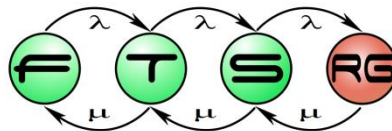


Állapot alapú modellezés

Budapesti Műszaki és Gazdaságtudományi Egyetem
Hibatűrő Rendszerek Kutatócsoport



Tartalom

Előismeretek

Állapotterek

Egyszerű (Mealy) állapotgépek

Összetett (Harel) állapotgépek

Kitekintés, szoftverek

Előismeretek

Állapottér

Mealy gépek

Harel gépek

Kitekintés

ELŐISMERETEK

Felépítési és viselkedési modellezés

■ Felépítési (*structural*)

- Statikus
- Rész és egész, összetevők
- Kapcsolatok, összeköttetések

A robotporszívó főbb részei a vezérlő, a hajtómű, és a porszívó.

■ Viselkedési (*behavioral*)

- Dinamikus
- Időbeli lefolyás
- Állapot, folyamat
- Reakciók a külvilágra

A hajtómű „jobbra” parancs hatására átvált „kanyarodás” üzemmódba



Viselkedésmodellek fő kérdései

- Mit „csinál” a rendszer?



Esemény alapú modell
Folyamat alapú modell
...

- Most „milyen”, és hogyan változik a rendszer?



Állapot alapú modell

Motiváló példa: virtuális billentyűzet

- Mi történik, ha megérintem a bal felső sarokban?
 - Q, q, 1 vagy =
 - Csak a „múlt” alapján eldönthető



Alapozás: diszkrét eseménysor

■ Esemény

- pillanatszerű változás (a rendszerben vagy ki/bemeneten)

■ Eseményfolyam

- Pl. input/output adatforrás

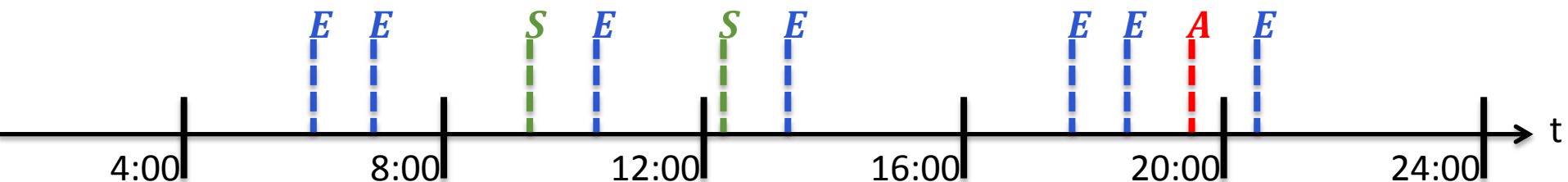
■ Eseménytér – {megengedett események}

- Beolvasható input értékek, kibocsátható output értékek

■ Pillanatszerű események sora, egyszerre ≤ 1

Eseményfolyam: telefon értesítési jelzései

Eseménytér: {*Email*, *SMS*, *Alacsony töltöttség*}



Eseményorientált programozás

The image shows a Chrome DevTools window with the 'Event Listeners' tab selected for the element `div#main-frame-error.interstitial-wrapper`. A red box highlights the `mousedown` event listener details. A red arrow points from this box to the 'Elements' tab, which shows the DOM tree with the same element selected.

Event Listener Details (highlighted):

- Event: `mousedown`
- Target: `document` (URL: `data:text/html,chrome...`)
- Handler: `Runner`
- isAttribute: `false`
- lineNumber: `1308`
- listenerBody: `"function (e) { ... ret...`
- node: `document`
- sourceName: `"data:text/html,chromeweb...`
- type: `"mousedown"`
- useCapture: `false`

DOM Tree (Elements tab):

```
<html>
  <head>
    <script>...</script>
    <script>...</script>
    <script>...</script>
    <script>...</script>
    <script>...</script>
  </head>
  <body id="t" i18n-values=...>
    <div id="main-frame-error" class="interstitial-wrapper" jstcache="0">
      <div id="sub-frame-error" jstcache="0">...</div>
      <div id="offline-resources" jstcache="0">...</div>
      <script jstcache="0">...</script>
      <script jstcache="0">...</script>
      <script jstcache="0">...</script>
      <script jstcache="0">...</script>
    </div>
  </body>
</html>
```


Előismeretek

Állapottér

Mealy gépek

Harel gépek

Kitekintés

ÁLLAPOTTEREK

Definíció: Állapottér

Az állapottér

- egymástól megkülönböztetett **rendszerállapotok** halmaza,
- amelynek minden időpontban pontosan egy eleme (a **pillanatnyi állapot**)

jellemzi a rendszert.

