Adatfolyam hálók

Dr. Bartha Tamás, Dr. Pataricza András fóliái

Adatfolyam modellezés

Nem determinisztikus DFN formalizmus

- [Jonsson, Cannata]

- Struktúra
  - Adatfolyam gráf (DFG)
    - csomópontok
    - irányított élek (FIFO csatornák)
- Viselkedés
  - Tüzelési szabályok: <s0; in=c0; s1; out=2; π>
- Adatok
  - Tokenek

Adatfolyam modellezés

Nem determinisztikus DFN formalizmus

- [Jonsson, Cannata]

- Struktúra
  - Adatfolyam gráf (DFG)
    - csomópontok
    - irányított élek (FIFO csatornák)
- Viselkedés
  - Tüzelési szabályok: **<s0; in=c0; s1; out=c2; π>**
- Adatok
  - Tokenek

Adatfolyam modellezés

Nem determinisztikus DFN formalizmus

- [Jonsson, Cannata]

- Struktúra
  - Adatfolyam gráf (DFG)
    - cs
    - irá (FIFO csatornák)
- Viselkedés
  - Tüzelési szabályok: **<s0; in=c0; s1; out=c2; π>**
- Adatok
  - Tokenek

Kiinduló állapot

Adatfolyam modellezés

Nem determinisztikus DFN formalizmus

- [Jonsson, Cannata]

- Struktúra
  - Adatfolyam gráf (DFG)
    - cs
    - irá (FIFO csatornák)
- Viselkedés
  - Tüzelési szabályok: **<s0; in=c0; s1; out=c2; π>**
- Adato
  - Tok

Kiinduló állapot

Bemeneti csatorna

Adatfolyam modellezés

Nem determinisztikus DFN formalizmus

- [Jonsson, Cannata]

- Struktúra
  - Adatfolyam gráf (
    - cs
    - irá (F
- Viselkedés
  - Tüzelési szabályok: **<s0; in=c0; s1; out=c2; π>**
- Adato
  - Tok

Kiinduló állapot

Bemeneti csatornáról elvett token

Bemeneti csatorna

Adatfolyam modellezés

Nem determinisztikus DFN formalizmus

- [Jonsson, Cannata]

- Struktúra
  - Adatfolyam gráf (
    - cs
    - irá (F
- Viselkedés
  - Tüzelési szabályok: **<s0; in=c0; s1; out=c2; π>**
- Adato
  - Tok

Kiinduló állapot

Bemeneti csatornáról elvett token

Bemeneti csatorna

Célállapot

Adatfolyam modellezés

Nem determinisztikus DFN formalizmus

- [Jonsson, Cannata]

- Struktúra
  - Adatfolyam gráf (
    - cs
    - irá (F
- Viselkedés
  - Tüzelési szabályok: **<s0; in=c0; s1; out=c2; π>**
- Adato
  - Tok

Kiinduló állapot

Bemeneti csatornáról elvett token

Kimeneti csatorna

Bemeneti csatorna

Célállapot

Adatfolyam modellezés

Nem determinisztikus DFN formalizmus

- [Jonsson, Cannata]

- Struktúra
  - Adatfolyam gráf (
    - cs
    - irá (F
- Viselkedés
  - Tüzelési szabályok: **<s0; in=c0; s1; out=c2; π>**
- Adato
  - Tok

Kiinduló állapot

Bemeneti csatornáról elvett token

Kimeneti csatorna

Bemeneti csatorna

Célállapot

Kimeneti csatornára kitett token

Adatfolyam modellezés

Nem determinisztikus DFN formalizmus

- [Jonsson, Cannata]

- Struktúra
  - Adatfolyam gráf (
    - cs
    - irá (F
- Viselkedés
  - Tüzelési szabályok: **<s0; in=c0; s1; out=c2; π>**
- Adato
  - Tok

Kiinduló állapot

Bemeneti csatornáról elvett token

Kimeneti csatorna

Prioritás

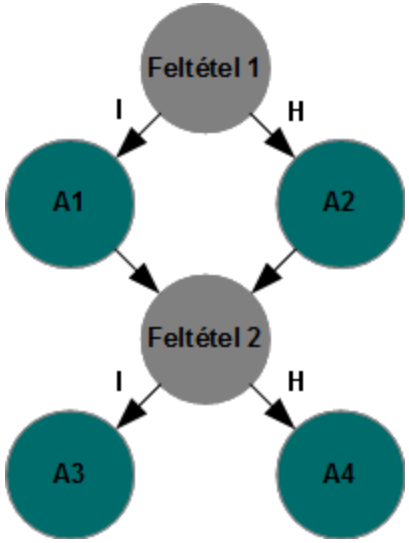
Bemeneti csatorna

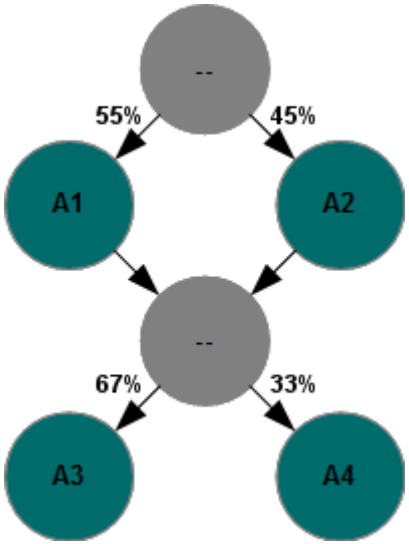
Célállapot

Kimeneti csatornára kitett token

Nem determinisztikus adatfolyam

- A rendszer determinisztikus:
  - Egy adott állapotban bekövetkező feltételek szerint hajt végre akciókat.
- A rendszer nem determinisztikus:
  - Az eddigi feltételek helyett az akciók végrehajtásának valószínűsége adott (randomizált modell).
  - A randomizált modell nem feltétlenül ,,ekvivalens'' a determinisztikus modellel.





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

Adatfolyam hálózat formális leírása

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

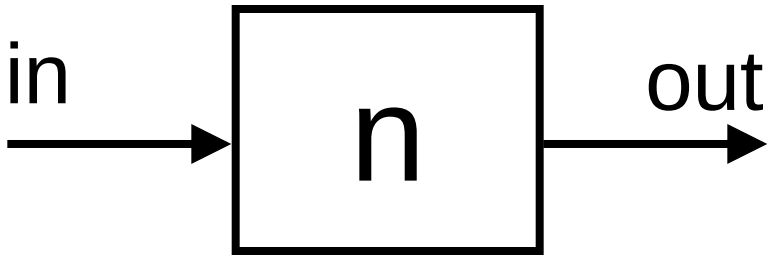
    } kapcsolat a külvilággal
    - IN: belső (csomópontok közötti) csatornák
  - *S* : állapotok halmaza
- **Adatfolyam csatorna**:
  - végtelen kapacitású FIFO csatorna,
  - egy bemeneti és egy kimeneti csomóponthoz kötve
  - állapota: $S_c = \times^{\infty} M_c$ tokenszekvencia

Adatfolyam csomópont formális leírása

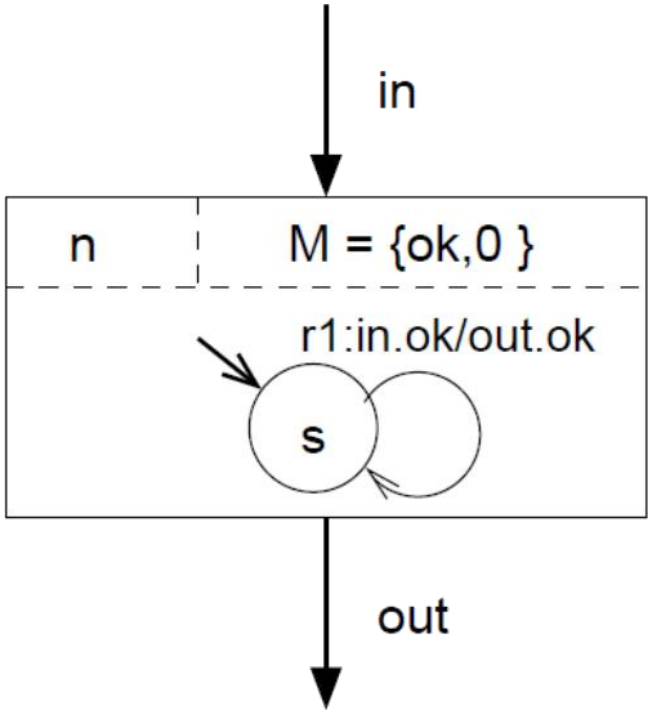
Adatfolyam csomópont: $n = (I_n, O_n, S_n, s_n^0, R_n, M_n)$, ahol

