Kiberfizikai rendszerek

A “fizikai” vonatkozásokról …
utolsó folytatás

2015. november 24.

Fuzzy modellezés és szabályozás

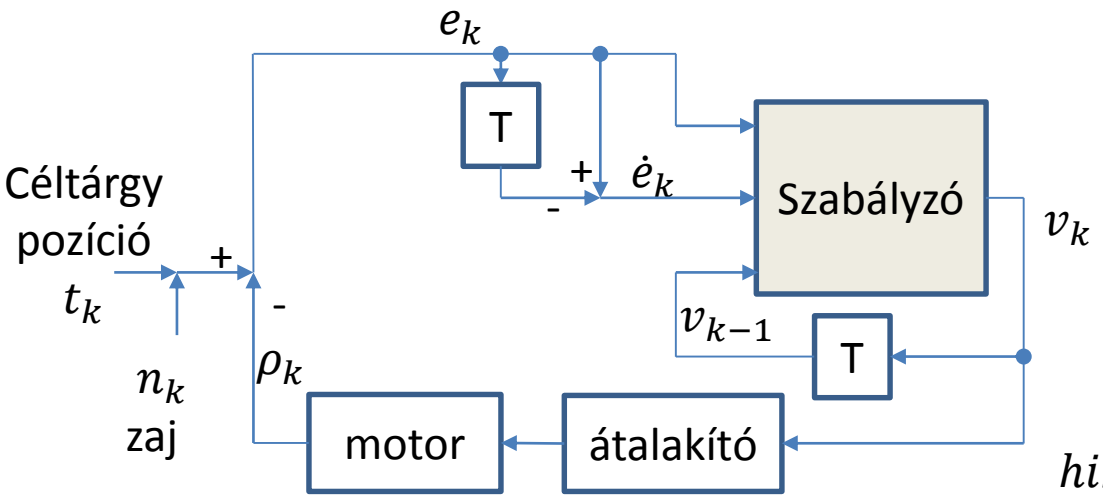
Példa: Adaptív célkövető rendszer:

egy azimut (0 … 180 fok) és egy emelkedés (0 … 90 fok) irányú forgató mechanizmussal.

Azimut: az a szög, amit a kijelölt irány vízszintes vetülete a déli vagy északi iránnyal bezár.

Szenzor: minden olyan eszköz alkalmas, amely kellő pontossággal képes a célra mutatni.

Laser, videokamera, nagy nyereségű antenna.



Jelölések:

t_k	céltárgy pozíció
n_k	megfigyelési zaj
ρ_k	célkövető pozíció
e_k	követési hiba
$\dot{e}_k$	követési hiba változás
v_k	becsült szögsebesség
T	mintavételi idő

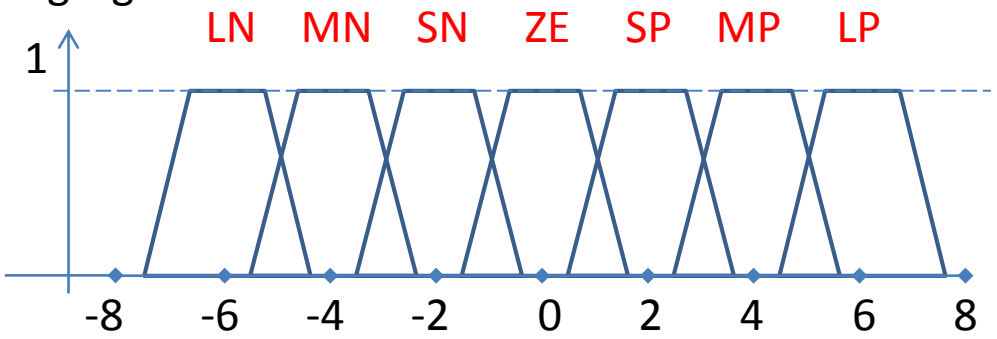
$$\rho_k = \rho_{k-1} + T v_{k-1} + hiba$$

$hiba =$ pozicionálási bizonytalanság

Fuzzy szabályozó: A becsült szögsebesség tartománya: [-6,6]. Mivel $|v_k| \le \frac{9.0}{T} fok/sec$ azimut irányban, $|v_k| \le \frac{4.5}{T} fok/sec$ emelkedés irányban, ezért az egyes csatornák erősítései: $\frac{1.5}{T}$ és $\frac{0.75}{T}$.