○ Állapottér példák

- {*hétfő, kedd, szerda, csütörtök, péntek, szombat, vasárnap*}
- mikró állapotai: {*teljes fokozat, kiolvasztó mód, kikapcsolva*}

○ Pillanatnyi állapot példák

- ma *szerda* van
- a mikró most éppen *kiolvasztó mód*ban üzemel



Az állapottér tulajdonságai

„pontosan egy eleme jellemzi a rendszert”

- Nem minden állapothalmaz lehet állapottér!

■ Teljesség

- Mindig legalább az egyik állapot fennáll
- Ellenpélda (nem alkalmas állapottérnek!)
 - $\{\text{hétfő, kedd, csütörtök, szombat}\}$ nem teljes

■ Kölcsönös kizárólagosság

- Egyszerre legfeljebb egy állapot állhat fent
- Ellenpéldák (nem alkalmasak állapottérnek!)
 - $\{\text{hétköznapi, hétvége, délután}\}$ nem kizárólagos (holott teljes)
 - mikró $\{\text{ajtaja tárva, kikapcsolva}\}$

Miért fontosak ezek a tulajdonságok?

- Szökőnap a Düsseldorf-i Reptéren

29. Februar 2016 | 13.46 Uhr

29. Februar

Schalttag legt Gepäckförderband am Flughafen lahm

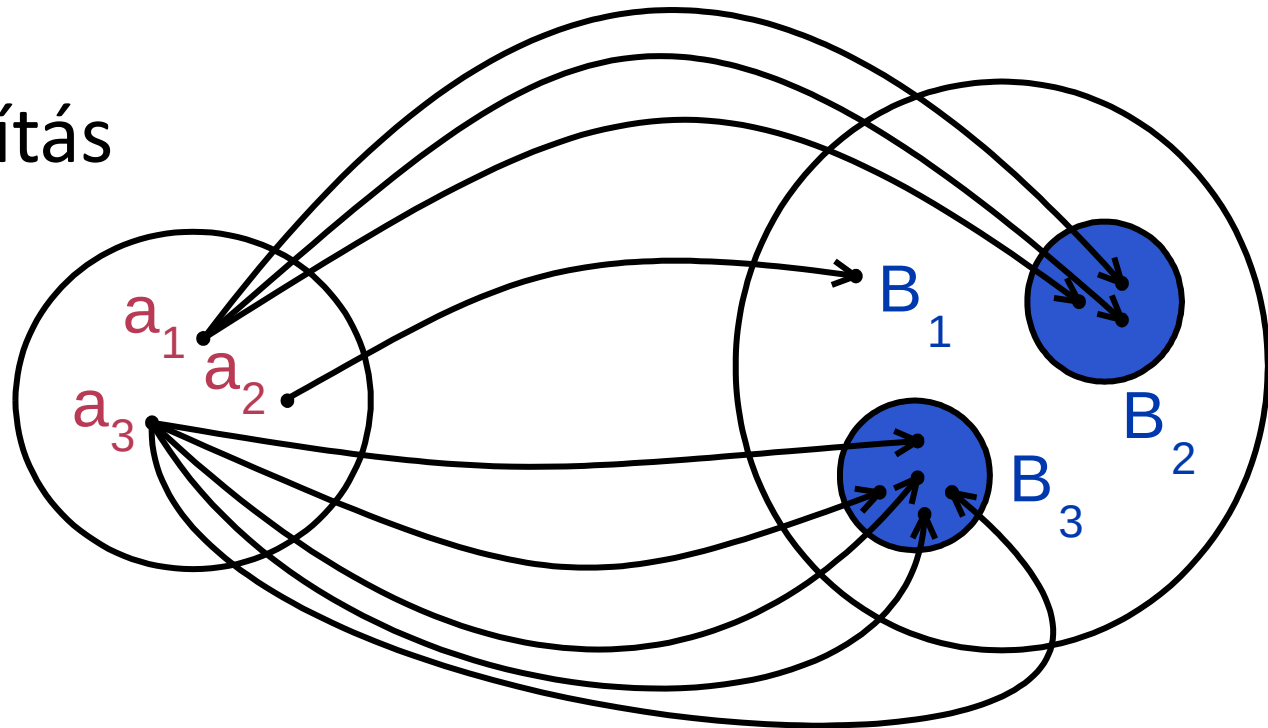


Definíció: Állapotfinomítás és -absztrakció