- 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, $X_{in} : I_n \rightarrow M_n$
  - X_{out} – kimenő leképzés, $X_{out} : O_n \rightarrow M_n$
  - π – tüzelés prioritása, $\pi \in N$

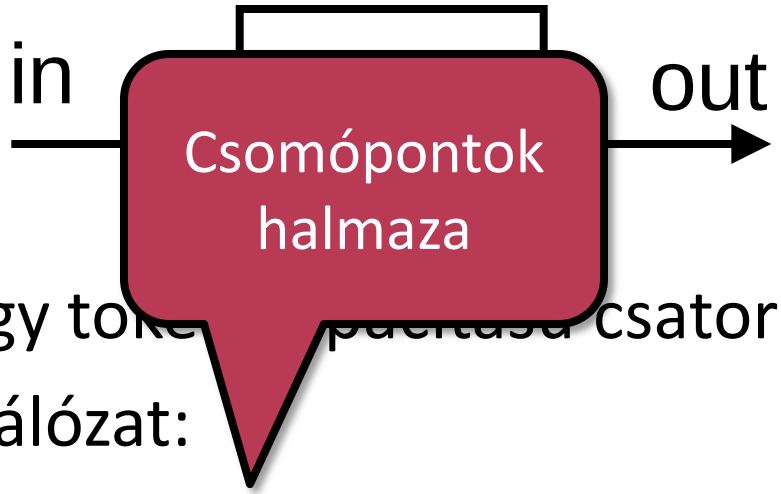
Egy példa



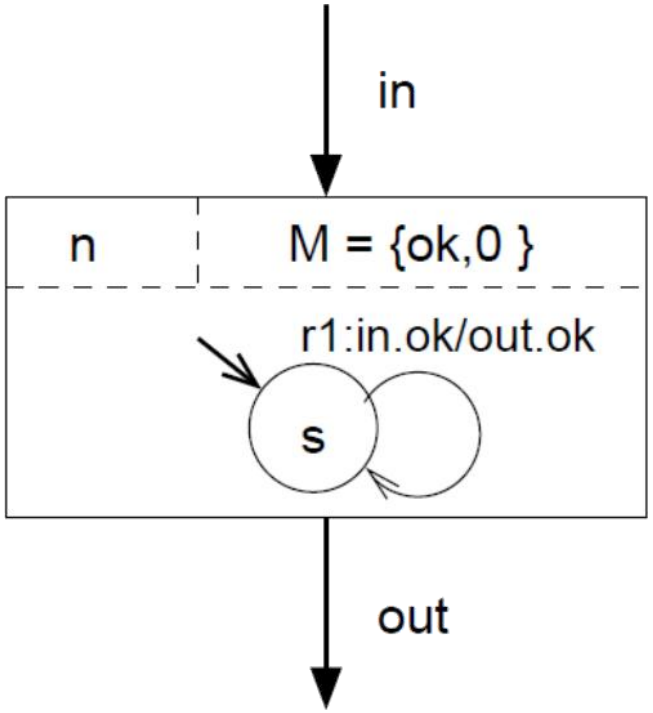
- Egy token kapacitású csatornák
- Hálózat:
  - DFN = ({n}, {in, out}, {(s,0,0), (s,ok,0), (s,0,ok), (s,ok,ok)})
- Csomópontok:
  - n = ({in}, {out}, {s}, s, {ok,0}, {r1})
- Tüzelések:
  - r1=<s; in=ok; s; out=ok; 0>



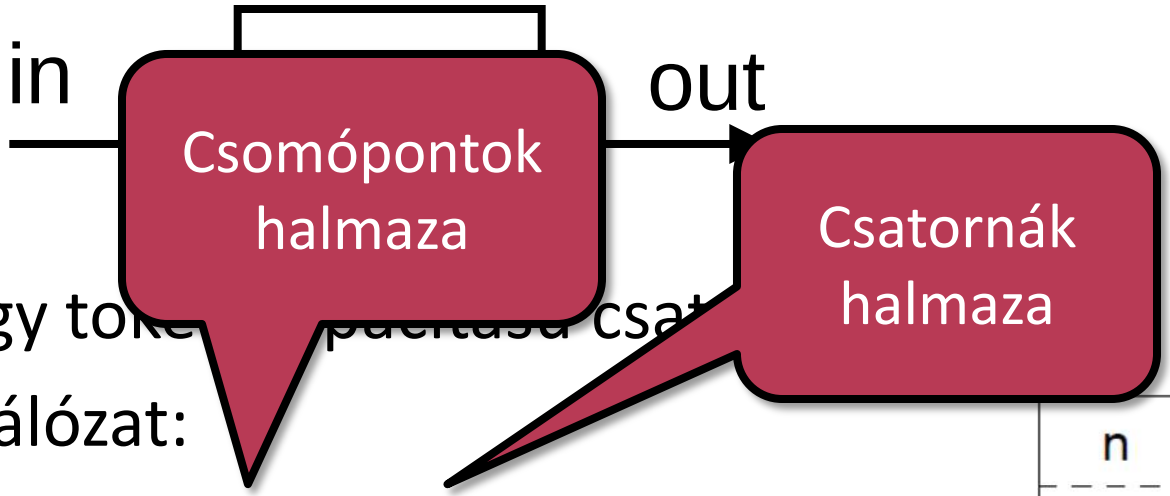
Egy példa



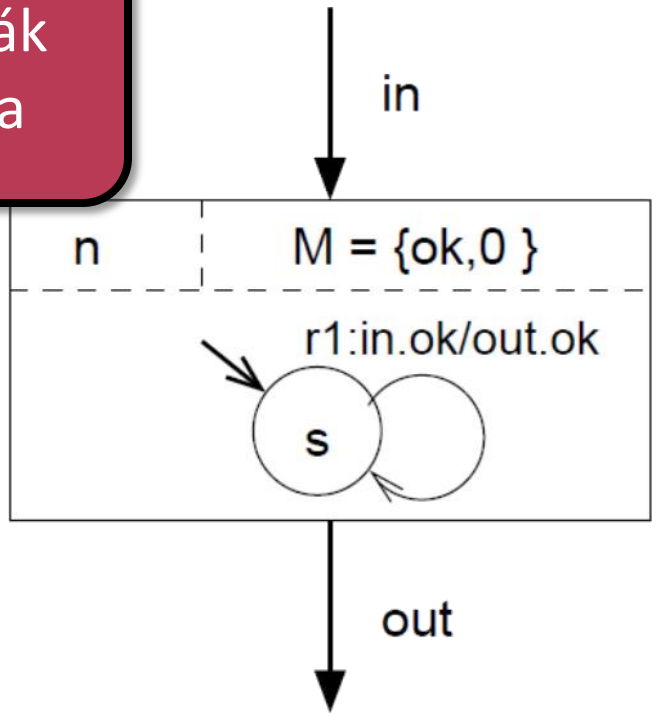
- Egy tok[illegible]puffású csatornák
- Hálózat:
  - DFN = ({n}, {in, out}, {(s,0,0), (s,ok,0), (s,0,ok), (s,ok,ok)})
- Csomópontok:
  - n = ({in}, {out}, {s}, s, {ok,0}, {r1})
- Tüzelések:
  - r1=<s; in=ok; s; out=ok; 0>