Fuzzy modellezés és szabályozás

Fuzzy szabályozó: Heurisztikus szintállító szabályokat tartalmaz az $e_k, \dot{e}_k$ és v_{k-1} értékei alapján. 7 fuzzy szint értéket definiálunk:



Minden bemenethez egy hételemű vektort rendelünk!

LN=Large Negative
MN=Medium Negative
SN=Small Negative
ZE=Zero
SP=Small Positive
MP=Medium Positive
LP=Large Positive

1 → (0 0 0 0.7 0.7 0 0)
-4 → (0 1 0 0 0 0 0)
3.8→ (0 0 0 0 0.1 1 0)

Fuzzy-asszociatív-memoria (FAM) szabályok:

$IF\ e_k = MP \wedge \dot{e}_k = SN \wedge v_{k-1} = ZE\ THEN\ v_k = SP$ (MP,SN,ZE;SP)

Az i-edik FAM szabály skalár értéke: $w_i = \min$(tagsági értékek).

Példa:

$e_k = 2.6$

$\dot{e}_k = -2.0$

$v_{k-1} = 1.8$

LN	MN	SN	ZE	SP	MP	LP
0	0	0	0	1	0.4	0
0	0	1	0	0	0	0
0	0	0	0.1	1	0	0

$m(\ldots)$: tagsági értékek

$m_{MP}(e_k) = 0.4$

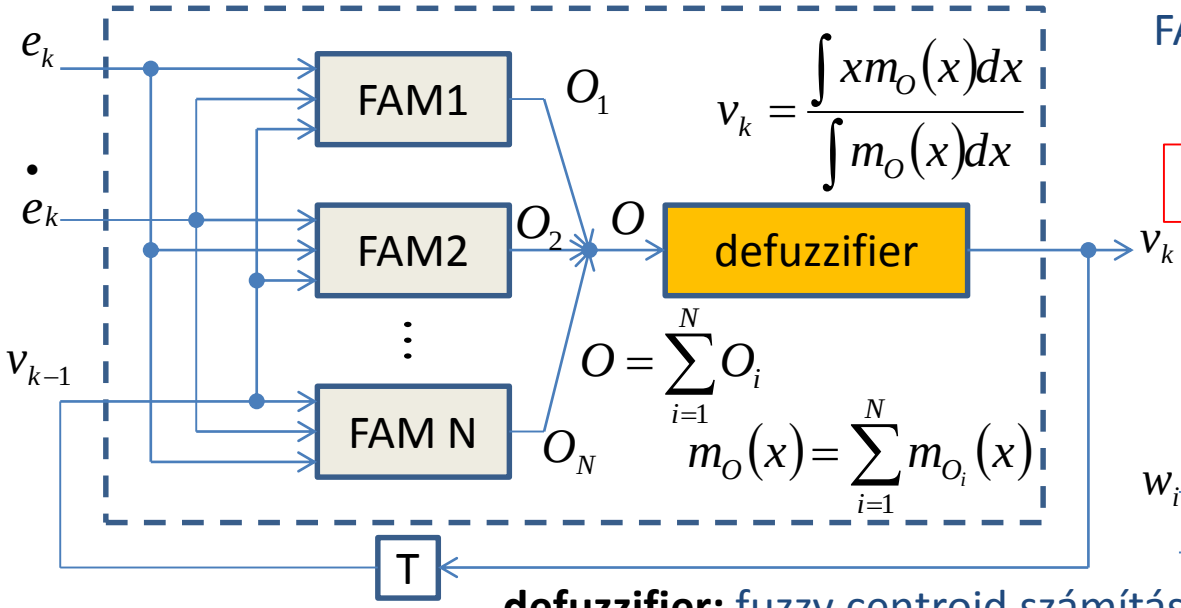
$m_{SN}(\dot{e}_k) = 1$

$m_{ZE}(v_{k-1}) = 0.1$

$w_i = \min(0.4,1,0.1) = 0.1$

Fuzzy modellezés és szabályozás

A szabályozó kialakítása:



defuzzifier: fuzzy centroid számítás

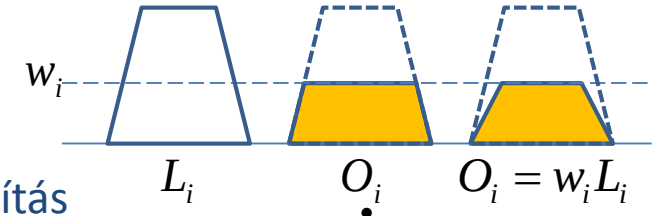
A kimeneti fuzzy halmaz alakja a FAM szabály kódolásától függ:

$$m_{O_i}(x) = \min\left(w_i, m_{L_i}(x)\right)$$

Korreláció-minimum kódolás

$$m_{O_i}(x) = w_i m_{L_i}(x)$$

Korreláció-szorzat kódolás



A fuzzy szabályzó implementációja: FAMi szabály: (MP,SN,ZE;SP)

$e_k = 2.6 \quad \dot{e}_k = -2.0 \quad v_{k-1} = 1.8$

$$w_i = \min\left(m_{MP}(e_k), m_{SN}(\dot{e}_k), m_{ZE}(v_{k-1})\right) = \min(0.4,1,0.1) = 0.1$$

$$w_i = \min\left(m_{ZE}(e_k - c_{MP}), m_{ZE}(\dot{e}_k - c_{SN}), m_{ZE}(v_{k-1} - c_{ZE})\right)$$

$$w_i = \min\left(m_{ZE}(-1.4), m_{ZE}(0), m_{ZE}(1.8)\right) = \min(0.4,1,0,1) = 0.1$$

Minden fuzzy halmaz alakja azonos:

Pl.: $m_{SP}(x) = m_{ZE}(x-2).$

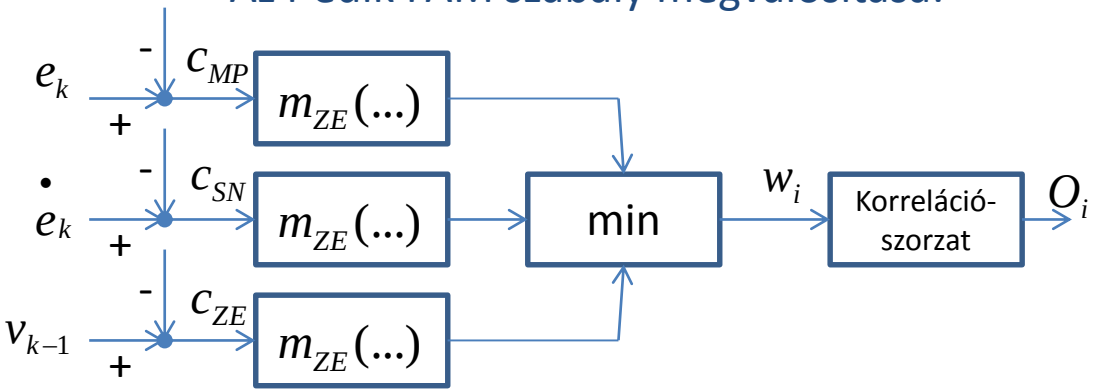
Általában: $m_{L_i}(x) = m_{ZE}(x - c_{L_i})$

c_{L_i} a tagsági függvény centroidja.