Az **állapotfinomítás** ill. **állapotabsztrakció** az állapottéren mint halmazon végzett **halmazfinomítás** ill. **halmazabsztrakció**, melynek eredménye egy újabb állapottér.

- (Másfajta absztrakciókkal is találkozunk majd...)

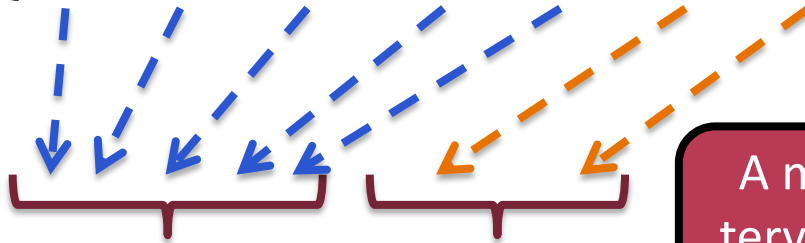
Ismétlés:
halmazfinomítás



Állapotfinomítás, állapotabsztrakció

- Állapotfinomítás/absztrakció: az állapottéren végzett halmazfinomítás/absztrakció

- {*Hé*, *Ke*, *Sze*, *Csü*, *Pé*, *Szo*, *Va*}

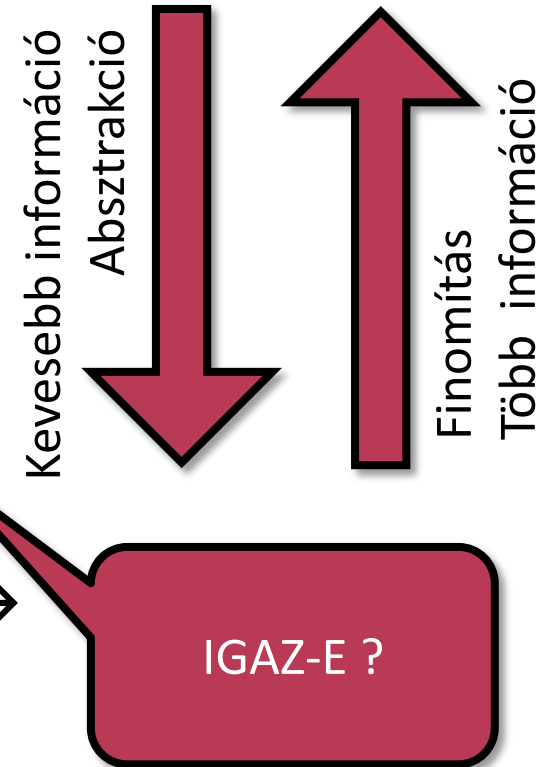


- {*hétköznap*, *hétvége*}

A menetrend tervezőjét csak ennyi érdekli

- Mikro állapotabsztrakciója

- {*teljes fokozat*, *olvasztó*, *kikapcsolva*} →
 {*bekapcsolva*, *kikapcsolva*}
- „*kikapcsolva*” változatlan



Finomítás motivációja

- Állapotfinomítás: miért?
 - Tervezés előrehaladása, több megvalósítási részlet
 - Pl. erősáramú tervezéshez fontos a mikró fokozata
 - Specializáció / kiegészítés
 - Pl. egy fejlettebb mikró már időzítőt is tartalmaz...
 - Több rendszer együttes vizsgálata (ld. később)
- Több információ, több tudás
 - Ez vajon mindig jó?

