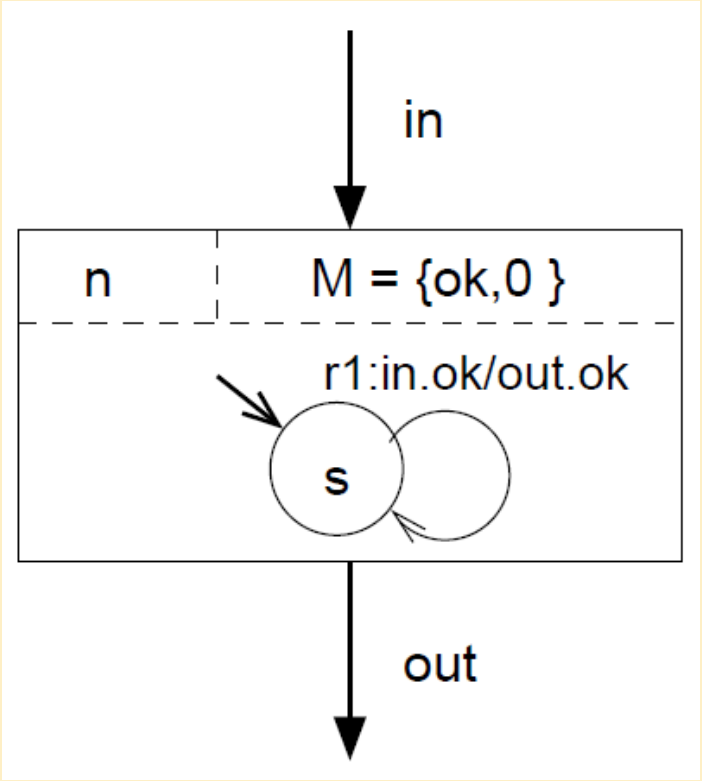
Adatfolyam hálózat jellemzése

- Adatfolyam háló állapota
  - Adatfolyam csomópontok állapota + adatfolyam csatornák állapota
- Adatfolyam csomópontok állapota
  - Az állapot a csomópont működését leíró állapotgép aktuális állapota
  - Állapotváltás: tüzelési szabályok „elsütése” során
- Adatfolyam csatornák állapota
  - Meghatározza a benne levő tokenek száma + színe + sorrendje
  - Kapacitása lehet végtelen vagy véges
    - Végtelen kapacitás $\Rightarrow$ mindig mehet bele token $\Rightarrow$ teljesen „aszinkron”
    - Véges kapacitás: a benne elhelyezett tokenek száma nem lehet több
      - Minden csatorna 1 kapacitású $\Rightarrow$ teljesen „szinkron”
  - Állapotváltás: tüzelési szabályok által megváltoztatott tokeneloszlás

Egy példa



- Egy token kapacitású csatornák
- Hálózat:
  - DFN = ({n}, {in, out},
  - {(s,0,0), (s,ok,0), (s,0,ok), (s,ok,ok)})
- Csomópontok:
  - n = ({in}, {out}, {s}, s, {ok,0}, {r1})
- Tüzelések:
  - r1=<s; in=ok; s; out=ok; 0>

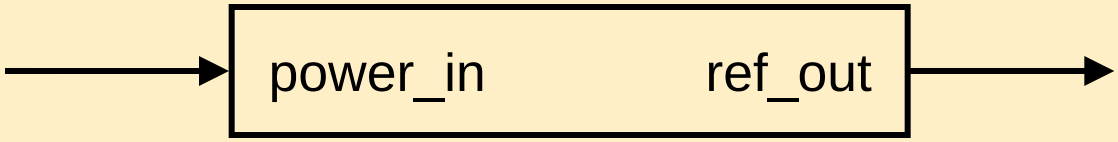


A módszer előnyei

Tulajdonság	**Alkalmas**
Grafikus, moduláris, kompakt, hierarchikus	Egyszerűen áttekinthető modell
Fekete és átlátszó doboz modell	Modellezés korai fázisban
Finomítási szabályok	Többszintű modellezés
Információáramlás direkt leírása	Hibaterjedés modellezése
Elosztott modell mind finom, mind durva pontossággal	Aszinkron, konkurens események
Adatvezérelt működés	Eseményvezérelt real-time rendszerek
Hívási átlátszóság, atomi tulajdonság, információrejtés	Hibatűrő alkalmazások
Matematikai formalizmus	Formális módszerek
Transzformáció: TTPN, PA	Validáció, időbeli analízis

Másik példa: referenciajel-generátor

(Analóg, de ez nem baj → kvalitatív leírás!)



Alapvető működés:

r0 = <s0; power_in=OK; s0; ref_out=OK>

Referenciajel-generátor modellje

Hibamodell:

OK – névleges érték

FTY – névlegestől eltérő érték

UNC – bizonytalan érték

Kiterjesztett működés (normál + hibás + bizonytalanság):

r0 = <s0; power_in=OK; s0; ref_out=OK>

r1 = <s0; power_in=FTY; s0; ref_out=UNC>

r2 = <s1; power_in=OK; s1; ref_out=FTY>

r3 = <s1; power_in=FTY; s1; ref_out=FTY>

(meghibásodás folyamata nincs modellezve)

Adatfolyam háló modellek szisztematikus bővítése: modellfinomítás

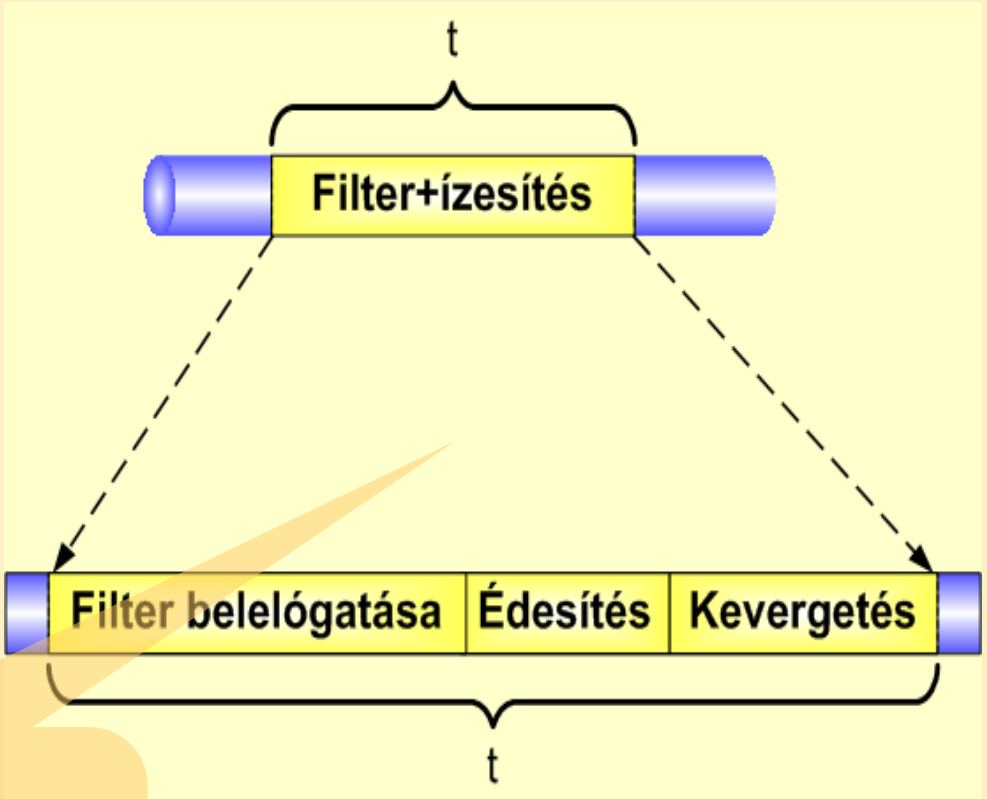
Modellfinomítás adatfolyam hálókra

Modellfinomítás:

- Többszintű modellezés
  - finomítás esetén „top-down” megközelítés
  - magasszintű, nem részletes, nemdeterminisztikus modell
  - finomítás során nemdeterminizmus korlátozása
- Szintek közötti átjárás (formalizált szabályok)
  - ellenőrzött modellváltoztatások
  - megfelelés halmazfinomítási értelemben
- Állapot és viselkedési konzisztencia megőrzése

Hierarchikus modellfinomítás

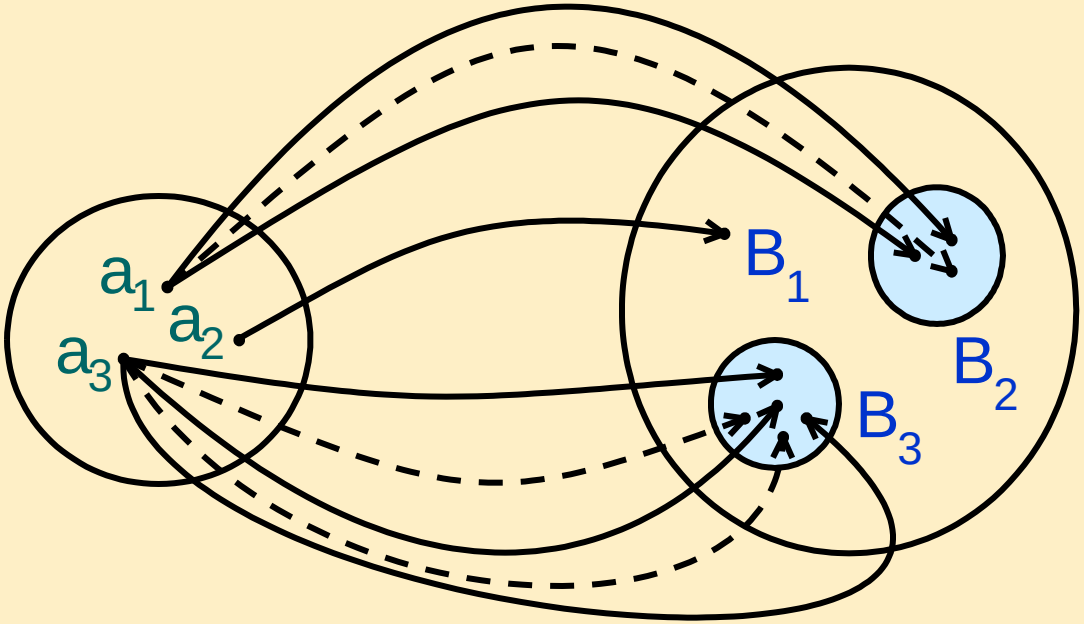
- Elemi eseményeket több részeseményre bontunk fel
- Az új események összideje = a régi esemény ideje
- Azonos bemenetekre azonos „lehetséges” válaszok