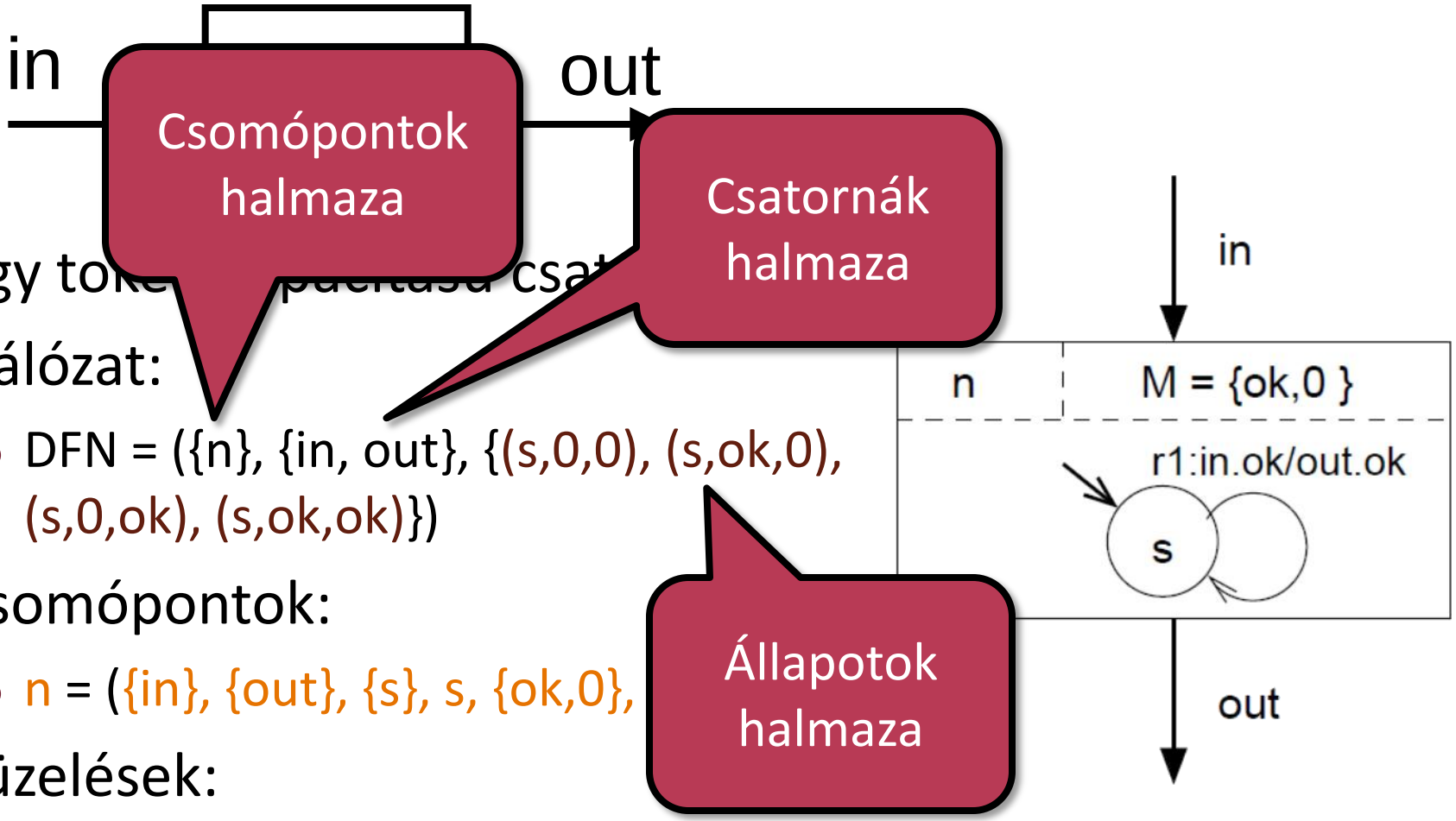
Egy példa



- Egy tok[illegible] csat
- 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>

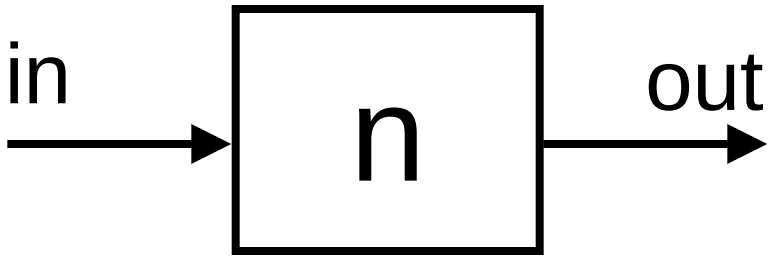


Egy példa

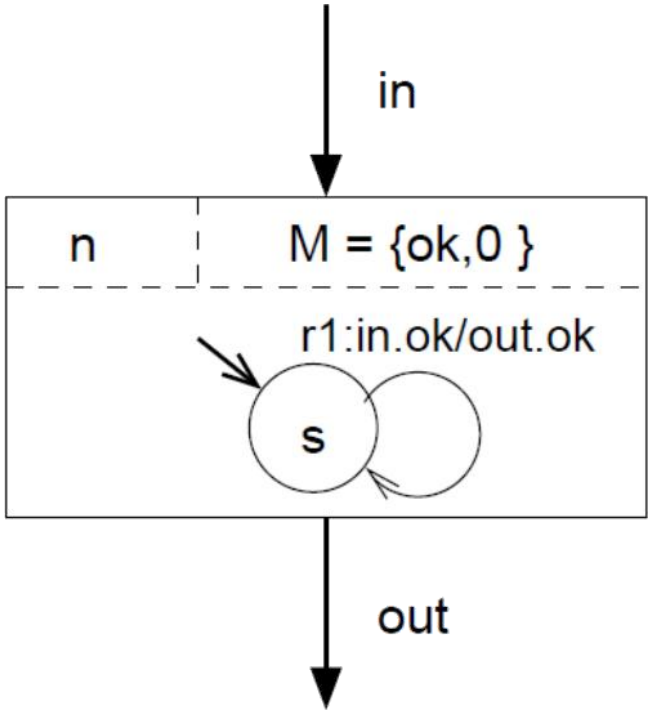


- Egy tok[illegible]puffású csat
- Hálózat:
  - DFN = ({n}, {in, out}, {(s,0,0), (s,ok,0), (s,0,ok), (s,ok,ok)})
- Csomópontok:
  - n = ({in}, {out}, {s}, s, {ok,0},
- Tüzelések:
  - r1=<s; in=ok; s; out=ok; 0>

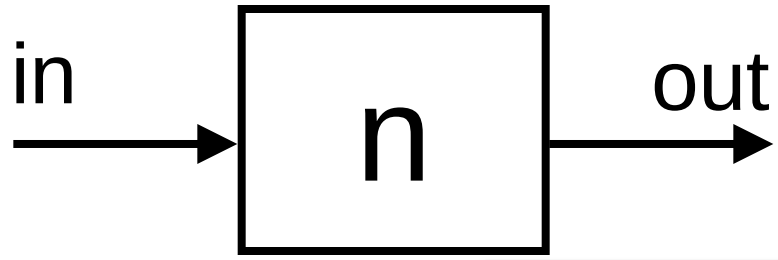
Egy példa



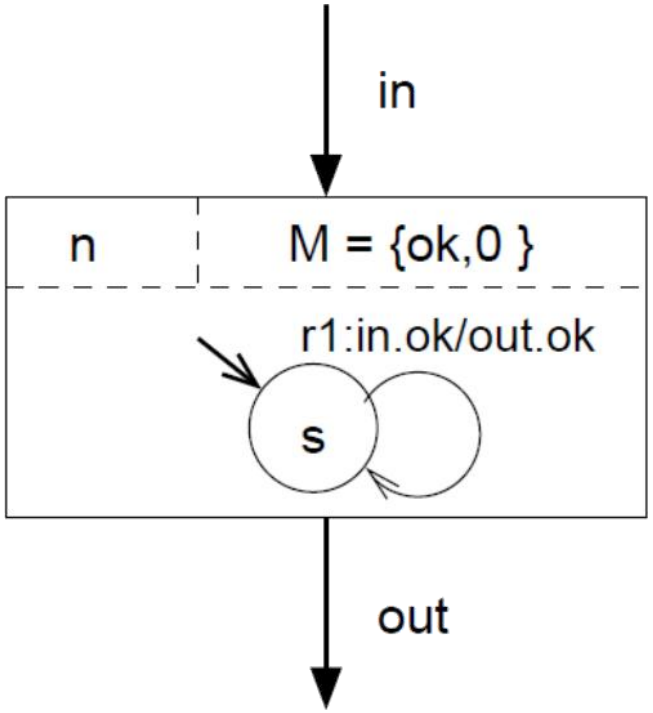
- acitású csatornák
- Bemenő csatornák halmaza
- , out}, {(s,0,0), (s,ok,0), (s,0,o s,ok,ok)})
- Csomópontok:
  - n = ({in}, {out}, {s}, s, {ok,0}, {r1})
- Tüzelések:
  - r1=<s; in=ok; s; out=ok; 0>