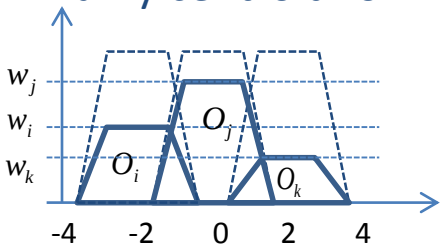
$$m_{O_i}(x) = w_i m_{ZE}(x - c_i)$$

Fuzzy modellezés és szabályozás

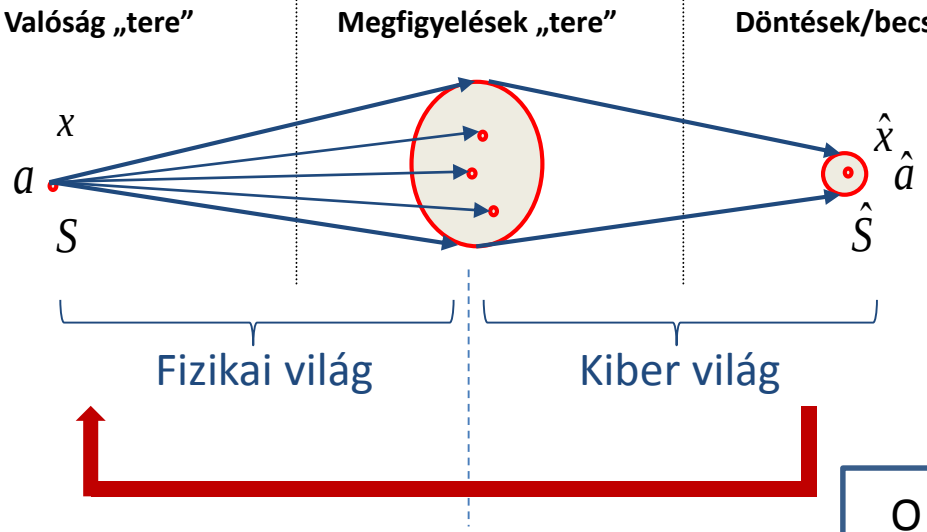
Az i-edik FAM szabály megvalósítása:

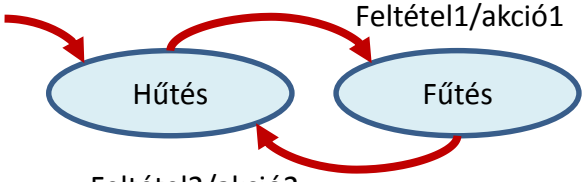


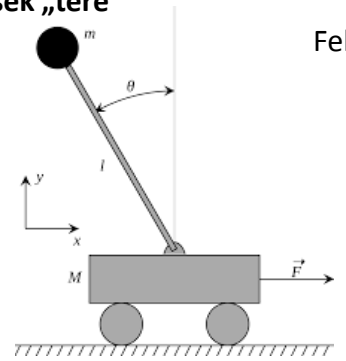
A fuzzy centroidhoz:

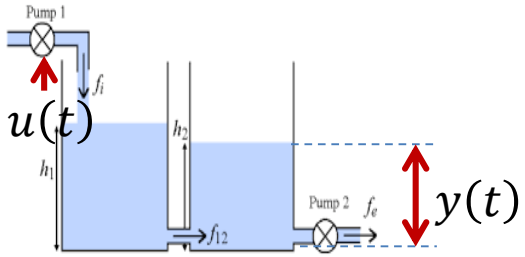


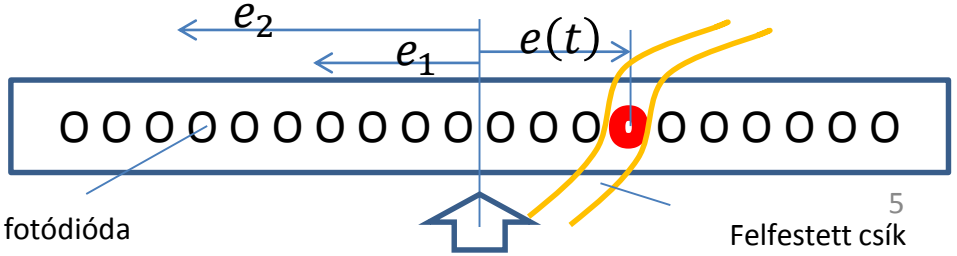
Összefoglalás:









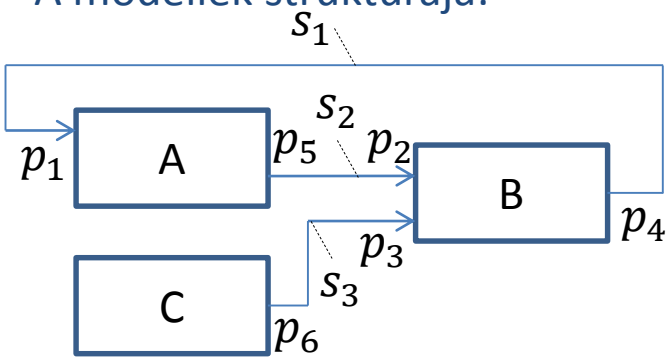


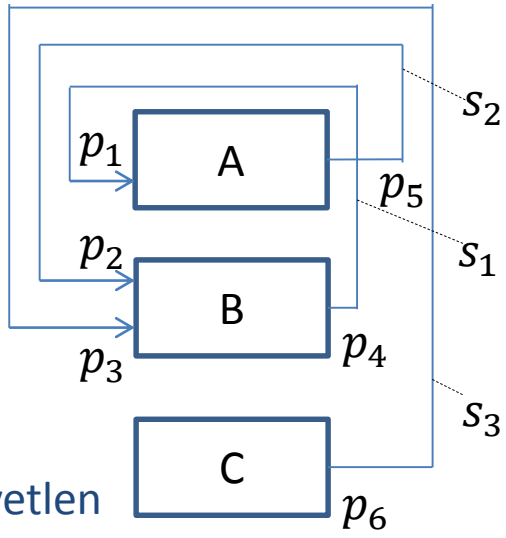
Konkurens számítási modellek (MoC)

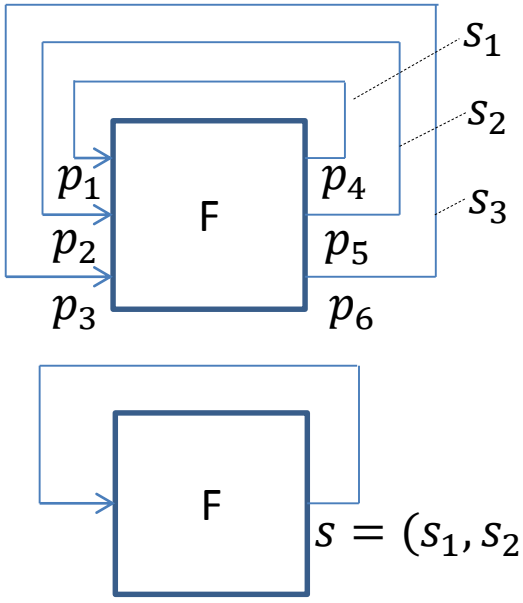
Komponensekből építkezünk. A komponensek bemeneti és kimeneti portokkal rendelkező aktorok, amelyek bemeneti stimulusok hatására végrehajtott akciók halmazával jellemezhetők.

Aktor: végrehajtó egység

A modellek struktúrája:







$s = (s_1, s_2, s_3)$

Az aktorok összeköttetése leírható egyetlen párhuzamos/összetett, visszacsatolt aktorral.

Az aktorok kommunikációja a jel, ami egy vagy több kommunikációs eseményből áll.

$s: R \rightarrow V_s \cup \{nincs\ esemény\};$ V_s a jel "jellegzetességeinek" halmaza, értékkészlete (type).

Példa: Egy órajel: $s(t) = \begin{cases} van\ esemény\ (present), ha\ t\ a\ P\ egészszámú\ többszöröse \\ nincs\ esemény\ (absent), egyébként \end{cases}$

Példa: $s: R \rightarrow V_s;$ Az $s(t)$ értékek megszámlálhatatlanul végtelen halmazának minden eleme egy kommunikációs esemény.

Konkurens számítási modellek (MoC)

Aktor hálózatok egyenletrendszerként: $s_2 = A(s_1);\ s_1 = B(s_2, s_3);\ s_3 = C(\Phi) =$ konstans.

$F(s_1, s_2, s_3) = \big(B(s_2, s_3), A(s_1), C(\Phi)\big)$. $F(s) = s,$ ez az ún. fixpont szemantika.