Kibontás
„egy az egyben”
behelyettesíthető
KOMPOZÍCIONALITÁS

Halmazfinomítás

Diszjunkt részhalmazok hozzárendelése elemekhez



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

Modellfinomítás adatfolyam hálókra

- Fekete doboz nézet
  - Csak a környezettel való kapcsolat jelenik meg
    - szintaktikus interfész: ki- és bemeneti csatornák, üzenettípusok
    - szemantikus interfész: ki- és bemenő üzenetek kapcsolata
- Átlátszó doboz nézet
  - Kommunikáció finomítás
    - komponens szintaktikus interfészének megváltoztatása
    - ki- és bemeneti csatornák és az üzenettípusok száma változik
  - Állapottér finomítás
    - állapotátmeneti reláció elemeinek száma változik
  - Eloszlás finomítás
    - dekompozíció, részkomponensekre bontás, struktúra változik

Modellfinomítás adatfolyam hálókra

Fekete- és átlátszó dobozokra megfogalmazott elvek általánosítása adatfolyam hálókra:

- Értelmezési tartomány finomítás
  a. Token halmaz finomítás
  b. Állapothalmaz finomítás
- Struktúra finomítás
  – Csomópont helyettesítése adatfolyam alhálóval

Az értelmezési tartomány finomítása
Állapottér bővítése, specifikálása

Értelmezési tartomány finomítás

Két típusa van:

1. Tokenek halmazának finomítása
   - M'_n finomítottja M_n-nek
2. Állapotok halmazának finomítása
   - S'_n finomítottja S_n-nek

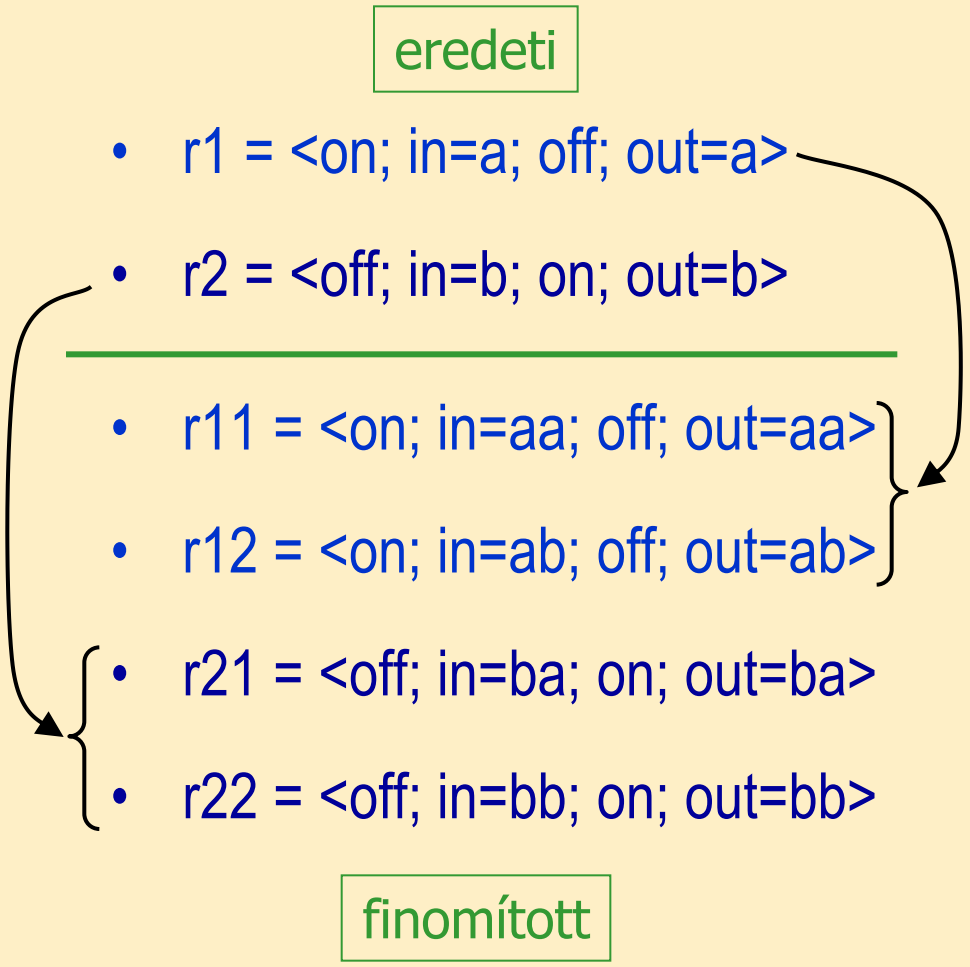
vagy-vagy, egyszerre csak az egyik

$\Rightarrow$Tüzelési szabályok megfelelő változtatása

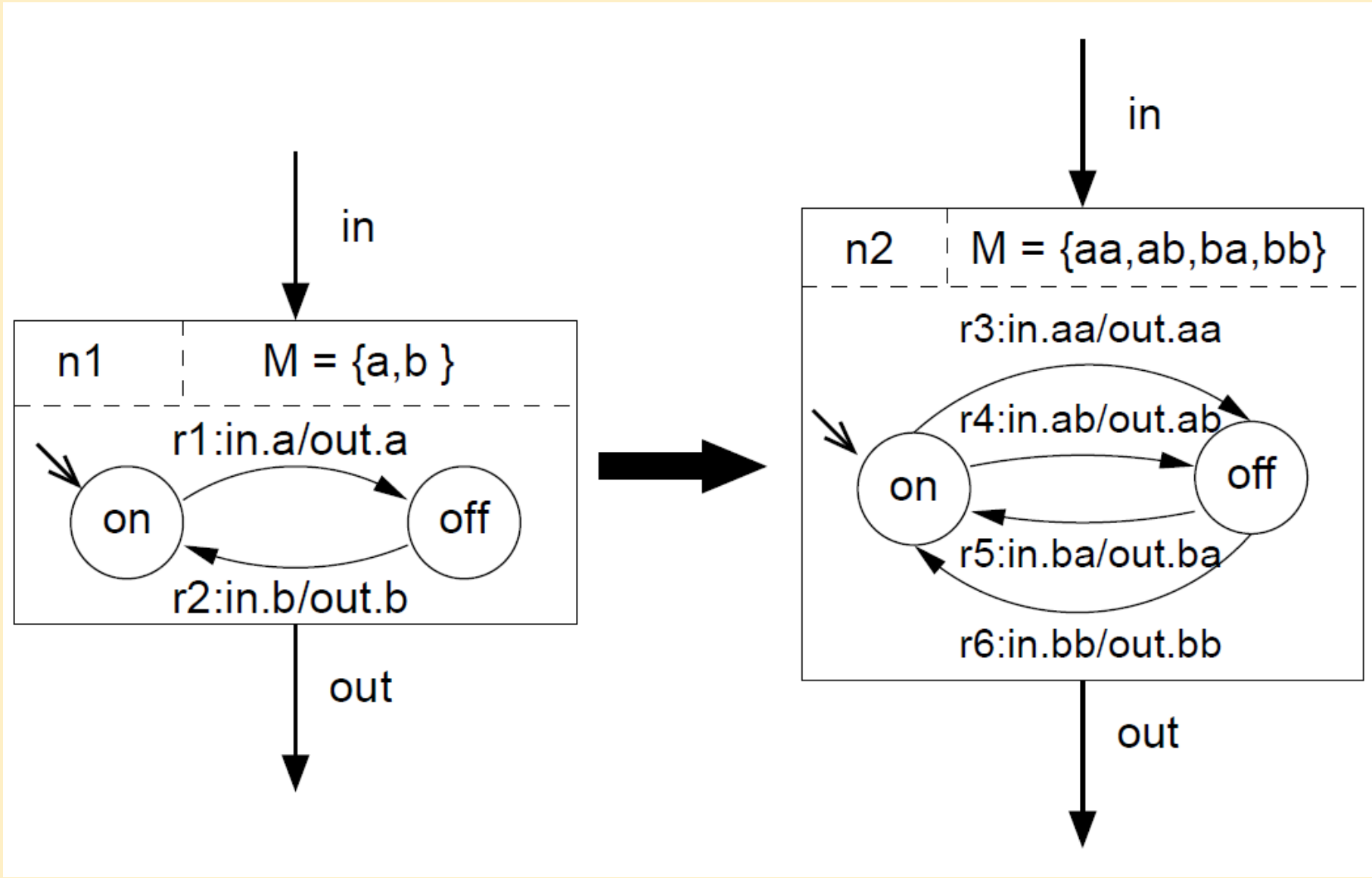
- Ki- és bemeneti csatornák változatlanok
- Struktúra változatlan

Tokenek halmazának finomítása: példa

	n_1	$n_2 = \Re(n_1)$
Állapotok	on off	{on} {off}
Tokenek	a b	{aa, ab} {ba, bb}
Tüzelési szabályok	r1 r2	{r11, r12} {r21, r22}



Értelmezési tartomány finomítás: tokenek halmaza



Állapotok halmazának finomítása: példa

	n_1	$n_2 = \Re(n_1)$
Állapotok	good	{good}
	fty	{hot, cold}
Tokenek	a	{a}
	b	{b}
	c	{c}
Tüzelési szabályok	r1	{r11}
	r2	{r21, r22}
	r3	{r31, r32}