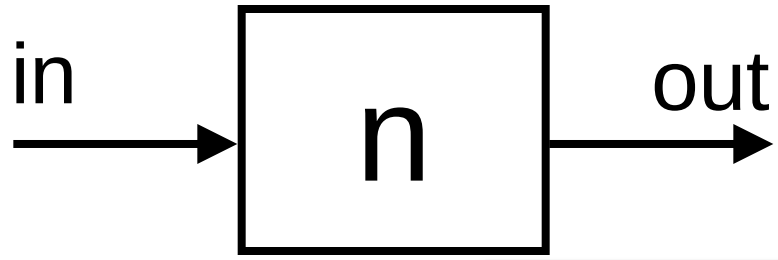
Egy példa



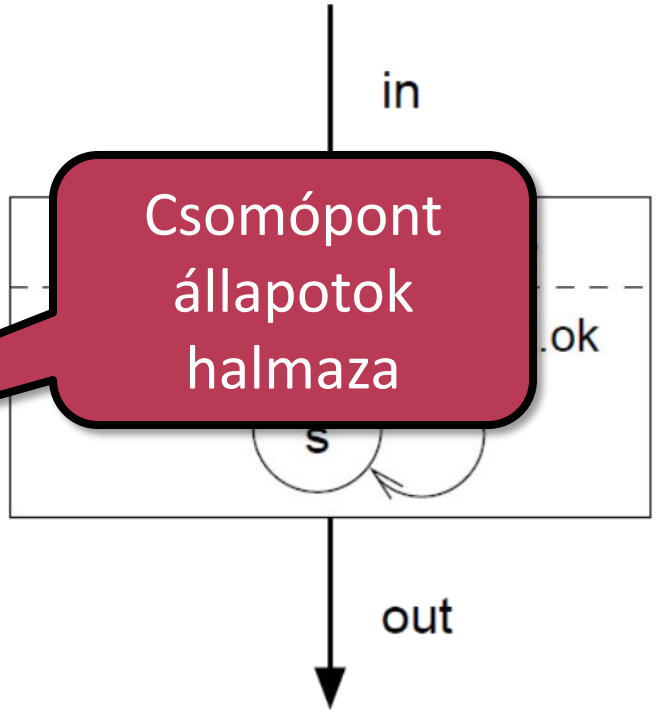
- Bemenő csatornák halmaza
- Kimenő csatornák halmaza
- Csomópontok:
  - n = ({in}, {out}, {s}, s, {ok,0}, {r1})
- Tüzelések:
  - r1=<s; in=ok; s; out=ok; 0>



Egy példa

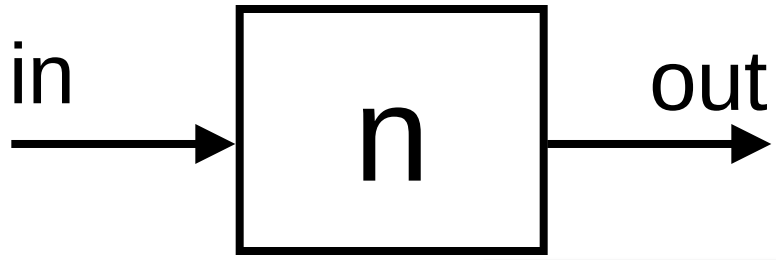


- Bemenő csatornák halmaza
- Kimenő csatornák halmaza
- Csomópont állapotok halmaza

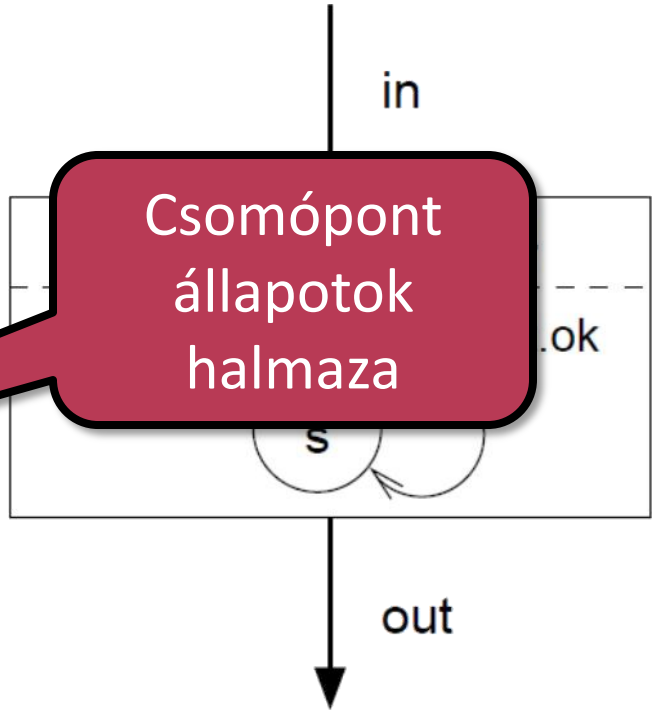


- Csomópontok:
  - n = ({in}, {out}, {s}, s, {ok,0}, {r1})
- Tüzelések:
  - r1=<s; in=ok; s; out=ok; 0>

Egy példa



- ac
- n, ou (0),
  (s,0,o s,ok,ok
- Csomópontok:
  - n = ({in}, {out}, {s}, s, {ok,0}, {r1})
- ; out=ok; 0>



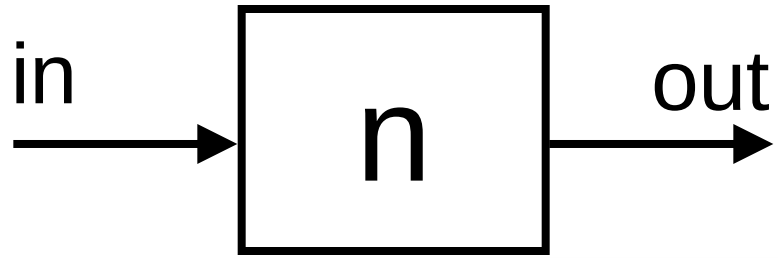
Bemenő csatornák halmaza

Kimenő csatornák halmaza

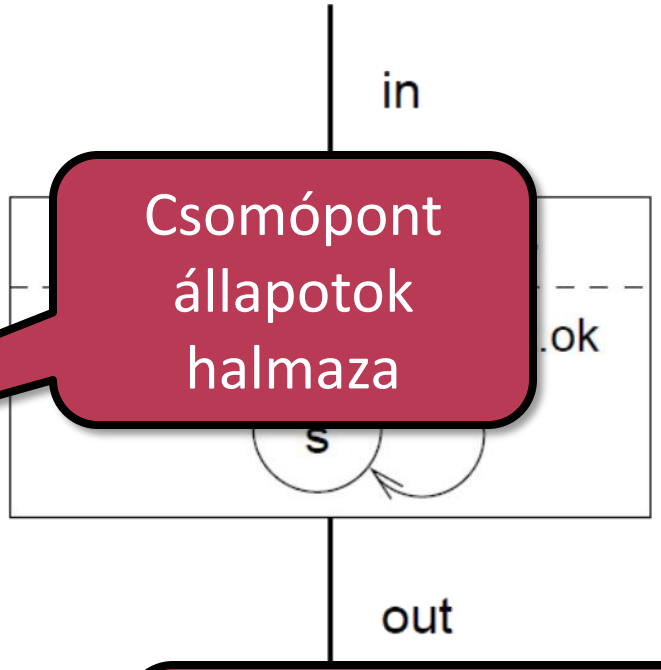
Csomópont állapotok halmaza

Tokenek halmaza

Egy példa



- ac
- n, ou
  (s,0,o s,ok,ok
- Csomópontok:
  - n = ({in}, {out}, {s}, s, {ok,0}, {r1})
- ; out=ok; 0>