Absztrakció motivációja

- Állapotabsztrakció: miért?
 - Akkor hasznos, ha az absztrakt állapotok
 - „egységesek” majdnem ekvivalensek
 - Valamilyen szempontból egyformák az összevont állapotok
 - Ld. „helyettesítési tulajdonságú partíció” → Digit
 - Bizonyos feladatokra kevesebb információ is elég
 - Kisebb, egyszerűbb állapottérrel könnyebb tervezni
 - Állapot tárolása, feldolgozása könnyebb
 - Elrejtett részletek szabadon változtathatóak
 - Szükséges eset: **állapotmentes** modell ($|S| = 1$)
 - Néha csak kevesebb információt szabad felfedni
- Gyakori formája: **dekompozíció** (ld. később)

Partícionálás több szempont szerint

- Egy rendszer – akár több helyes állapottér
- Pl.: a mikró két külön állapottere
 - üzemteljesítmény szerint
 - $\{teljes\ fokozat, olvasztó\ mód, kikapcsolva\}$
 - az ajtó nyitottsága szerint
 - $\{nyitva, zárva\}$
 - nem teljesen függetlenek: ha nyitva, akkor kikapcsolva
- részrendszer állapottere $\rightarrow$ a teljes rendszerállapot ismerete nélkül eldönthető, melyik áll fenn



Állapotterek (direkt) szorzata

- Két állapottér együttes figyelembevételé
 - $S_1 = \{\text{teljes fokozat, olvasztó mód, kikapcsolva}\}$
 - $S_2 = \{\text{nyitva, zárva}\}$

$S_1 \times S_2$	<i>nyitva</i>	<i>zárva</i>
<i>teljes</i>	<i>teljes és nyitva</i>	<i>teljes és zárva</i>
<i>olvasztó</i>	<i>olvasztó és nyitva</i>	<i>olvasztó és zárva</i>
<i>kikapcsolva</i>	<i>kikapcsolva és nyitva</i>	<i>kikapcsolva és zárva</i>

állapotabsztrakció

állapotabsztrakció
(vetítés)

Észrevétel: $|S_1 \times S_2| = |S_1| \cdot |S_2|$

Definíció: Állapotterek (direkt) szorzata

Az állapotterek **direkt szorzata**

- komponens állapottereken végzett **kompozíciós művelet**,
- melynek **eredménye** egy újabb állapottér (**szorzat állapottér**),
 - amely a komponens állapotterek mint halmazok Descartes-szorzataként áll elő

A szorzat állapottérben

- a komponens állapotterek minden állapot-kombinációjának
- egy-egy összetett állapot (**állapotvektor**)

felel meg.

$$\begin{aligned} &\{AM, PM\} \times \{1h..12h\} \times \{0m..59m\} \\ &\quad \cup \\ &\langle PM, 12h, 08m \rangle \end{aligned}$$



Definíció: Állapottér vetítése komponensre

A komponens(ek)re történő **vetítés**

- egy **állapotabsztrakciós művelet**, amely
- a szorzat állapottérből
 - egy vagy több komponenst tart meg,
 - a többit elhanyagolja.

$$\{AM, PM\} \times \{1h..12h\} \times \{0m..59m\}$$

$\Downarrow$

$$\langle PM, 12h, 08m \rangle$$



$$\langle PM, 12h \rangle$$

(Gondolkodtató feladat:
mi köze van a reláció vetítéséhez?)



Állapotváltozók finomított kompozíciója

- Nem független állapotváltozók
 - Nem minden kombináció fordul elő ténylegesen
 - A szorzatnál finomabb kompozit állapottér



Még egyszer a finomításról

- „A szorzatnál ***finomabb*** kompozit állapottér”
 - ... mivel két kompozit állapot előfordulását kizártuk
 - Itt a finomított állapottérnek van ***kevesebb*** eleme!
 - Ez tehát *nem* állapotfinomítás, ott *nőtt* az állapotok száma

Tanulság:

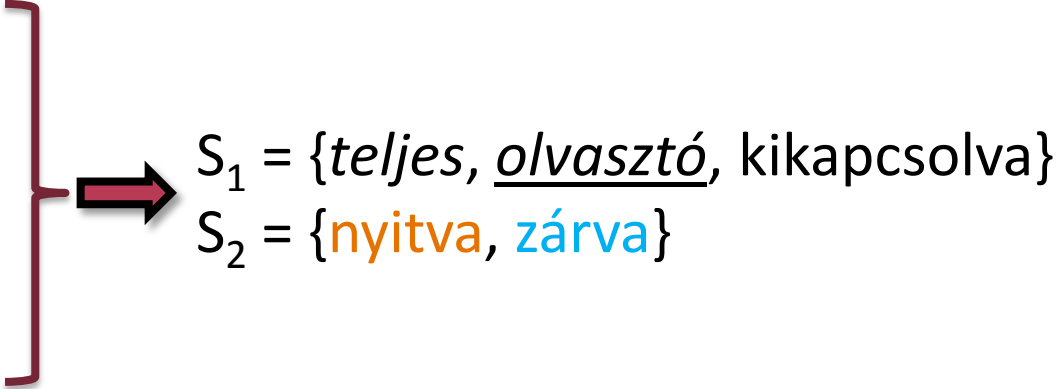
finomításoktól nőhet is, csökkenhet is az állapottér mérete

A lényeg, hogy
**többet tudunk
a rendszerről**

Avagy:
**kevesebb
rendszer illik
a modellre**

$S \subseteq S_1 \times S_2$	<i>nyitva</i>	<i>zárva</i>
<i>teljes</i>	<i>teljes és nyitva</i>	<i>teljes és zárva</i>
<i>olvasztó</i>	<i>olvasztó és nyitva</i>	<i>olvasztó és zárva</i>
<i>kikapcsolva</i>	<i>kikapcsolva és nyitva</i>	<i>kikapcsolva és zárva</i>

Dekompozíció állapotváltozókra

- Dekompozíció: szorzat / kompozíció megfordítása
 - kikapcsolva és nyitva
 - kikapcsolva és zárva
 - olvasztó és zárva
 - teljes és zárva
$$S_1 = \{teljes, \underline{olvasztó}, kikapcsolva\}$$
$$S_2 = \{nyitva, zárva\}$$
- A vetített állapotváltozók absztrakciók
- Miért dekomponálunk?
 - Állapotváltozók külön kezelhetőek
 - Állapotváltozók külön tárolhatóak

Kitekintés: szakmai példák

- Hol jön elő az állapot alapú modellezés az IT-ban?
- Közösségi háló – Jancsi és Juliska ismeretsége
 - (ettől függ néhány funkció, pl. milyen képek látszanak)
 - Állapotváltozók:
 - Jancsi bejelölte-e Juliskát
 - Vice versa