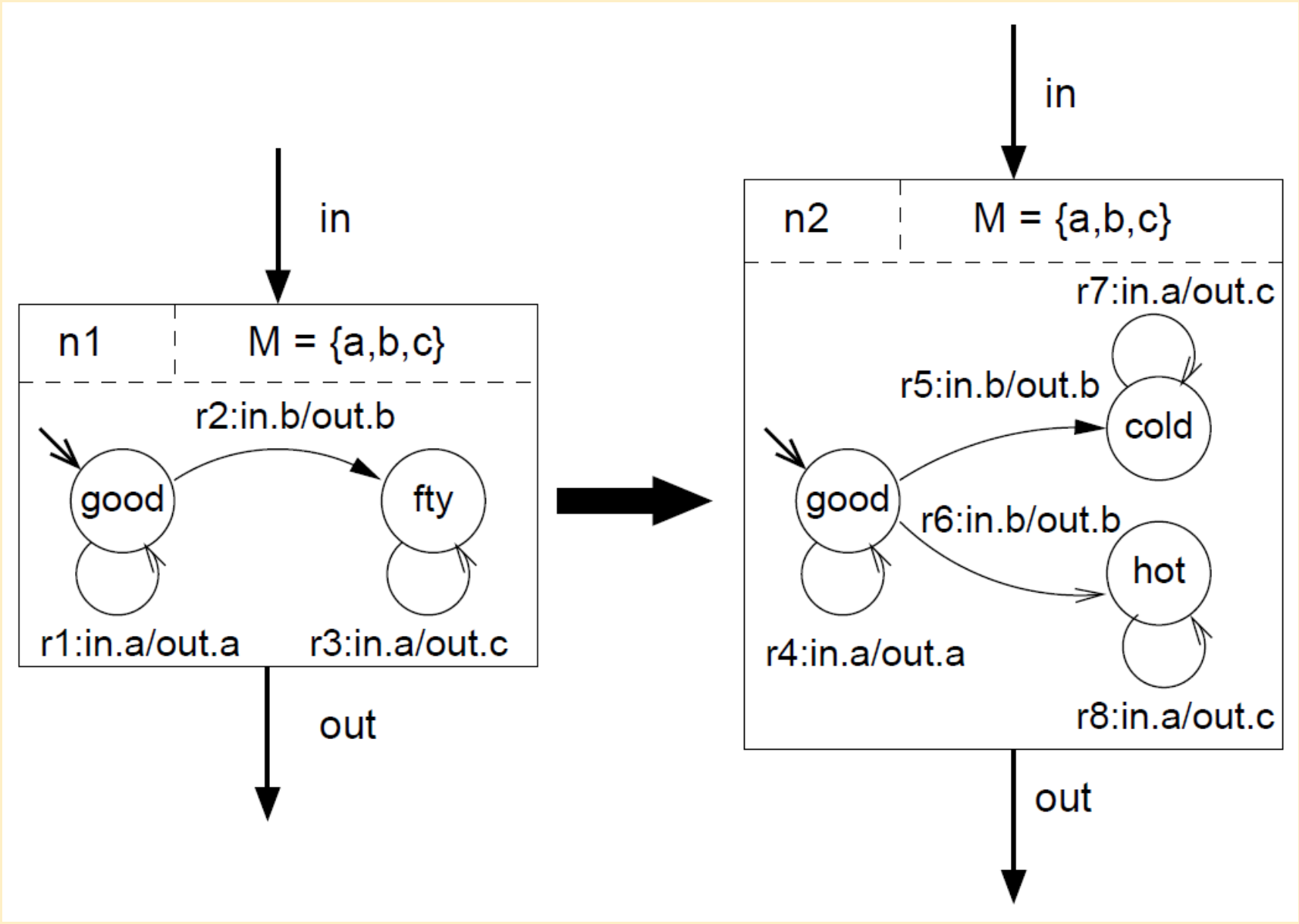
eredeti

- r1 = <good; in=a; good; out=a>
- r2 = <good; in=b; fty; out=b>
- r3 = <fty; in=a; fty; out=c>

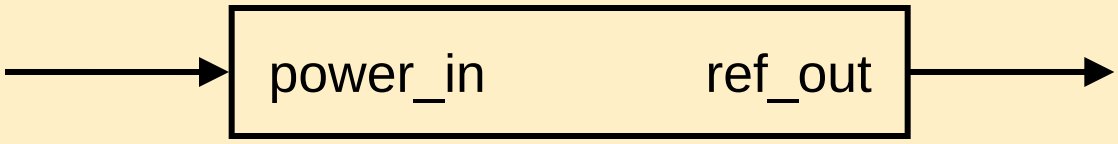
- r11 = <good; in=a; good; out=a>
- r21 = <good; in=b; cold; out=b>
- r22 = <good; in=b; hot; out=b>
- r31 = <cold; in=a; cold; out=c>
- r32 = <hot; in=a; hot; out=c>

finomított

Értelmezési tartomány finomítás: állapotok halmaza



Példa: referenciajel-generátor



- Hibamodell:
  - OK – névleges feszültség
  - FTY – névlegestől eltérő feszültség
- Működés:
  - r0 = <s0; power_in=OK; s0; ref_out=OK>
  - r1 = <s0; power_in=FTY; s0; ref_out=OK>
  - r2 = <s0; power_in=FTY; s0; ref_out=FTY>

  } (eddig UNC)
  - r3 = <s1; power_in=OK; s1; ref_out=FTY>
  - r4 = <s1; power_in=FTY; s1; ref_out=FTY>

Példa: referenciajel-generátor (finomított működés)

1. Állapothalmaz finomítás: s1 → s1a, s1b

   r0=<s0; power_in=OK; s0; ref_out=OK>

   r1=<s0; power_in=FTY; s0; ref_out=OK>

   r2=<s0; power_in=FTY; s0; ref_out=FTY>

   r31=<s1a; power_in=OK; s1a; ref_out=FTY>

   r32=<s1b; power_in=OK; s1b; ref_out=FTY>

   r41=<s1a; power_in=FTY; s1a; ref_out=FTY>

   r42=<s1b; power_in=FTY; s1b; ref_out=FTY>

2. Tokenfinomítás: FTY → LOW, HIGH (s0 állapot)
3. Tokenfinomítás: FTY → LOW, HIGH (s1 állapot)

Példa: referenciajel-generátor (finomított működés)

1. Állapothalmaz finomítás: s1 → s1a, s1b
2. Tokenfinomítás: FTY → LOW, HIGH (s0 állapot)

   r0=<s0; power_in=OK; s0; ref_out=OK>

   r11=<s0; power_in=HIGH; s0; ref_out=OK>

   r21=<s0; power_in=LOW; s0; ref_out=LOW>

   r31=<s1a; power_in=OK; s1a; ref_out=FTY>

   r32=<s1b; power_in=OK; s1b; ref_out=FTY>

   r41=<s1a; power_in=FTY; s1a; ref_out=FTY>

   r42=<s1b; power_in=FTY; s1b; ref_out=FTY>
3. Tokenfinomítás: FTY → LOW, HIGH (s1 állapot)

Példa: referenciajel-generátor (finomított működés)

1. Állapothalmaz finomítás: s1 → s1a, s1b
2. Tokenfinomítás: FTY → LOW, HIGH (s0 állapot)
3. Tokenfinomítás: FTY → LOW, HIGH (s1 állapot)

r0=<s0; power_in=OK; s0; ref_out=OK>

r11=<s0; power_in=LOW; s0; ref_out=LOW>

r21=<s0; power_in=HIGH; s0; ref_out=OK>

r311=<s1a; power_in=OK; s1a; ref_out=LOW>

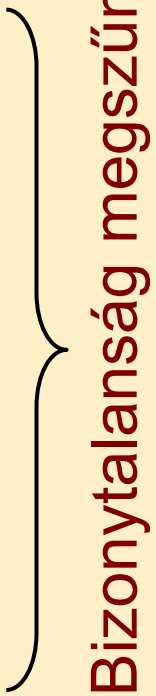
r321=<s1b; power_in=OK; s1b; ref_out=HIGH>

r411=<s1a; power_in=LOW; s1a; ref_out=LOW>

r412=<s1a; power_in=HIGH; s1a; ref_out=LOW>

r421=<s1b; power_in=LOW; s1b; ref_out=LOW>

r422=<s1b; power_in=HIGH; s1b; ref_out=HIGH>

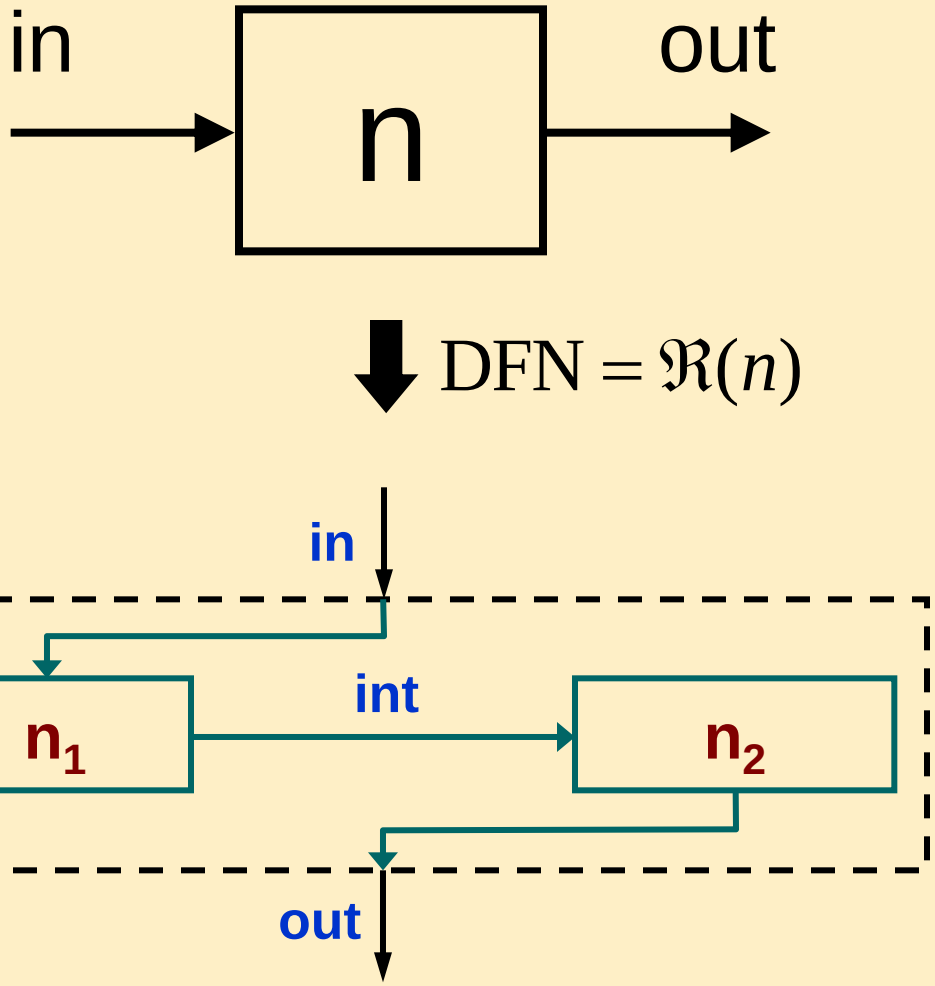


A struktúra finomítása
A modellelemek (hierarchikus) bővítése

Struktúra finomítás

- Struktúra finomítás:
  - Struktúra módosítás
    - környezethez kapcsolódó csatornák változatlanok
    - belső csomópontok és csatornák keletkeznek
  - Állapotmegfeleltetés: csomópont $\leftrightarrow$ részháló
  - Tokenek halmaza változatlan
  - Tüzelésekből *tüzelési szekvenciák* megfelelő kialakítása
    - bemeneti csatornákon érkező azonos tokenszekvenciákra ugyanazon lehetséges kimeneti tokenszekvenciákat állítja elő