Bemenő csatornák halmaza

Kimenő csatornák halmaza

Csomópont állapotok halmaza

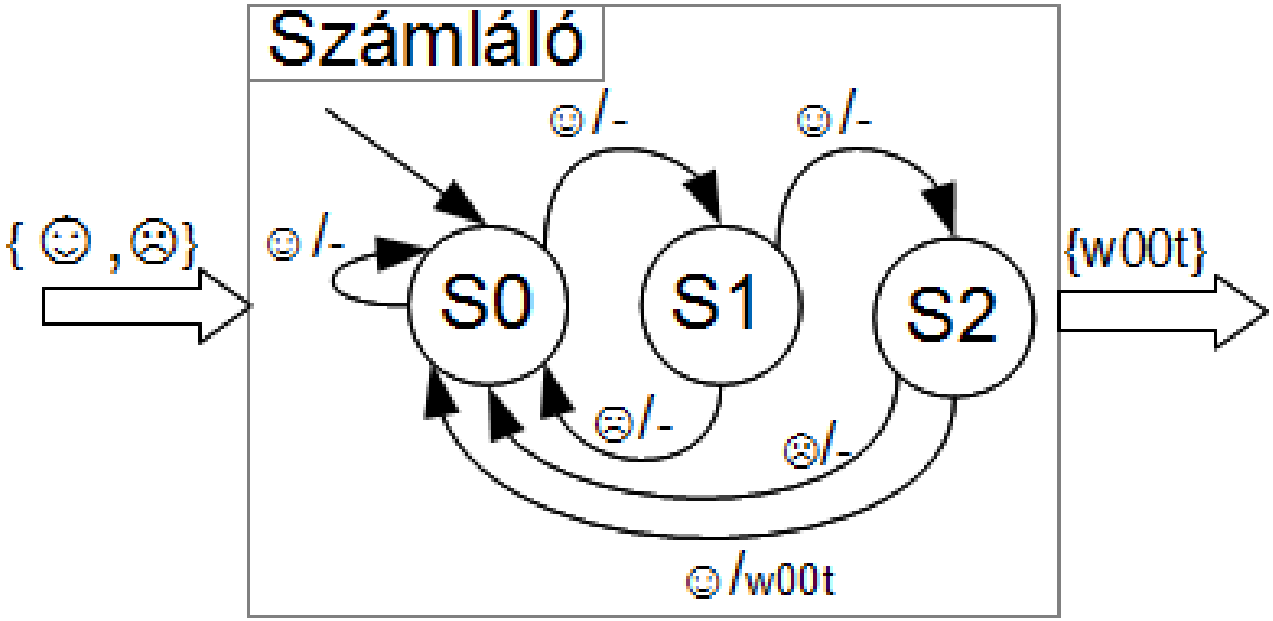
Tokenek halmaza

Tüzelések halmaza

Példa - Számláló

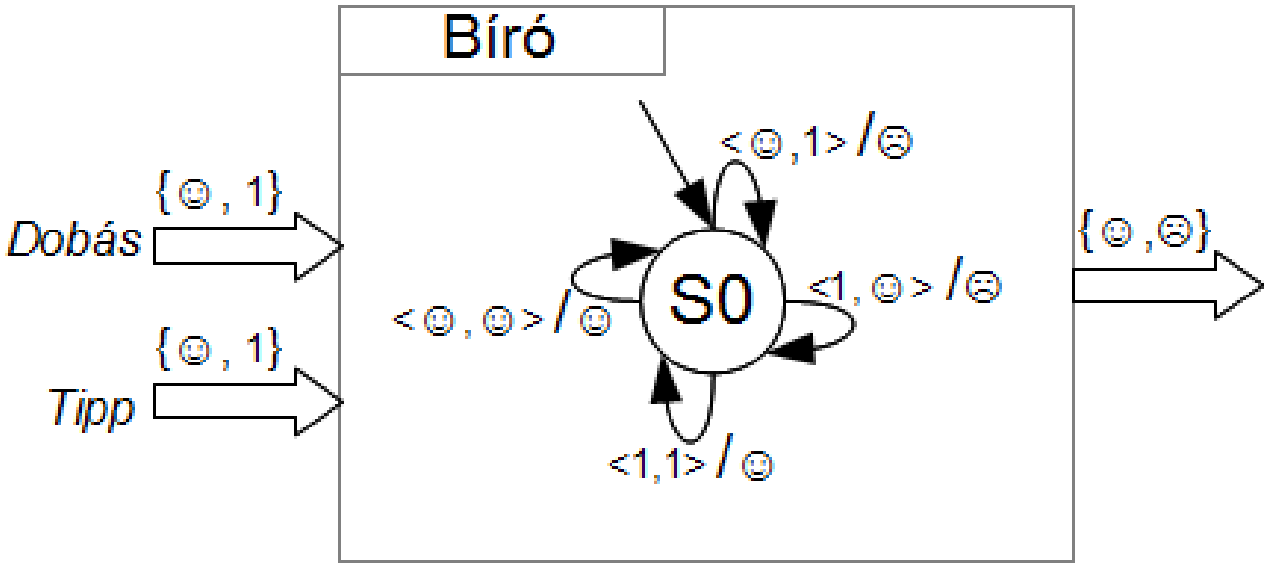
- Készítsük el egy adatfolyam „Számláló” csomópontját,

  amely számláló bemenetén ☺ és ☹ tokeneket kap, majd a kimenetén a *w00t* token jelenik meg, amennyiben egymás után 3 db ☺ jelet olvas a bementről.



Példa - Bíró

- Készítsük el egy adatfolyam „Bíró” csomópontját. A csomópont két bemenetéről egyszerre olvassa be egy érme feldobásának eredményét és a játékos tippjét. Ha a dobás és a tipp megegyezik a kimeneten a ☺ jelet, egyébként a ☹ jelet adja ki.



{Tipp.1,Dobás.☺} := <1,☺>

Adatfolyam modellek kiértékelése

+ Interaktív szimuláció
- Validáció, helyességbizonyítás (direkt/indirekt)
  - Dinamikus tulajdonságok: elérhetőség, holtpontmentesség
+ Időbeli analízis (indirekt)
  - Tüzelési szabályokban végrehajtási idő, mint valószínűségi változó
+ Hibaszimuláció (direkt, diszkrét esemény szimuláció)
  - Működési modell kiegészítése hibamodellel, hibahatások elemzése
+ Teszttervezés (indirekt)
  - Tesztgenerálás, tesztelhetőségi analízis, tesztkészlet optimalizálás
- Hibahatás analízis (direkt)
  - FMEA: hibamód és hatás analízis, hibafa és eseményfa generálás
- (Megbízhatósági analízis) (indirekt)
  - Klasszikus mértékek: megbízhatóság, rendelkezésre állás, MTBF, …

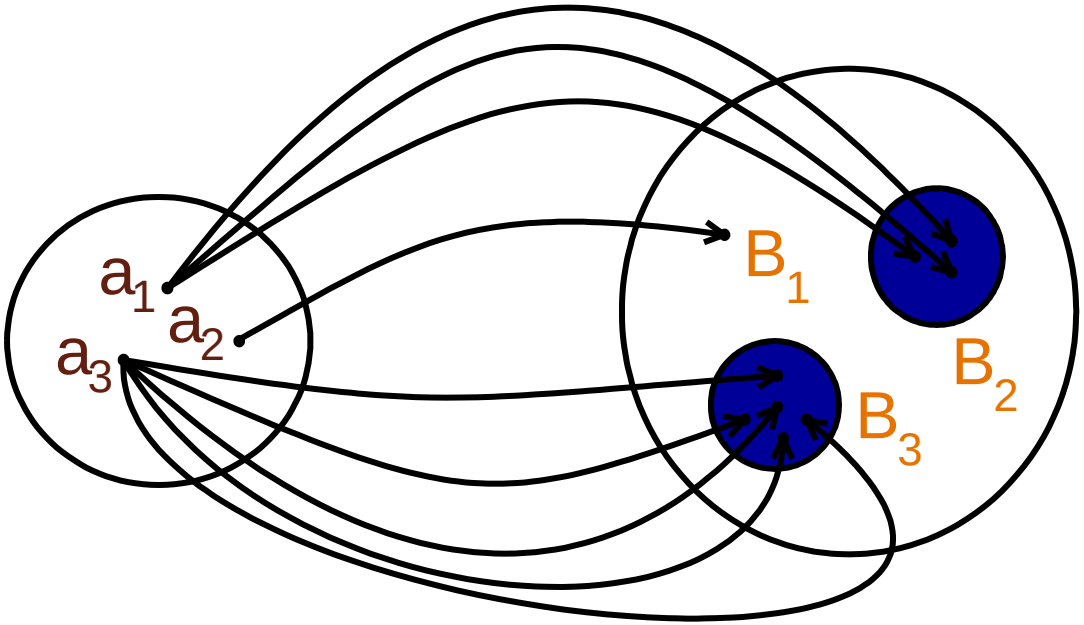
Modellfinomítás adatfolyam hálókra

Modellfinomítás:

- Többszintű modellezés
- Szintek közötti átjárás (formalizált szabályok)
- Állapot és viselkedési konzisztencia megőrzése

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
  - Eloszlás finomítás
    - dekompozíció, részkomponensekre bontás

Modellfinomítás adatfolyam hálókra

Fekete- és átlátszó dobozokra megfogalmazott elvek általánosítása adatfolyam hálókra:

- Értelmezési tartomány finomítás
  - Token halmaz finomítás
  - Állapothalmaz finomítás
- Struktúra finomítás
  - Csomópont helyettesítése adatfolyam alhálóval