I. Szinkron-reaktív (SR) számítási modellek: a végrehajtás szimultán és pillanatszerű (tüzelés).

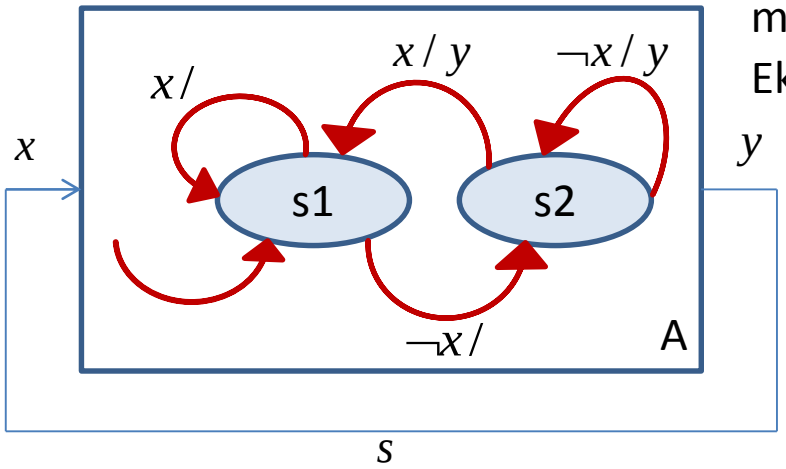
Az óra n-edik ütésére: $a_n : V_1 \cup \{nincs\_esemény\} \to V_2 \cup \{nincs\_esemény\}$, $s_2(n) = a_n\big(s_1(n)\big)$.

Az *F* aktor esetében a végrehajtás minden óraütésre egy fixpont:

$f_n\big(s_1(n), s_2(n), s_3(n)\big) = \big(s_1(n), s_2(n), s_3(n)\big)$.

Hogyan?

Példa: Ha az A aktor a reakció bekövetkezésekor az s1 állapotban van, akkor s(n)=*absent*, mert a reakciónak s1-ből kiinduló átmenete ezt ad ki. Ekkor viszont A átlép s2-be. Ha az A aktor a reakciókor s2 állapotban van, akkor s(n)=*present*, és átlép s1-be.



Az i-edik állapot tüzelő függvénye:

$a_i : \{present, absent\} \to \{present, absent\}$

$s(n) = a_i\big(s(n)\big) \to a_i$ fixpontja.

Megjegyzés: Lehet, hogy nincs fixpont, lehet, hogy több van: alternatív szemantikák.

Konkurens számítási modellek (MoC)

Szinkron-reaktív nyelvek: Elsődleges cél interakció fenntartása a környezettel.

Példák: Lustre (1991), Esterel (1992), Signal (1991), Statecharts (1987), Statemate (1990), SCADE (Safety Critical Application Development Environment) (2003) → Airbus felhasználás.

Előny: erősen támogatják a formális analízis és verifikációs technikákat.

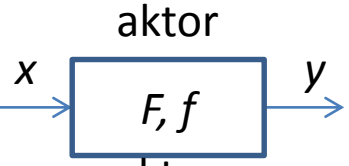
II. Adatfolyam modellek: A reakciók lehetnek szinkronok vagy aszinkronok. A reakciókra vonatkozó legfőbb kényszer az adatfüggőség, nem a szinkronitás.

Az aktorok bemeneti szekvenciát kimeneti szekvenciára képeznek le: → aktor függvény.

Az aktorok közötti kommunkáció eszköze a token/token sorozat: → tüzelő függvény/sorozat.

Példa: Skálázó:

$a \in \mathbb{R},\ V_x = \mathbb{R},$

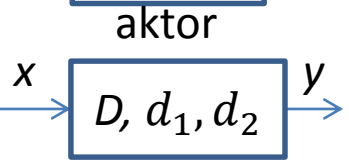


aktor függvény: $F(x_1, x_2, x_3, \dots) = (ax_1, ax_2, ax_3, \dots)$

tüzelő függvény: $f(x_1, x_2, x_3, \dots) = f(x_1) = (ax_1)$

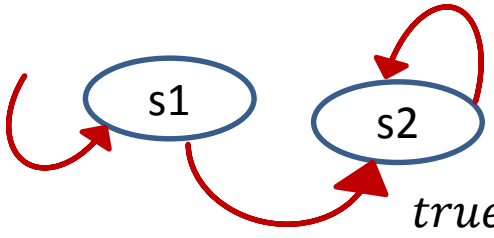
Példa: Késleltető:

$d \in \mathbb{R},\ V_x = \mathbb{R},$



aktor függvény: $D(x_1, x_2, x_3, \dots) = (d, x_1, x_2, x_3, \dots)$

tüzelő függvény: $d_1(s) = (d)$ $d_2(x_1, \dots) = (x_1)$

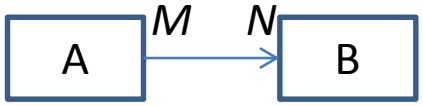


s: tetszőleges szekvencia.