Struktúra finomítás: példa



Struktúra finomítás: példa

	n_1	$n_2 = \Re(n_1)$
Állapotok	good fty	{{good, good, X}, {good, fty, X}} {{fty, good, X}, {fty, fty, X}}
Tokenek	a b	{a} {b}
Tüzelési szabályok	r1 r2	{⟨$r_{n1}1$; $r_{n2}1$⟩; ⟨$r_{n1}1$; $r_{n2}3$⟩} {⟨$r_{n1}2$; $r_{n2}2$⟩; ⟨$r_{n1}2$; $r_{n2}4$⟩}

- r_n1 = <good; in=a; good; out=a>
- r_n2 = <good; in=b; fty; out=b>

n1

- $r_{n1}1$ = <good; in=a; good; int=a>
- $r_{n1}2$ = <good; in=b; fty; int=b>

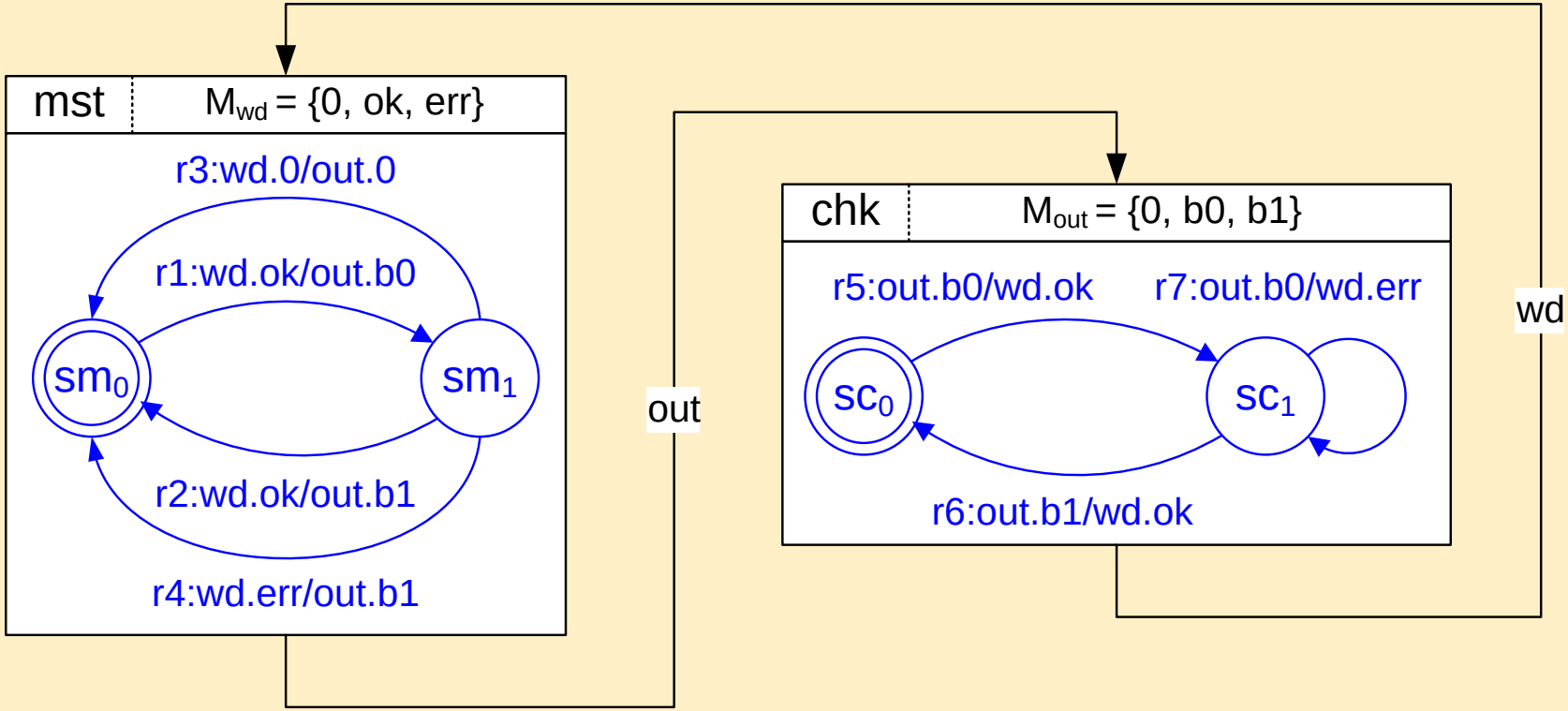
n2

- $r_{n2}1$ = <good; in=a; good; out=a>
- $r_{n2}2$ = <good; in=b; good; out=b>
- $r_{n2}3$ = <fty; in=a; fty; out=a>
- $r_{n2}4$ = <fty; in=b; fty; out=b>

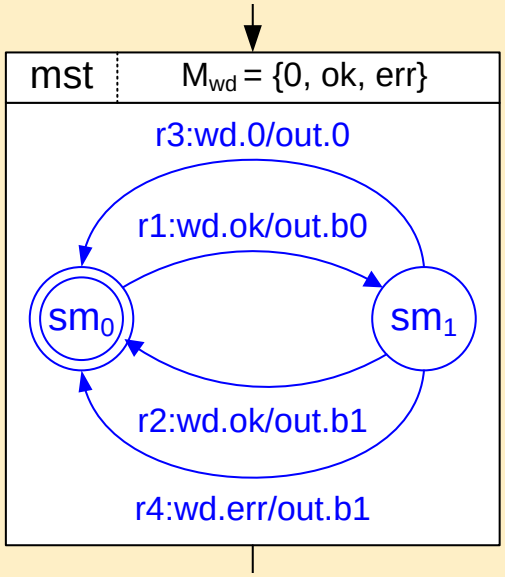
Adatfolyam hálók széthajtogatása

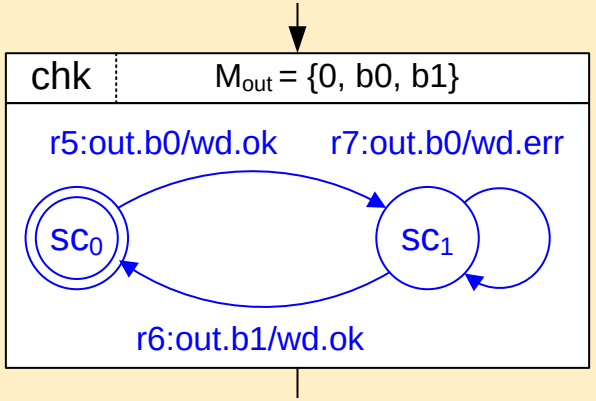
Példa adatfolyam háló

- master: mst = ({wd}, {out}, {sm0, sm1}, sm0,
  {r1, r2, r3, r4}, {ok, err, b0, b1}),
- checker: chk = ({out}, {wd}, {sc0, sc1}, sc0,
  {r5, r6, r7}, {b0, b1, ok, err}).



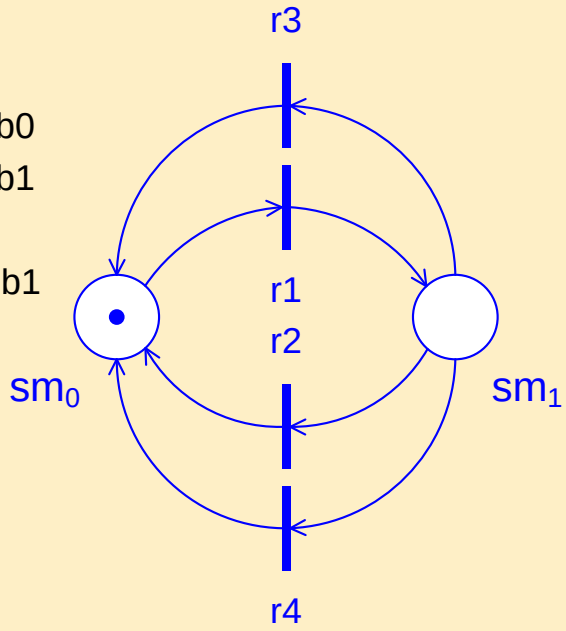
Adatfolyam csomópontok „széthajtogatása”