Értelmezési tartomány finomítás

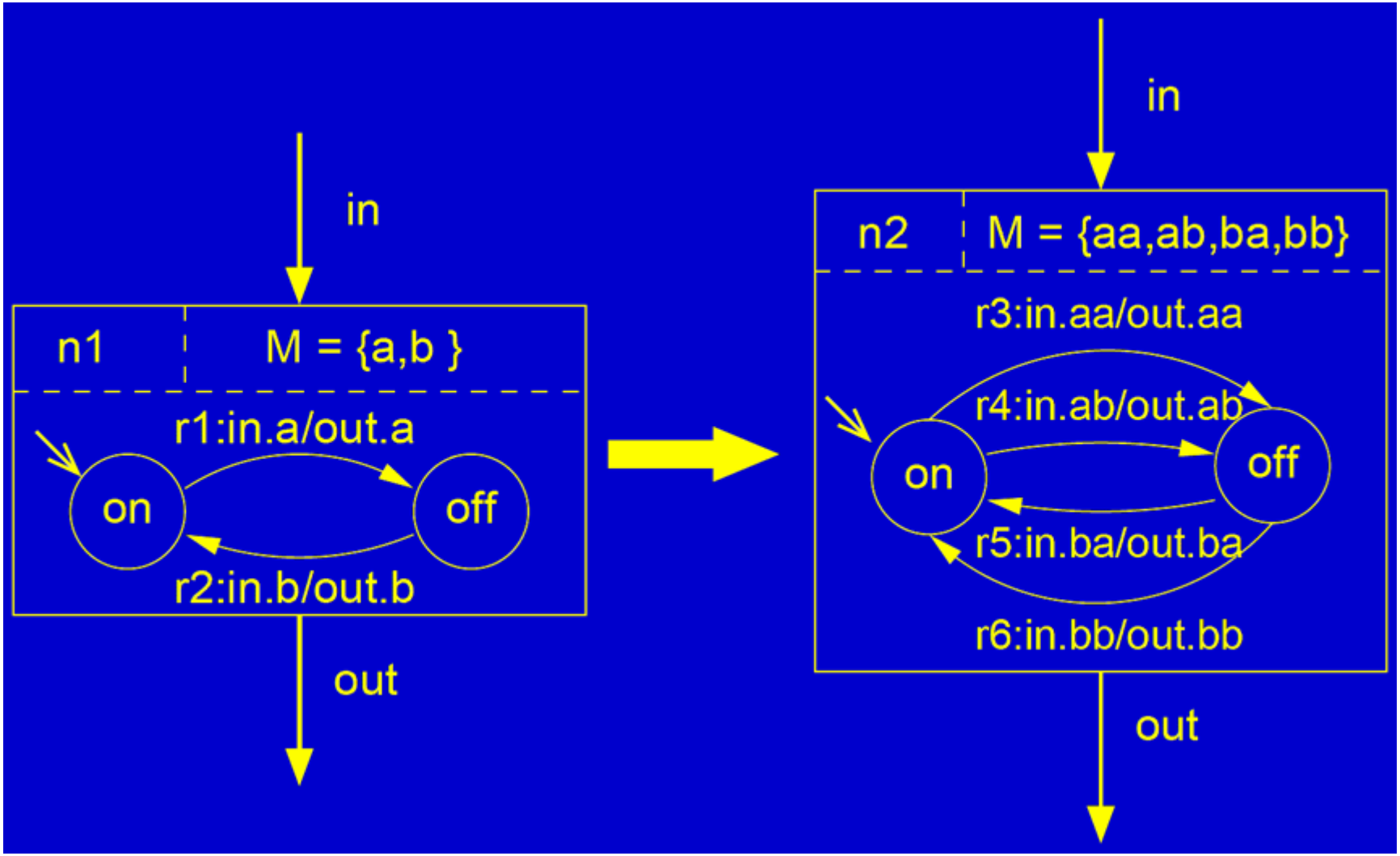
- Értelmezési tartomány finomítás
  - Tokenek halmazának finomítása: M'_n finomítottja M_n-nek
  - Állapotok halmazának finomítása: S'_n finomítottja S_n-nek
- Ki- és bemeneti csatornák változatlanok
- Tüzelési szabályok megfelelő változtatása

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}

- r1 = <on; in=a; off; out=a>
- r2 = <off; in=b; on; out=b>
- r11 = <on; in=aa; off; out=aa>
- r12 = <on; in=ab; off; out=ab>
- r21 = <off; in=ba; on; out=ba>
- r22 = <off; in=bb; on; out=bb>

Értelmezési tartomány finomítás: tokenek



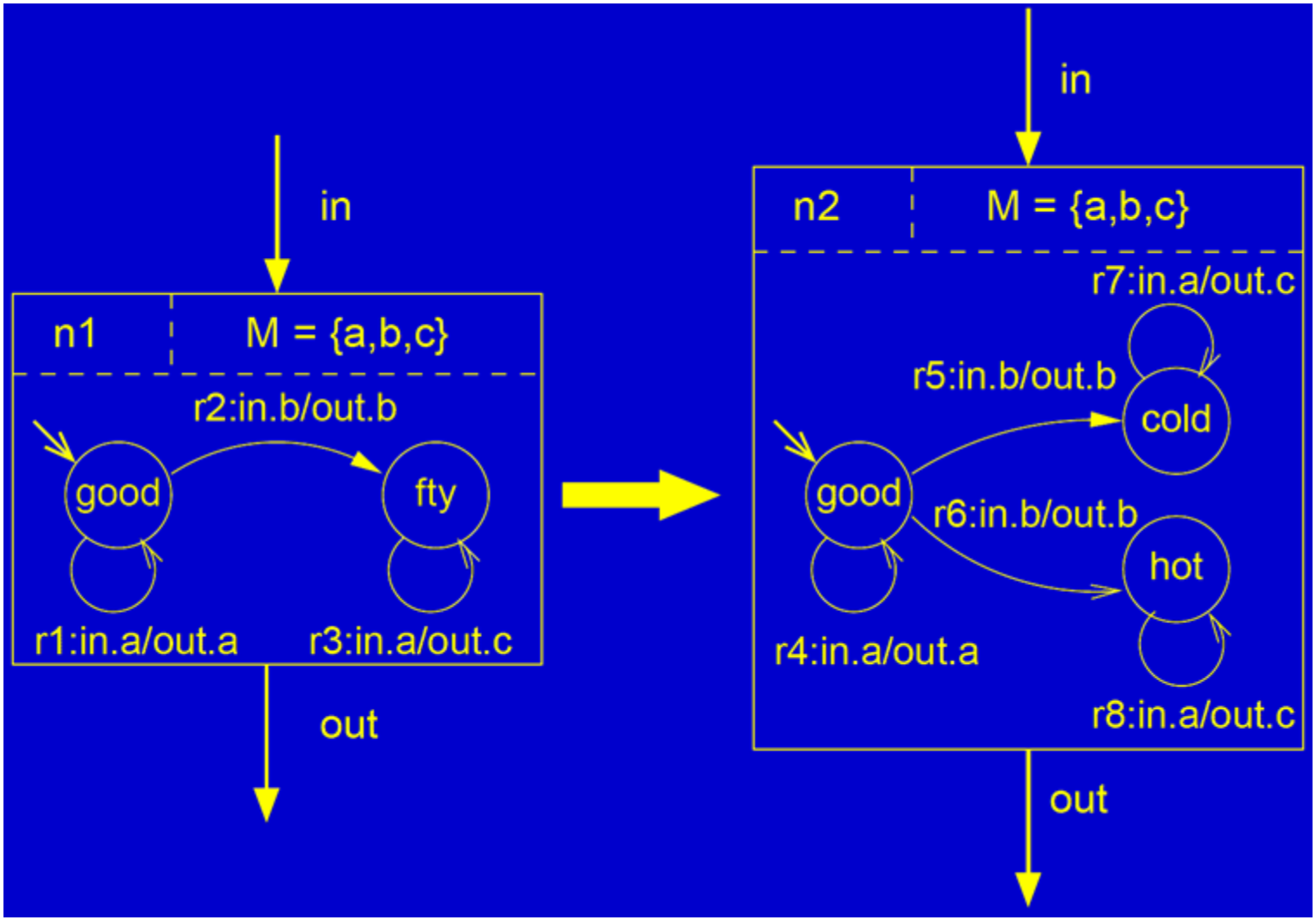
Állapotok halmazának finomítása: példa

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

- 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>

Értelmezési tartomány finomítás: állapotok



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; s1; ref_out=FTY>

  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; s1a; 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; s1b; ref_out=FTY>

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

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

   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=LOW; s0; ref_out=OK>

   r21=<s0; power_in=HIGH; s1a; ref_out=HIGH>

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

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

   r41=<s1a; power_in=FTY; s1b; 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; s1a; ref_out=HIGH>