Az $s1$ állapothoz a d_1, az s2 állapothoz a d_2 függvény tartozik.

Konkurens számítási modellek (MoC)

Szinkron adatfolyam modellek:



Egyensúlyi egyenlet: $q_A M = q_B N$

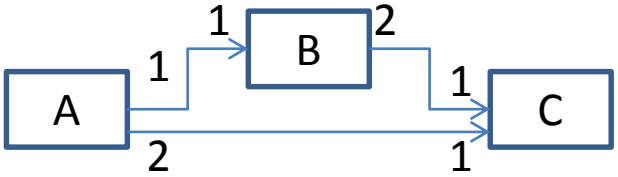
SDF modell: Tüzeléskor az A aktor *M* tokent hoz létre, a B aktor pedig *N*-et fogyaszt.

Példa: Ha *M*=2 és *N*=3, akkor $q_A = 3$ és $q_B = 2$ kielégíti az egyensúlyi egyenletet.

Egy lehetséges ütemezés: A,A,A,B,B, ami korlátlanul ismételhető a végtelenségig.

Kevesebb memóriát igénylő ütemezés: A,A,B,A,B. B azonnal tüzel, amint van elég tokenje.

Példa: Konzisztens SDF modell:

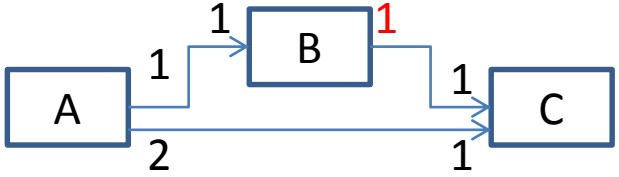


$q_A = q_B,\ 2q_B = q_C,\ 2q_A = q_C.$

Megoldás: $q_A = q_B = 1,$ $q_C = 2.$

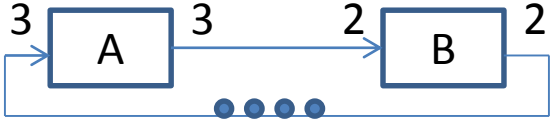
Ütemezés: A,B,C,C → véges memória, korlátlanul ismételhető.

Példa: Inkonzisztens SDF modell:



Holtpont: Visszacsatolt hálózatban kell **késleltető aktor.**

Példa:



$3q_A = 2q_B \quad q_A = 2$

$2q_B = 3q_A \quad q_B = 3$

Konzisztens SDF modell kezdeti token-ekkel:

Ütemezés: A,B,A,B,B → véges memória, korlátlanul …

Megjegyzések: 1. Három token nem lenne elegendő!

2. Létezik eljárás az egyensúlyi egyenletek megoldására.

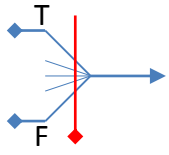
3. Létezik eljárás a korlátlan ismétlést lehetővé tevő ütemezésre, ha ilyen létezik. Ha nem létezik, ezt az eljárás maga bizonyítja.

Konkurens számítási modellek (MoC)

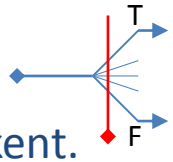
Dinamikus adatfolyam modellek (DDF): Az SDF aktorok nem tudnak feltételesen tüzelni.

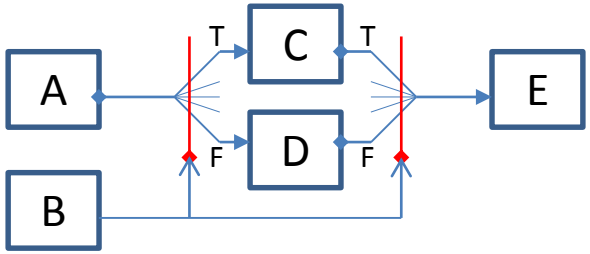
A DDF aktoroknak többféle tüzelési szabálya lehet, nem kell azonos számú tokent kiadniuk.