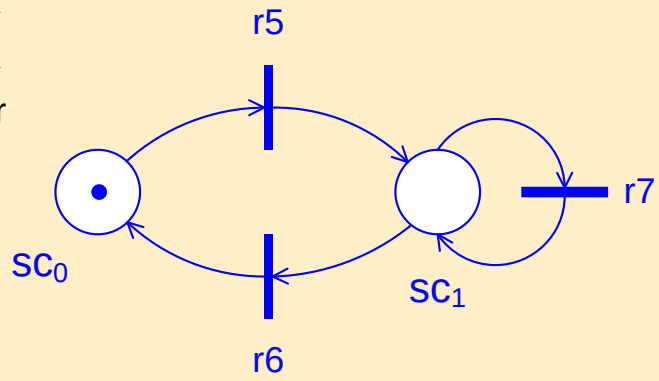
mst

r1:wd.ok/out.b0
r2:wd.ok/out.b1
r3:wd.0/out.0
r4:wd.err/out.b1

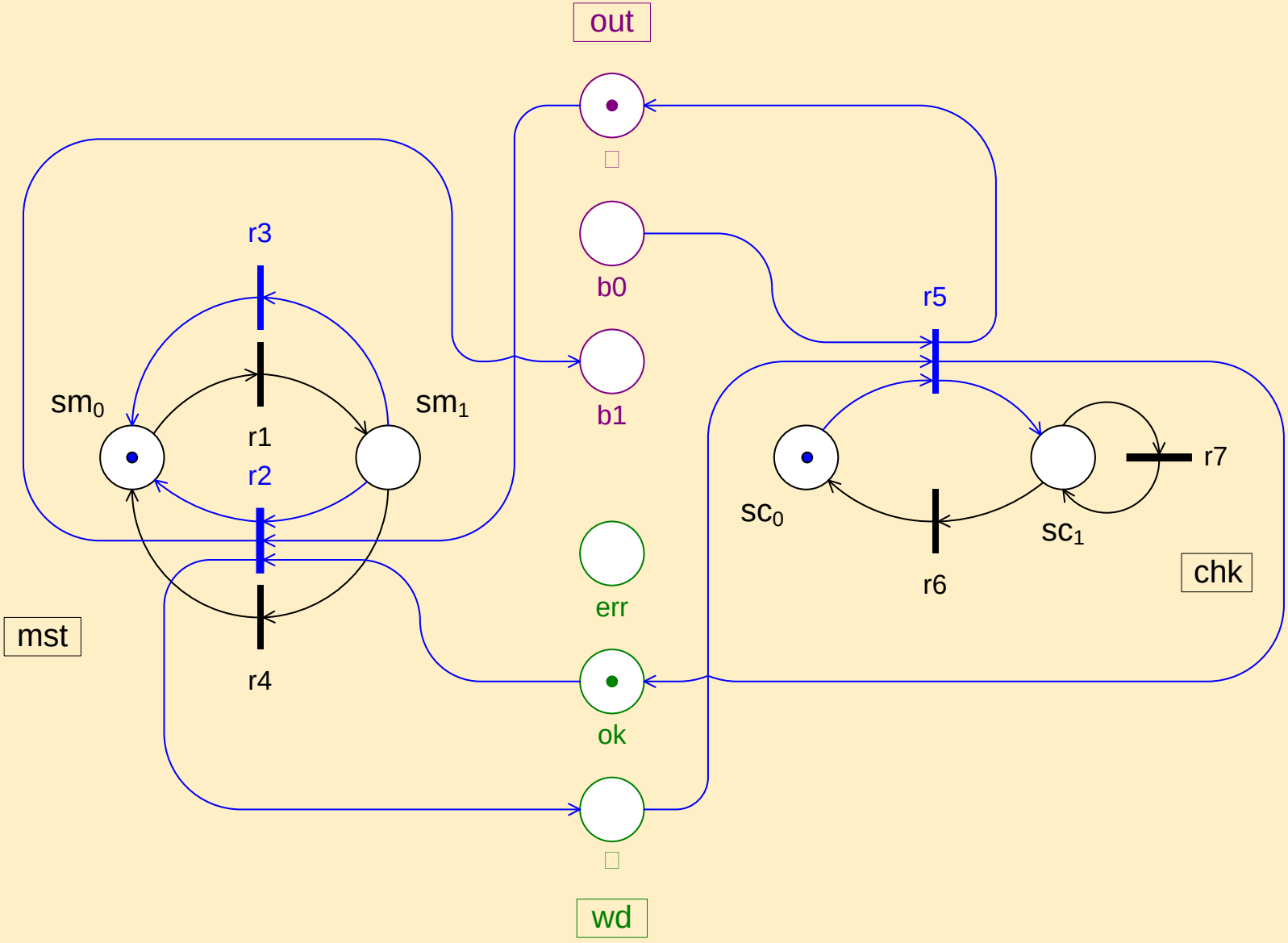


chk

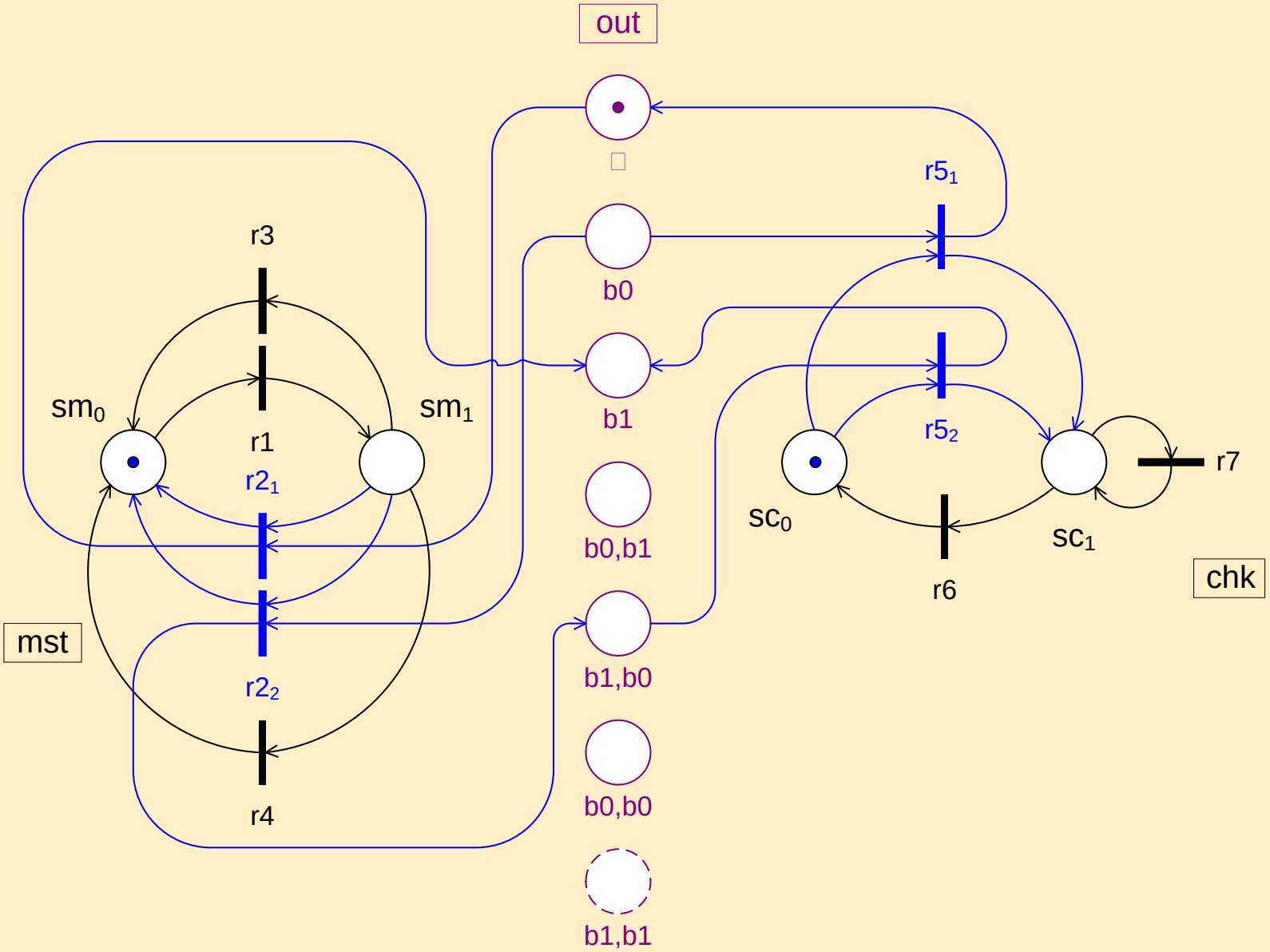
r5:out.b0/wd.ok
r6:out.b1/wd.ok
r7:out.b0/wd.err



A csatornák modellje és a tüzelési szabályok



Széthajtogatás 2 kapacitású csatorna esetén



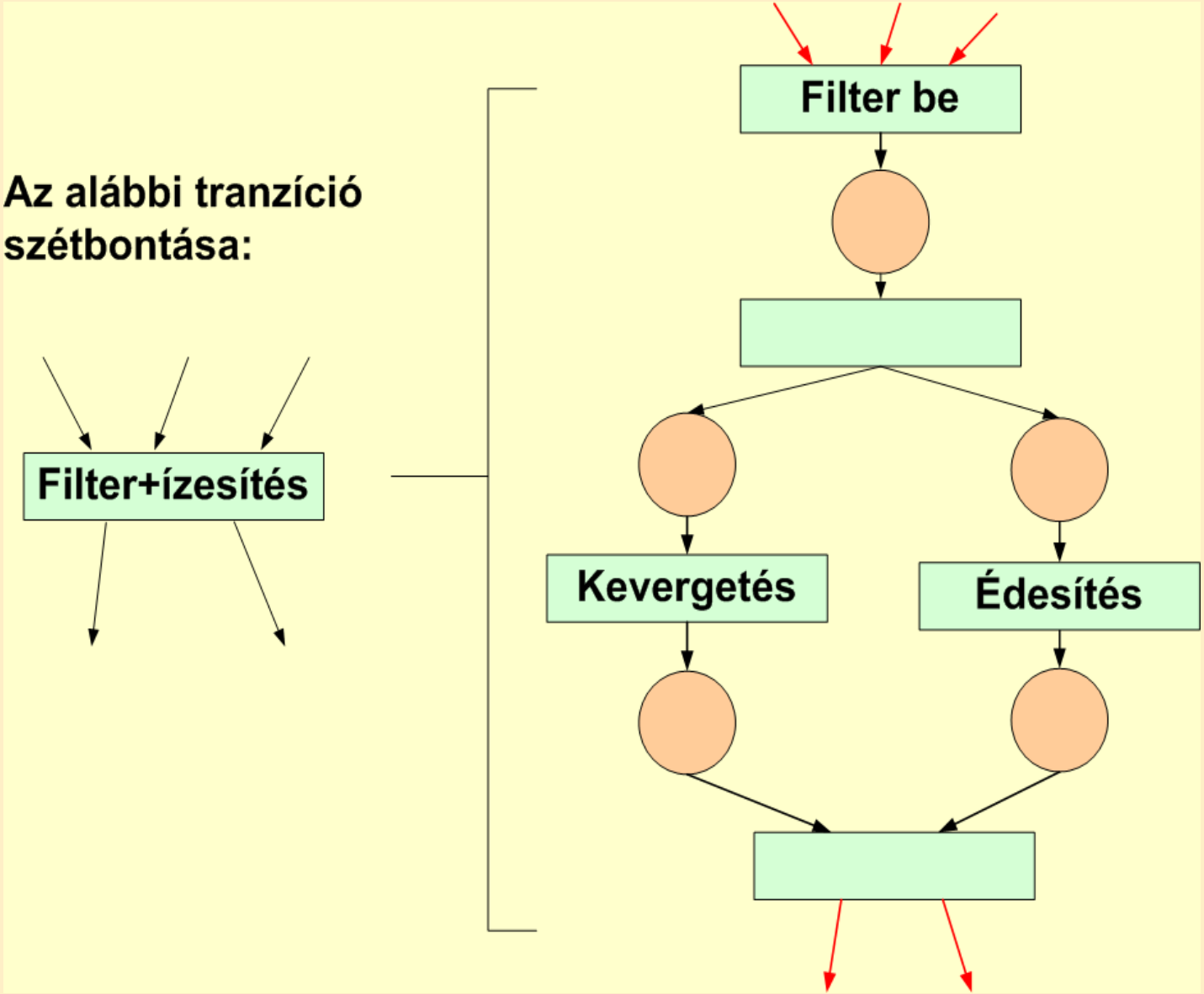
Hierarchikus Petri hálók
Modellfinomítás Petri hálók esetén