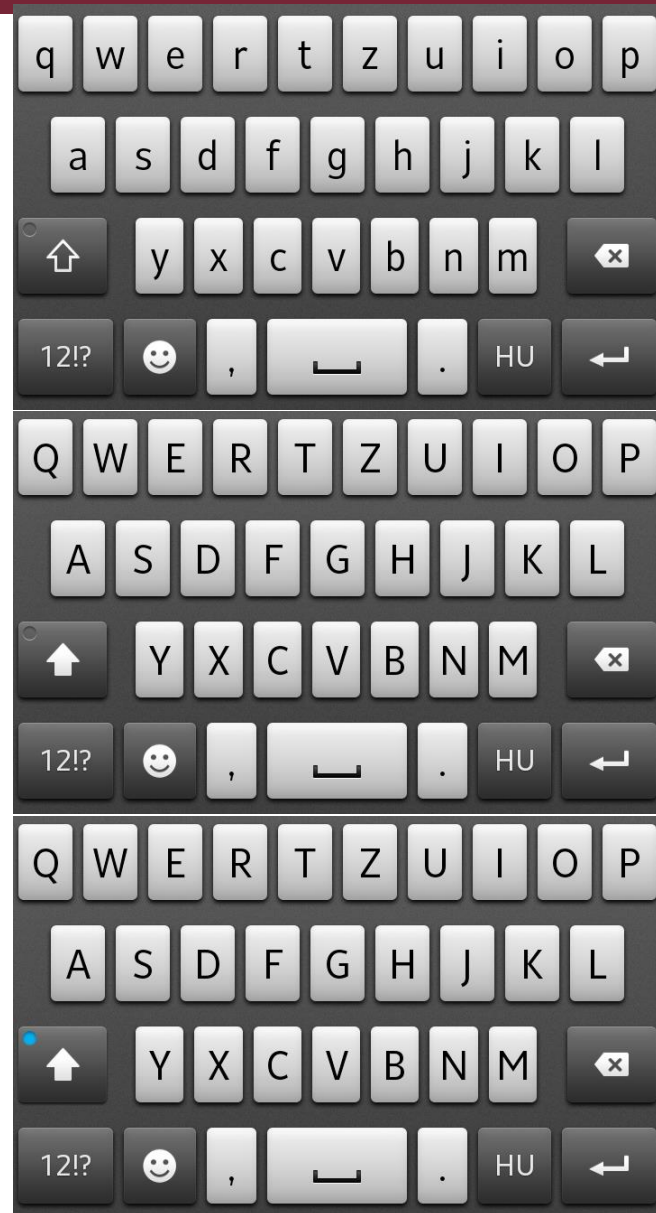
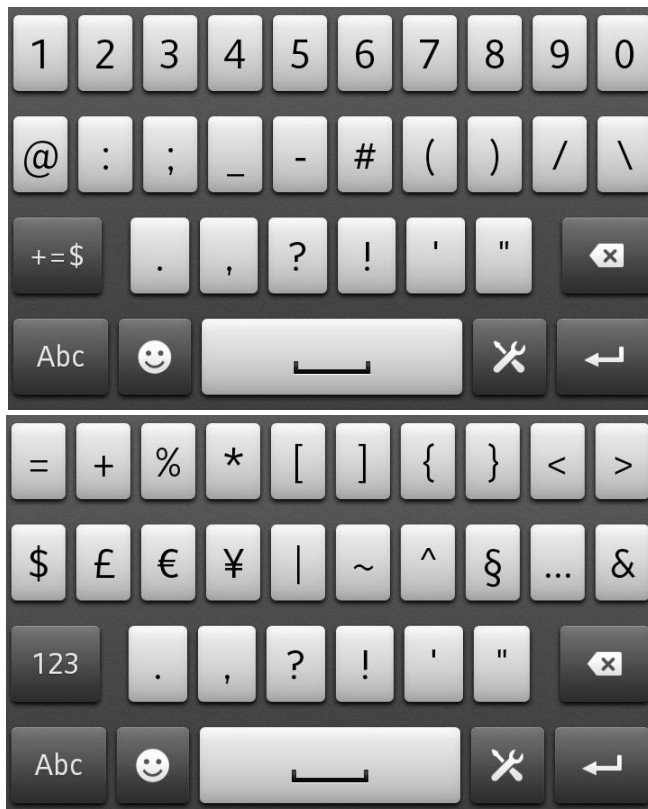
Tárolandó:

- Memóriában
- Adatbázisban (hosszabb távra)

$S_1 \times S_2$		
	nem ismerősök	Jancsi bejelölte Juliskát
	Juliska bejelölte Jancsit	ismerősök

Kitekintés: szakmai példák

- Virtuális billentyűzet érintőképernyőre
 - állapotváltók?



Kitekintés: szakmai példák

- Programozás: hogy tároljuk az állapotot?
 - Megfelelő értékkészletű változó (objektum mező, stb.)

```
enum VirtualKeyboardState {  
    LOWER_CASE,  
    UPPER_CASE_ONCE,  
    UPPER_CASE_LOCK,  
    NUMBERS_COMMON_SYMBOLS,  
    RARE_SYMBOLS  
}  
// ...  
VirtualKeyboardState keyboardState;
```



- Kiegészítés: emlékezzünk a SHIFT állására!
 - Az alfanumerikus mód az elhagyott SHIFT állással tér vissza

Kitekintés: szakmai példák

- Programozás: hogy tároljuk az állapotot?
 - Kiegészítés: emlékezzünk a SHIFT állapotára!

```
enum VirtualKeyboardStateWithMemory {  
    LOWER_CASE,  
    UPPER_CASE_ONCE,  
    UPPER_CASE_LOCK,  
    NUMBERS_COMMON_SYMBOLS_WITH_LOWER_CASE,  
    NUMBERS_COMMON_SYMBOLS_WITH_UPPER_CASE_ONCE,  
    NUMBERS_COMMON_SYMBOLS_WITH_UPPER_CASE_LOCK,  
    RARE_SYMBOLS_WITH_LOWER_CASE,  
    RARE_SYMBOLS_WITH_UPPER_CASE_ONCE,  
    RARE_SYMBOLS_WITH_UPPER_CASE_LOCK  
}  
// ...  
VirtualKeyboardStateWithMemory keyboardStateWithMemory;
```