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

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

r411=<s1a; power_in=LOW; s1b; ref_out=LOW>

r412=<s1a; power_in=HIGH; s1b; ref_out=LOW>

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

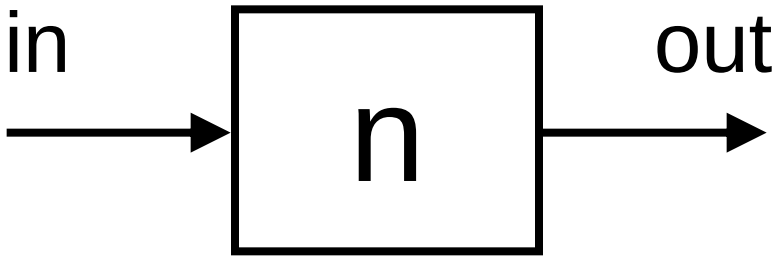
r422=<s1b; power_in=HIGH; s1b; ref_out=HIGH>

Bizonytalanság megszűnt

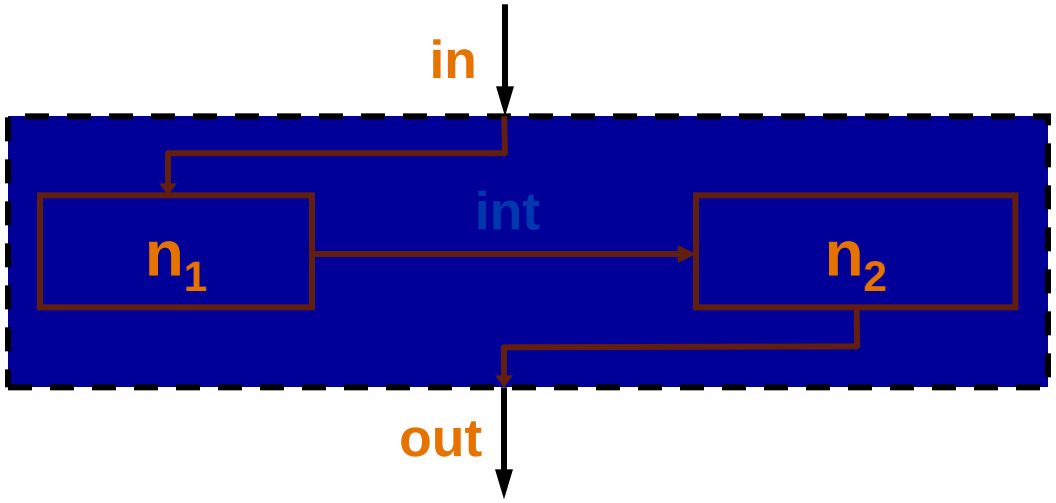
Struktúra finomítás

- 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

Struktúra finomítás: példa



$$\text{DFN} = \Re(n)$$



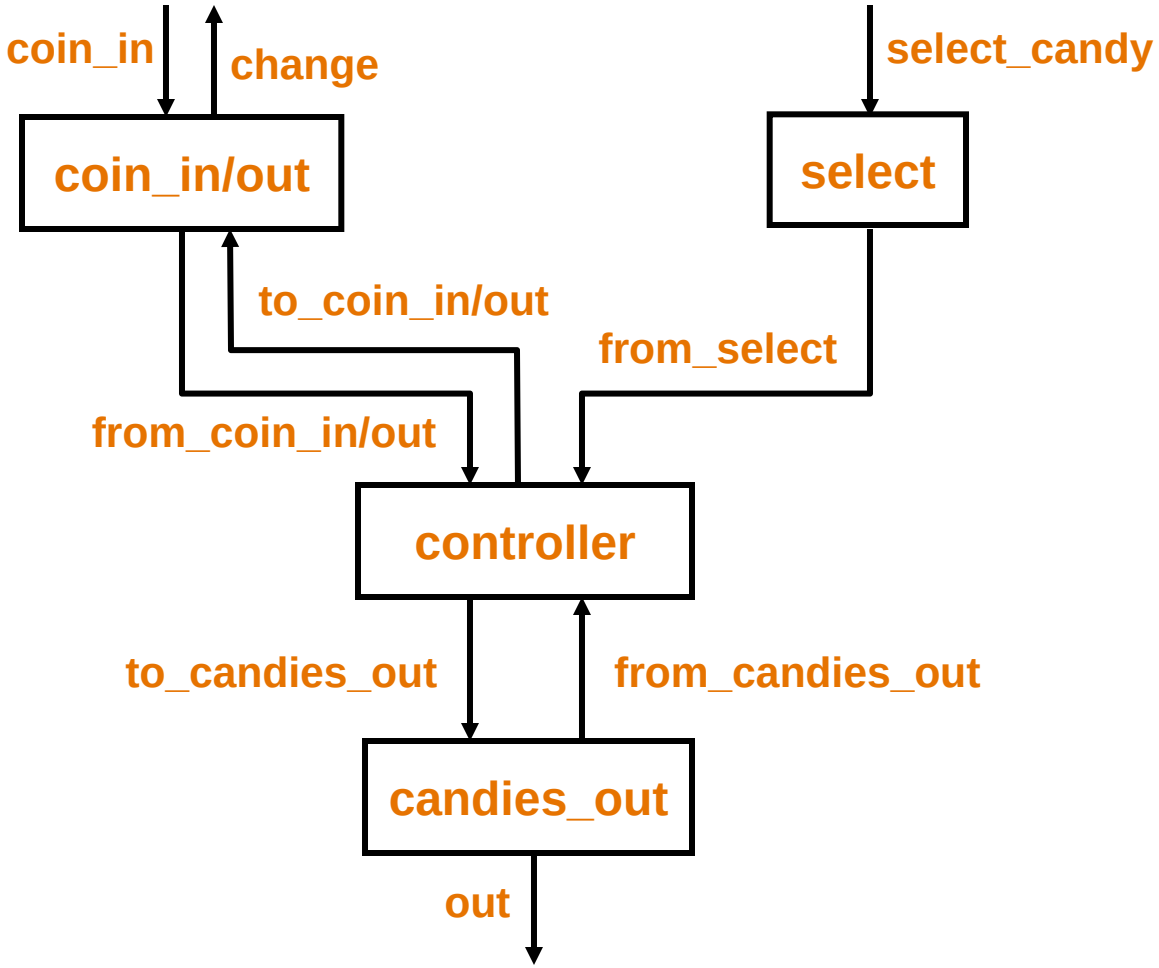
Struktúra finomítás: példa

	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	{⟨$r_{n1}1$; $r_{n2}1$⟩; ⟨$r_{n1}1$; $r_{n2}3$⟩} {⟨$r_{n1}2$; $r_{n2}2$⟩; ⟨$r_{n1}2$; $r_{n2}4$⟩}

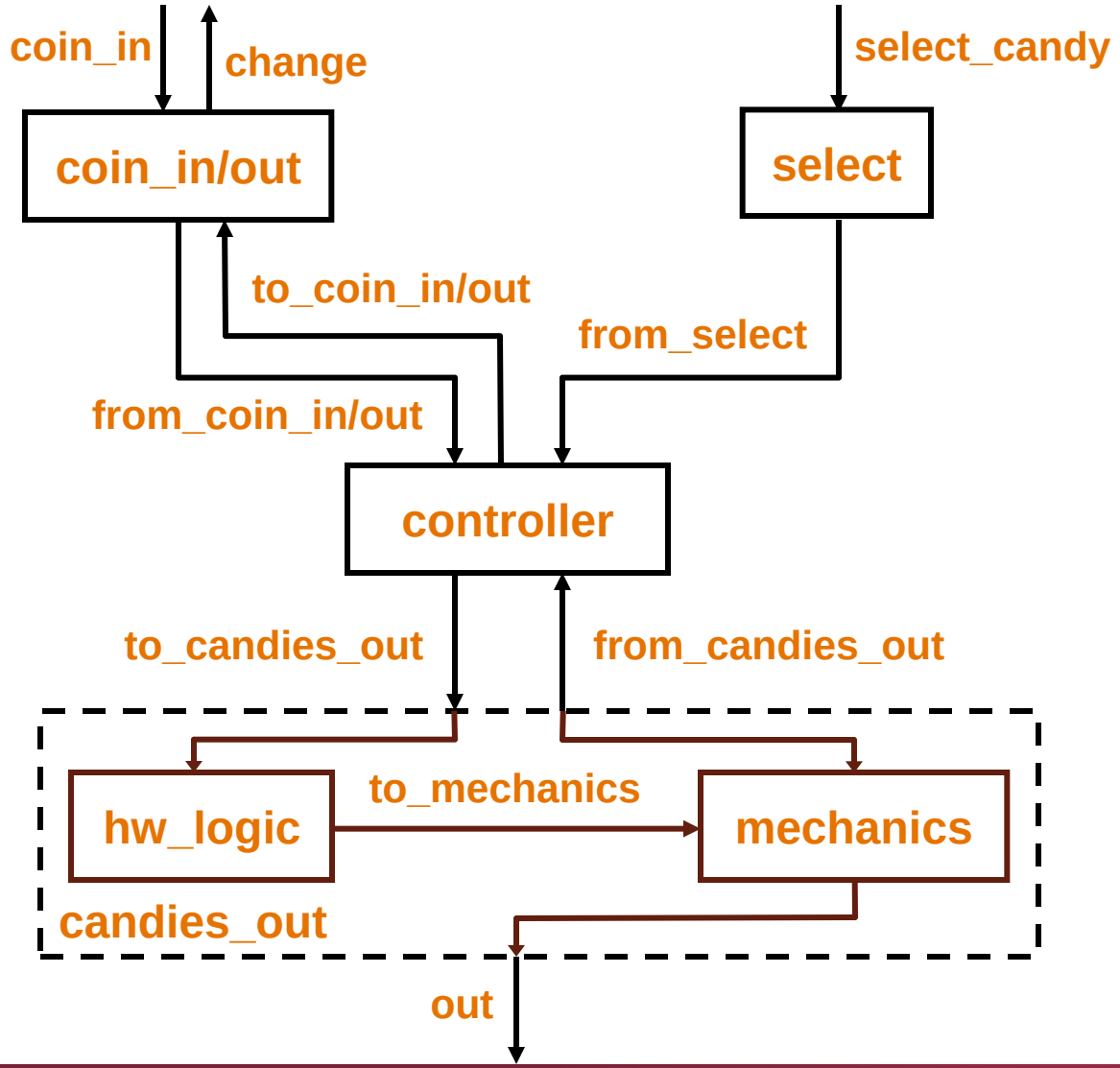
- r_n1 = <good; in=a; good; out=a>
- r_n2 = <good; in=b; fty; out=b>