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:



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

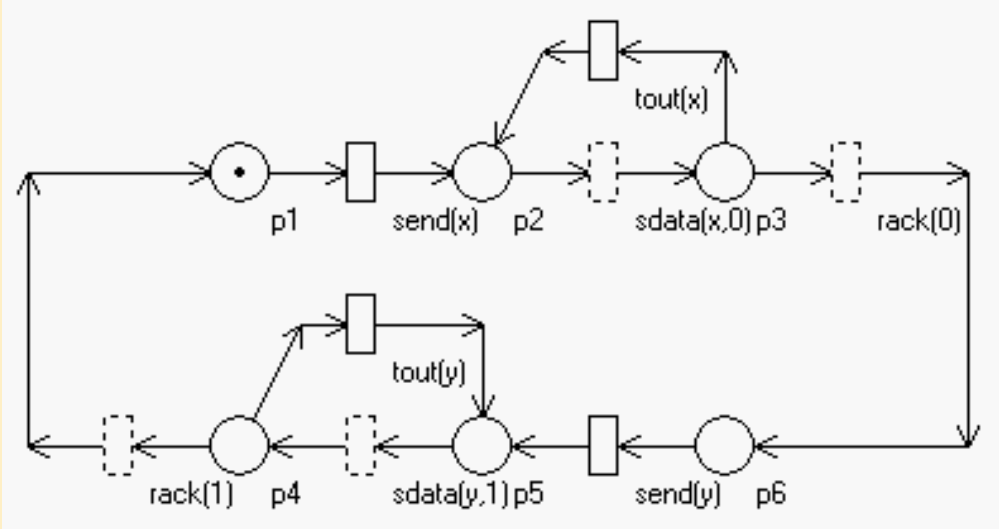
- A hierarchikus felépítés előnyei
  - a részfeladatok megoldására koncentrál
  - egyszerűsíti a leírást az egyes szinteken
  - áttekinthető a részrendszerek kapcsolata
- A hierarchia eszközei
  - alhálók, „fő” háló
  - interakciós pontok: helyek és átmenetek
  - subnet-to-subnet élek

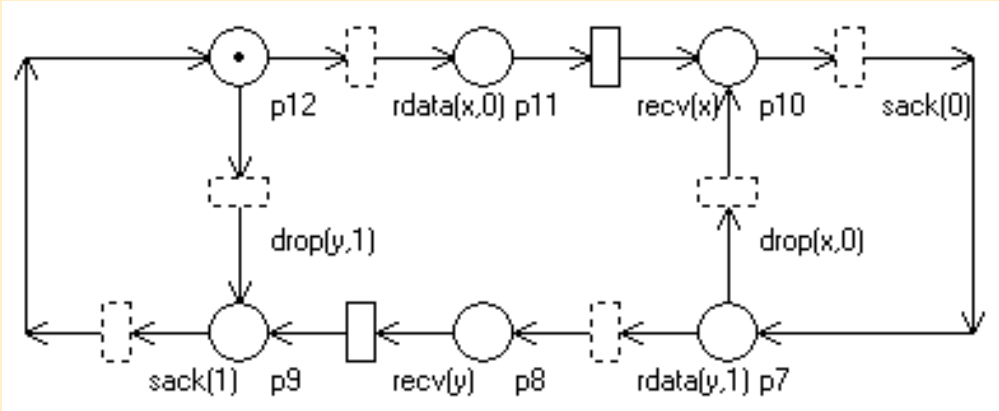
Hierarchikus felépítés eszközei

- 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

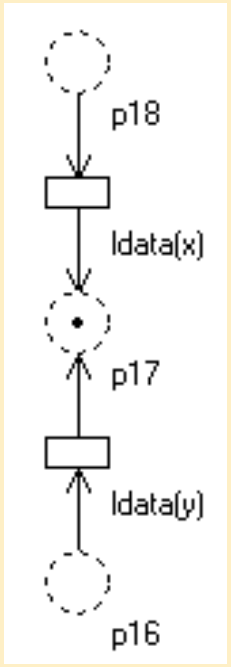
- Nem strukturált eszköz: alháló
  - Bármit helyettesíthet, de emiatt kevésbé áttekinthető
  - Nem támogatja a szisztematikus modellbővítést, modellfinomítást
  - Támogatja a modellelemek újrafelhasználását