Select és **Switch** aktorok: (A késleltető aktor közvetlenül támogatott.)



1. Bináris token érkezik a piros portra.
2. ha False, akkor az F porton,
3. ha True, akkor a T porton várunk tokent.

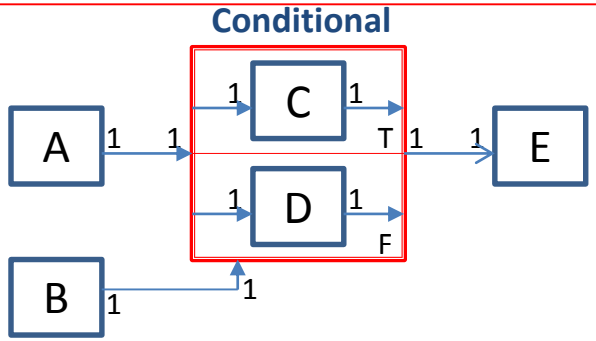




Mindez ekvivalens az **IF-THEN-ELSE** szerkezettel.

Probléma: nincs garancia, hogy véges számú token kell, és a holtpont elkerülhető. **(goto …)**

Strukturált adatfolyam modellek: **Conditional** aktor:



Magasabb rendű, belül paraméterezett aktor, ami kifelé **SDF** aktorként viselkedik. Az **NI LabVIEW** Ezt a modellt használja.

A **LabWIEW** ezzel oldja meg a **for**, a **do-while** szerkezeteket, valamint a **case** típusú feltételes (rész)modell végrehajtást.

Folyamathálózatok (PN): (Kahn)

Aszinkron adatfolyam modell, tüzelési függvények nélkül.

A PN aktor adat-token-t olvas, és adat-token-t ír. Az írás művelet nem blokkolódik, az üres buffer olvasása blokkol („token”-re vár).

A bufferek korlátossága és a holtpont elkerülés problémái itt is fennállnak.

Konkurens számítási modellek (MoC)

III. A számítások időzített modelljei: Az időnek eddig nem volt szerepe, csak szekvencia …

Idővezérelt rendszerek: TTA, TTP/C, TTP/A, FlexRay → hibatűrés → járműipari alkalmazások.

Hasonló az SR modellekhez: a globális óra koordinálja, de a végrehajtás nem pillanatszerű.
A számítások csak a következő óraütéskor jutnak érvényre, a modellek konstruktívak.

- A **Simulink**, a **Real-Time Workshop**, a **LabView** idővezérelt számítási modelleket (is) alkalmaz.

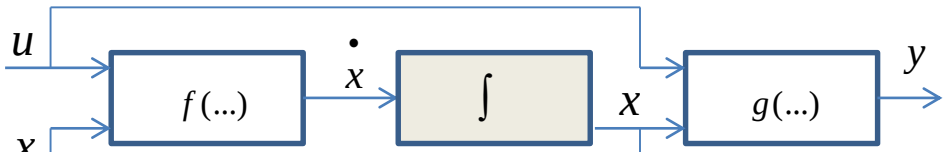
Diszkrét eseményű rendszerek (DE): Az eseményekhez időbélyeg tartozik (ez a „token”).

Az aktorok a bemeneti eseményekre időrendben reagálnak és kimenetük is időben rendezett.

Az események egy közös, rendezett listába kerülnek, feldolgozásuk ennek megfelelő sorrendű.
Az ütemezés szekvenciális, nem valós idejű. Sokféle variánsa létezik.

Folytonos idejű rendszerek: Leírás differenciálegyenlettel, pl.:

$$\dot{x} = f(x,u,t)$$
$$y = g(x,u,t)$$
$$x(t_0) = x_0$$



Euler megoldó:

$$x(h) = x(0) + hf(x(0),u(0);0)$$
$$x(2h) = x(h) + hf(x(h),u(h);h)$$
$$x(3h) = x(2h) + hf(x(2h),u(2h);2h)$$
$$\ldots$$

Felfogható egy aktor hálózatként.

Simulink, LabVIEW, MATRIXX, …

Milyen a jó MoC?

- Nincsenek felesleges kényszerek;
- Elég kötött, hogy
  - hasznos eredmény
  - hatékonyan …

Többféle MoC együttes használata …