- Állapottér-robbanás jelensége

Kitekintés: szakmai példák

- Programozás: hogy tároljuk az állapotot?
 - Tömör megoldás: több állapotváltozóval

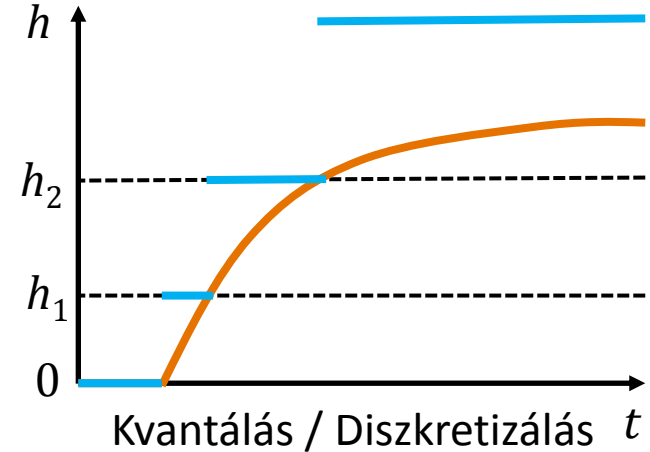
```
enum VirtualKeyboardFacet {  
    ALPHABETIC,  
    NUMBERS_COMMON_SYMBOLS,  
    RARE_SYMBOLS  
}  
  
enum CapsState {  
    LOWER_CASE,  
    UPPER_CASE_ONCE,  
    UPPER_CASE_LOCK  
}  
  
// ...  
VirtualKeyboardFacet keyboardFacet;  
CapsState capsState;
```

Kitekintés: más mérnöki diszciplínák

■ Potenciálisan végtelen (akár kontinuum) állapottér

○ Pl. a repülő állapotváltozói

- $v \in \mathbb{R}$ sebesség
- $h \in \mathbb{R}$ magasság
- $\alpha \in [-\pi/2, \pi/2]$ emelkedési szög



■ Ezzel szemben tipikus IT rendszermodellekben

- **Diszkrét állapotok** (nincs köztük folytonos átmenet)
- Gyakran **véges állapottér** (ellenpélda: számláló $\in \mathbb{N}$)