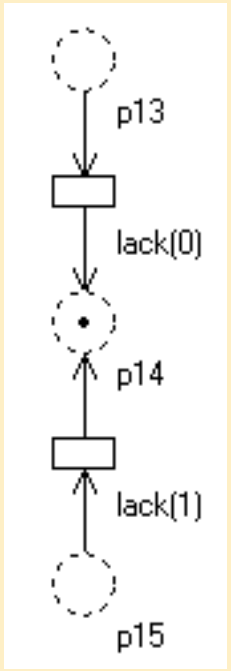


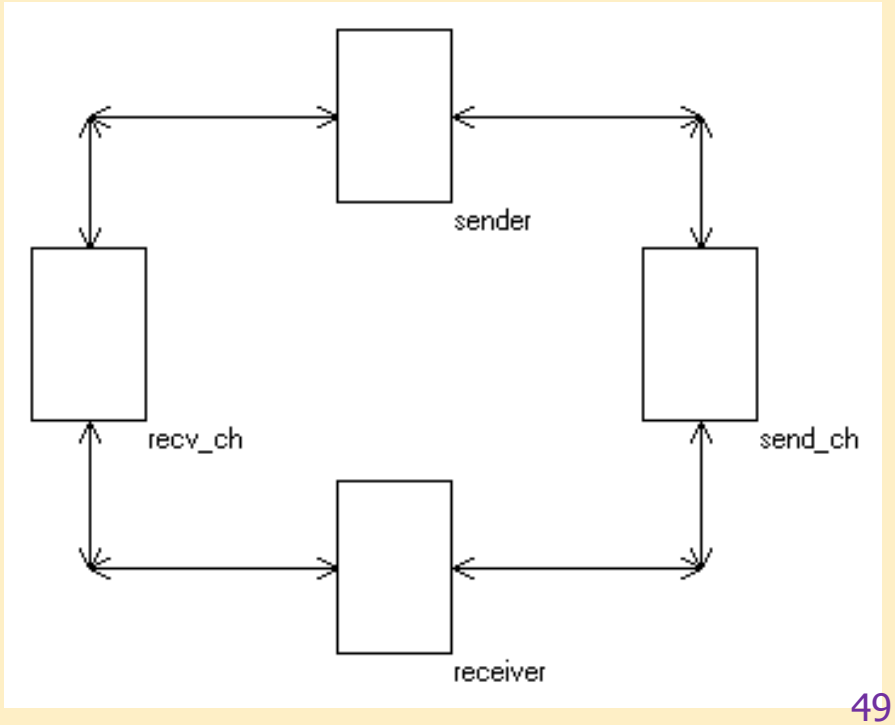


DNAnet: Az alrendszerek egyesítése

- 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

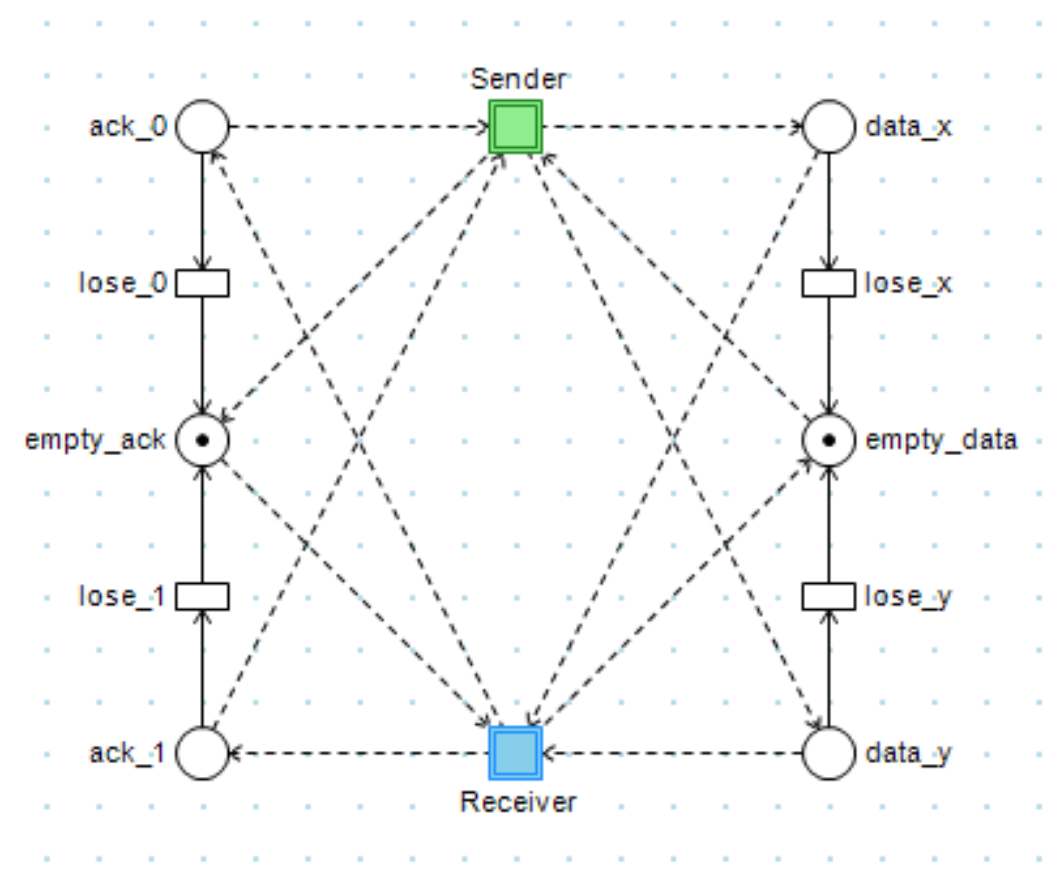




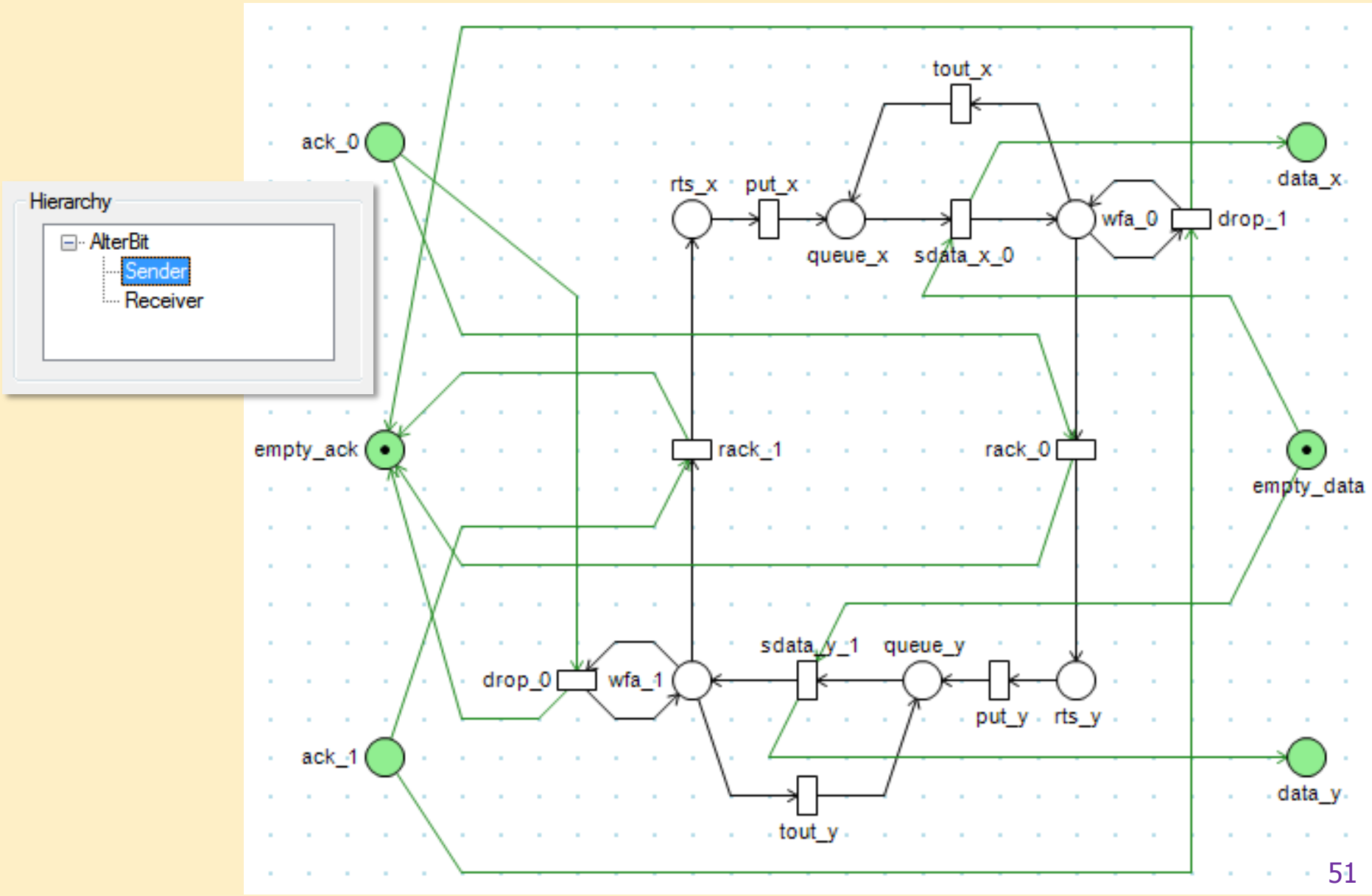


PetriDotNet, Snoopy: Alrendszerek egyesítése

- 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



Hierarchikus színezett Petri hálók

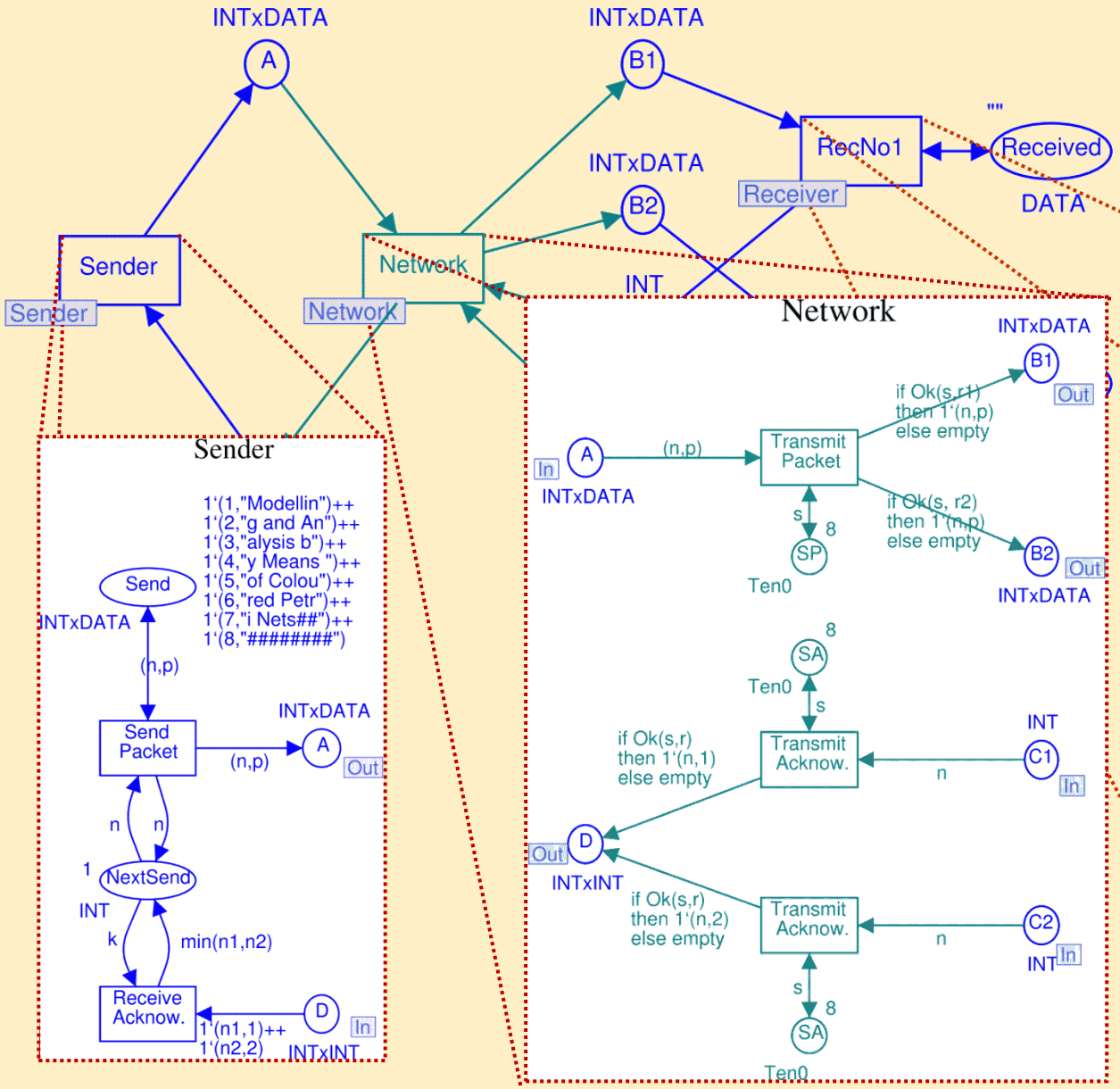
Hierarchikus színezett Petri hálók

- Alhálók integrálása egyetlen összetett CPN hálóvá hierarchikus rendszerben
  - lapok: színezett Petri hálók
    - lapszám, lapnév: alternatívák a lapra való hivatkozáshoz
    - a lapok példányosíthatók (a hierarchia bármelyik szintján)
  - hierachikus jelölések
    - al-lap $\leftrightarrow$ felettes lap
    - lap példányok: a jelölés (tokeneloszlás) minden példányra egyedi
      - fő (prime) lap: legfelső szint, másodlagos lap példányok (al-lapok)
      - azonosítás: lap példány azonosító szám
      - laphierarchia gráf