- $r_{n1}1$ = <good; in=a; good; int=a>
- $r_{n1}2$ = <good; in=b; fty; int=b>
- $r_{n2}1$ = <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>

Édességautomata



Édességautomata finomítása



Finomítás ellenőrzés

1. Finomítási szabály-vezérelt tervezőrendszer
2. Definíciók alkalmazása próbálgatással
3. Véges automaták (FSM) felhasználása
   - Struktúra ellenőrzés
   - Csomópont-csomópont, illetve csomópont-részháló párok transzformációja → NDFST
   - Automata párokra biszimuláció keresése

Naplózott modellváltoztatások!

Adatfolyam csomópont → nemdeterminisztikus véges automata átalakítás

NDFST	jelentés	átírás az *n* csomópontból
Σ	Bemenő ABC	$(M_n)^i,\ i=\lvert I_n\rvert$
Γ	Kimenő ABC	$(M_n)^o,\ o=\lvert O_n\rvert$
S	Állapotok halmaza	S_n
S_0	Induló állapot	$S_n{}^0$
$\delta : S \times \Sigma \rightarrow S^*$	Állapotátmeneti függvény	$\delta\{r_i(s),\ r_i(X_{in}),\ r_i(s')\}$
$\omega : S \times \Sigma \rightarrow \Gamma^*$	Kimeneti függvény	$\omega\{(r_i(s),\ r_i(X_{in}),\ r_i(X_{out})\}$

Adatfolyam hálózat → nemdeterminisztikus véges automata átalakítás

DFN(N, C, S)

$\Sigma = (M_n)^i,\ i = |I| \text{ és } n \in N$

$\Gamma = (M_n)^o,\ o = |O| \text{ és } n \in N$

$S = S_{DFN},\ s_0 = s^0_{SFN}$

$\delta\{s(fs),\ im(fs),\ s'(fs)\},\ \forall fs \in FS$

$\omega\{s(fs),\ im(fs),\ om(fs)\},\ \forall fs \in FS$

Véges automaták biszimulációja

Cél:

- eldönteni két véges automatáról, hogy ekvivalensen viselkednek-e?

Kibővített kimeneti függvény ω: $S \times \Sigma^* \to \Gamma^*$

Definíció:

- ω_2 finomítása ω_1-nek, ha S_2 S_1-nek, Σ_2 Σ_1-nek, Γ_2 pedig Γ_1-nek finomítása

Definíció:

- $NDFST_1$ és $NDFST_2$ véges automaták között pontosan akkor van biszimuláció, ha ω_2 finomítása ω_1-nek.

Finomítás ellenőrzése

Adatfolyam csomópont → nemdeterminisztikus véges automata ($NDFST_1$)

Adatfolyam hálózat → nemdeterminisztikus véges automata ($NDFST_2$)

- Akkor helyes a finomítás, ha $NDFST_2$ finomítása $NDFST_1$-nek

```
function Refinement_Check()
    if Diff_Struct() then return FAILURE
    for all n1∈N1
        n2 vagy SDFN2 megfeleltetése n1-nek
        • n1 transzformációja NDFST1-vé
        • n2 vagy SDFN2 transzformációja NDFST2 -vé
        if ¬Bisimulation() then return FAILURE
    end for
    return SUCCESS
end function
```

Modellbővítés

Modellezni kívánt mechanizmusok:

- Hibák
- Hibahatások
- Hibaterjedés

Ez megtehető a hibátlan modell szisztematikus kibővítésével (a hibamodell alapján).

Modellbővítés

1. Fizikai modell (alacsony szint)
   - Szoros kapcsolat a fizikai hibaokkal
2. Logikai modell (magasabb szinteken)
   - Modell perturbáció
     - Szisztematikus módon bővítik a modellt a hibás működéssel
     - if-then-else vagy switch-case típusú leírás
     - Perturbációk listája a hibalista
   - Gráfmodellek
     - Csomópontok a rendszer komponenseit modellezik
     - Mindegyik leírásában egy előre kiszámított hibás működés
     - Hibás komponensnél hiba a kimeneten továbbterjed

Hibamodellezés

Neminterpretált (kvalitatív) modellezés esetén:

- Token lehet jó vagy hibás (színezés)
- Részletesebb hibamodell $\rightarrow$ többszínű

Hiba komolysága alapján:

- korrekt
- inkorrekt
- fatális
- katasztrofális

Lebegőpontos számítás eredménye:

- pontos
- közelítőleg pontos
- túl kicsi
- túl nagy

Hibamodellezés

Hibát jelző tokenek mellett a csomópontok állapotait is ki kell bővíteni hibát jelző állapotokkal.

→ Új tüzelési szabályok bevezetése

Hibatűrés aspektusai

error-free operation	<ok; in=ok; ok; out=ok; 0>
erroneous operation	<fty; in=ok; fty; out=fty; 0>
internal fault	<ok; ; fty; ; 0>
external fault	<ok; in=fty; fty; out=fty; 0>
repair	<fty; in=ok; ok; out=ok; 0>
error correction	<ok; in=fty; ok; out=ok; 0>
error masking	<fty; in=fty; fty; out=ok; 0>
error propagation	<ok; in=fty; ok; out=fty; 0>

Bizonytalanság modellezése

Szubjektív és nem teljes ismeretek ábrázolása.

Nemdeterminizmus lehet

- Funkcionális
- Alulspecifikáltság következménye

Bizonytalanság modellezése

Bizonytalanság

→ konkurens átmenetek

- különböző megoldások a választásra

- (Állapotátmeneti) valószínűségek
  - sztochasztikus modellek (pl. Markov hálók)
  - bonyolult kiértékelési módszerek
- Érték nemdeterminizmus
  - küldött token értéke bizonytalan
  - megoldható tüzelési (funkcionális) nemdeterminizmusra visszavezetve is

Édességautomata

komponens	hiba
pénz be/ki egység	ok rec
kiválasztó egység	ok cont
vezérlő egység	ok inc
édességkiadó egység	ok ctrl stk

Édességkiadó egység kibővített modellje

r1 = <ok; to_candies_out = ok; ok; from_candies_out = ok; out = ok; 0>

r2 = <ok; to_candies_out = inc; ok; from_candies_out = ok; out = inc; 0>
r3 = <ok; to_candies_out = x; ok; from_candies_out = ok; out = x; 0>

r4 = <ctrl; to_candies_out = ok; ctrl; from_candies_out = ok; out = inc; 0>
r5 = <ctrl; to_candies_out = inc; ctrl; from_candies_out = ok; out = x; 0>
r6 = <ctrl; to_candies_out = x; ctrl; from_candies_out = ok; out = x; 0>

r7 = <stk; to_candies_out = ok; stk; from_candies_out = dead; out = dead; 0>
r8 = <stk; to_candies_out = inc; stk; from_candies_out = dead; out = dead; 0>
r9 = <stk; to_candies_out = x; stk; from_candies_out = dead; out = dead; 0>