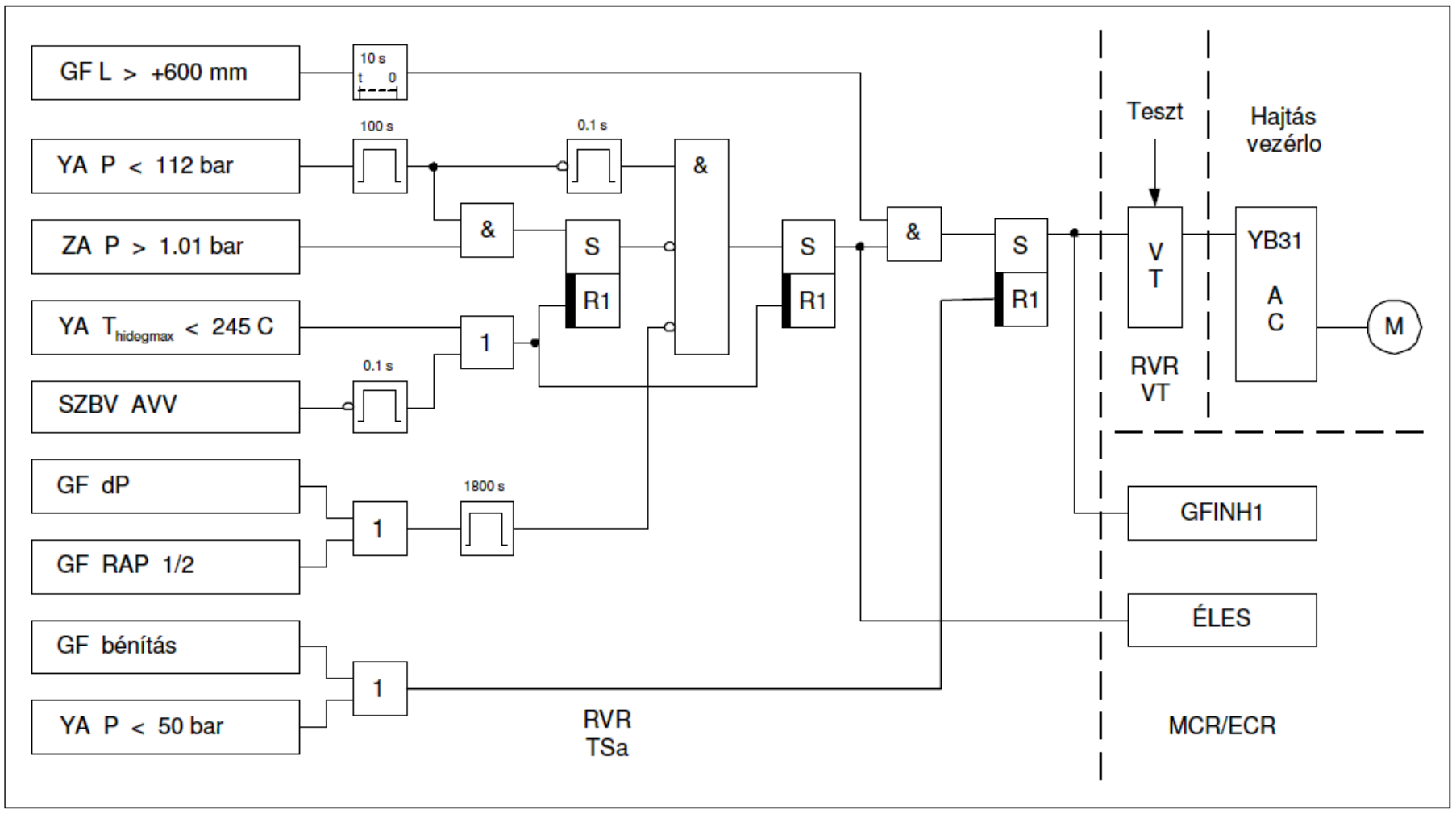
Hierarchikus felépítés eszközei

1. Helyettesítő tranzíciók
   - port kiosztás (interfészek): al-lap → felettes lap
     1. „port” hely (az al-lapon) → ezekkel kapcsolódik az alháló
        port típus: bemenet, kimenet, I/O (kétirányú), általánosított
     2. „socket” hely (a főlapon) → az alhálók beillesztési pontjai
2. Fúziós helyek
   - Azonos névvel, de több példányban létrehozott helyek, amik ugyanazon helyet jelölik a háló több pontján
   - A tokenek egyszerre kerülnek be / távolítódnak el egy adott fúziós helyhez tartozó helyhalmazba/-ból

Példa: az egyszerű protokoll hierarchikus változata

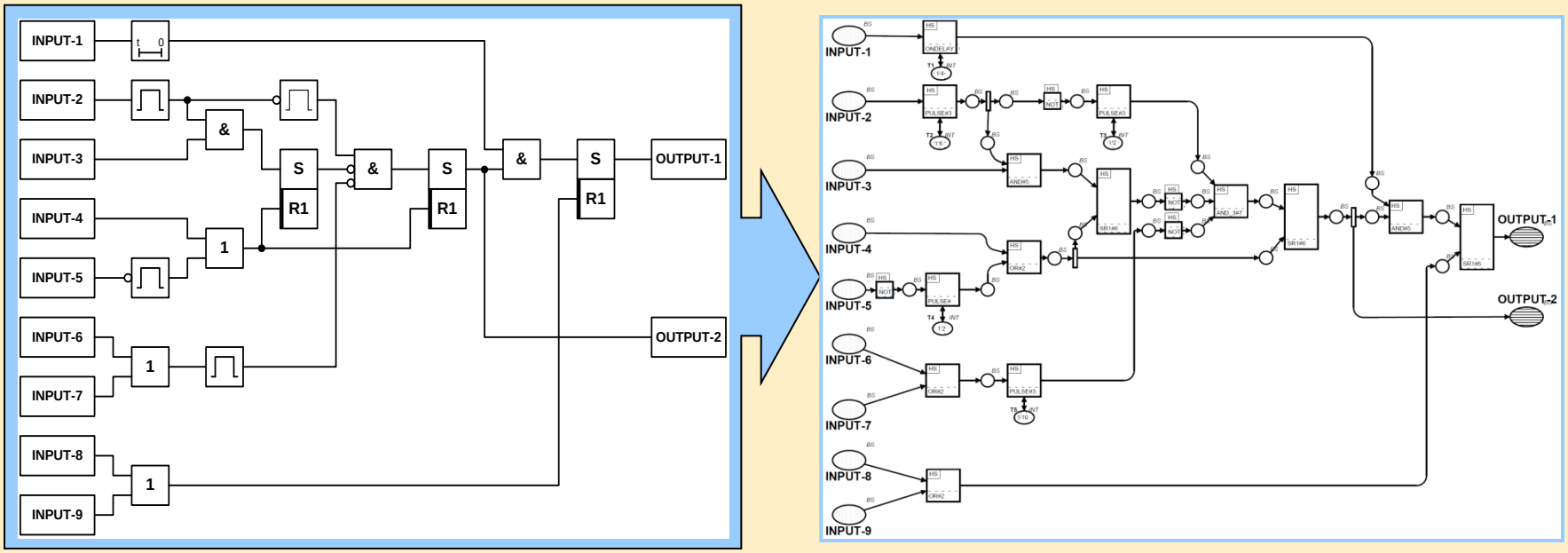


Másik példa: PRISE élesítő biztonsági logika



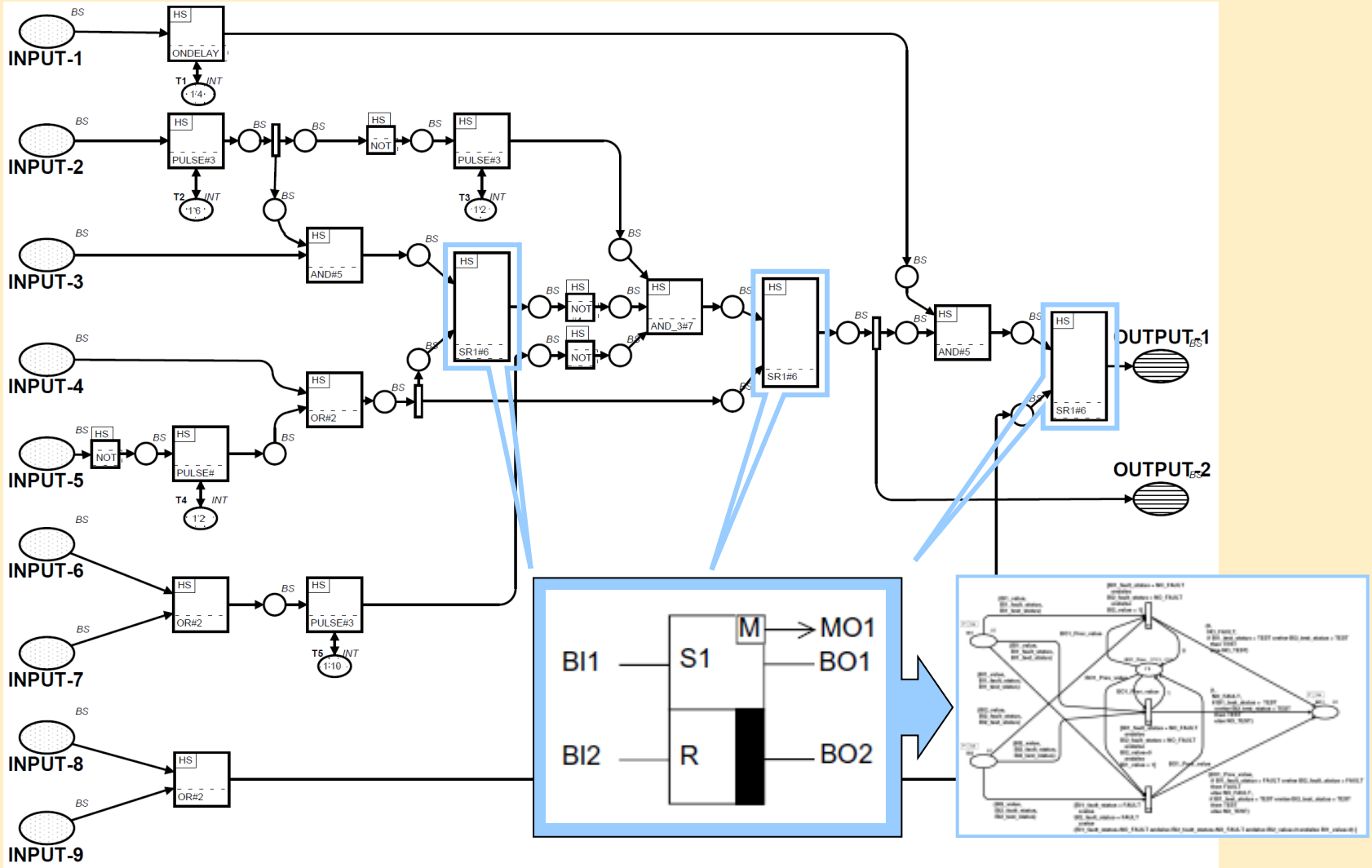
A PRISE logika magas szintű modellje

Design/CPN



- A CPN formalizmus jól illeszkedik az FBD alapú leíráshoz
  - Mindkettő az adatfolyamot írja le a modellezett rendszerben
  - Az eredeti FBD és az előállított CPN struktúrája nagyon hasonló
  - Funkcionális blokkon modellezhetők CPN alhálókkal → könyvtár

Funkcionális blokkok, mint alhálók



Elkészíthető a funkcionális blokkok CPN alháló könyvtára

A statikus RS flip-flop CPN alháló modellje