Előismeretek

Állapottér

Mealy gépek

Harel gépek

Kitekintés

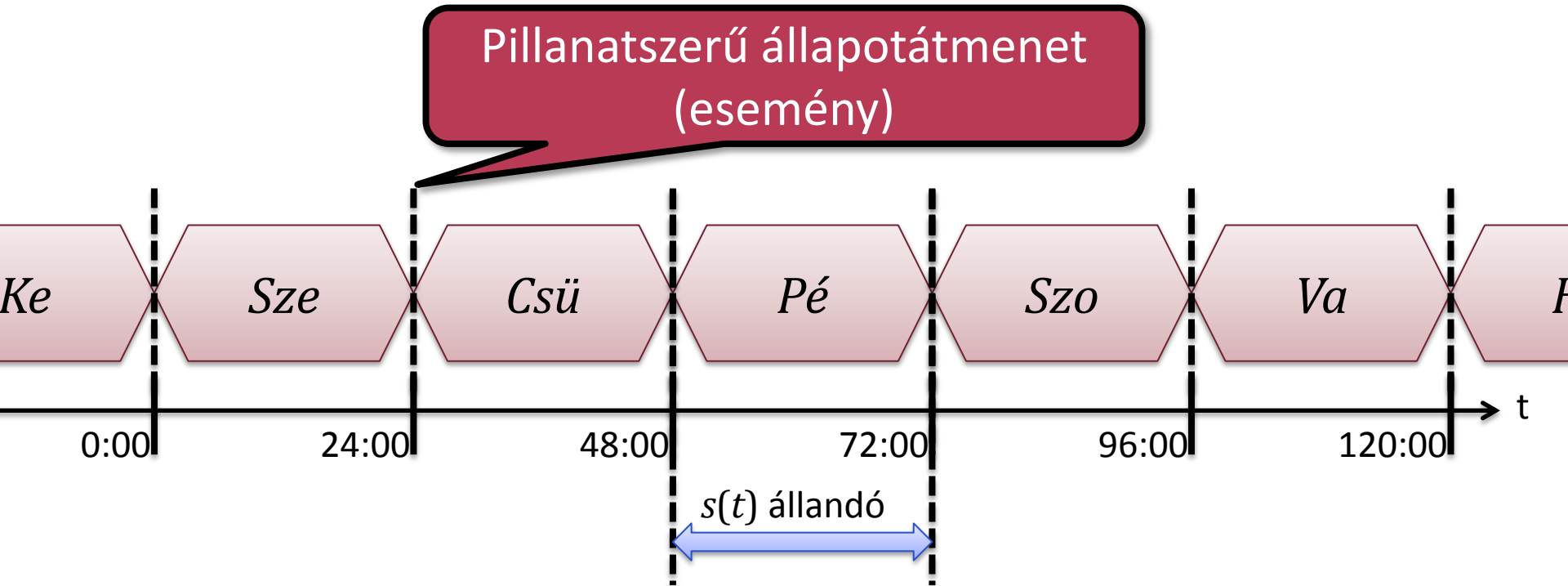
EGYSZERŰ (MEALY) ÁLLAPOTGÉPEK

Kitekintés: más mérnöki diszciplínák

- Potenciálisan végtelen (akár kontinuum) állapottér
 - Pl. a repülő állapotváltozói
 - $v \in \mathbb{R}$ sebesség
 - $h \in \mathbb{R}$ magasság
 - $\alpha \in [-\pi/2, \pi/2]$ emelkedési szög
 - Az állapot időbeli változása lehet folytonos
 - Pl. a repülő emelkedése: $\partial h / \partial t = v \sin \alpha$
- Ezzel szemben tipikus IT rendszermodellekben
 - **Diszkrét állapotok** (nincs köztük folytonos átmenet)
 - Gyakran **véges állapottér** (ellenpélda: számláló $\in \mathbb{N}$)
 - Pillanatszerű **állapotátmenetek**, köztük állandó állapot

Állapotátmenetek

- Állapottér: S
 - Pl. $S = \{Hé, Ke, Sze, Csü, Pé, Szo, Va\}$
- $s(t) \in S$
 - A pillanatnyi állapot az idő függvényeként

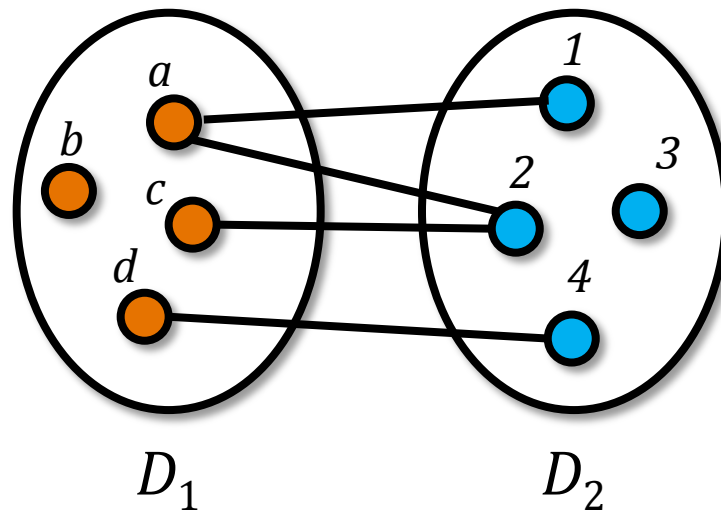


Ismétlés: (bináris) reláció

■ Bináris reláció:

- két halmaz Descartes-szorzatának a részhalmaza

- $R \subseteq D_1 \times D_2$



$$R = \{(a, 1), (a, 2), (c, 2), (d, 4)\}$$

Állapotátmenetek

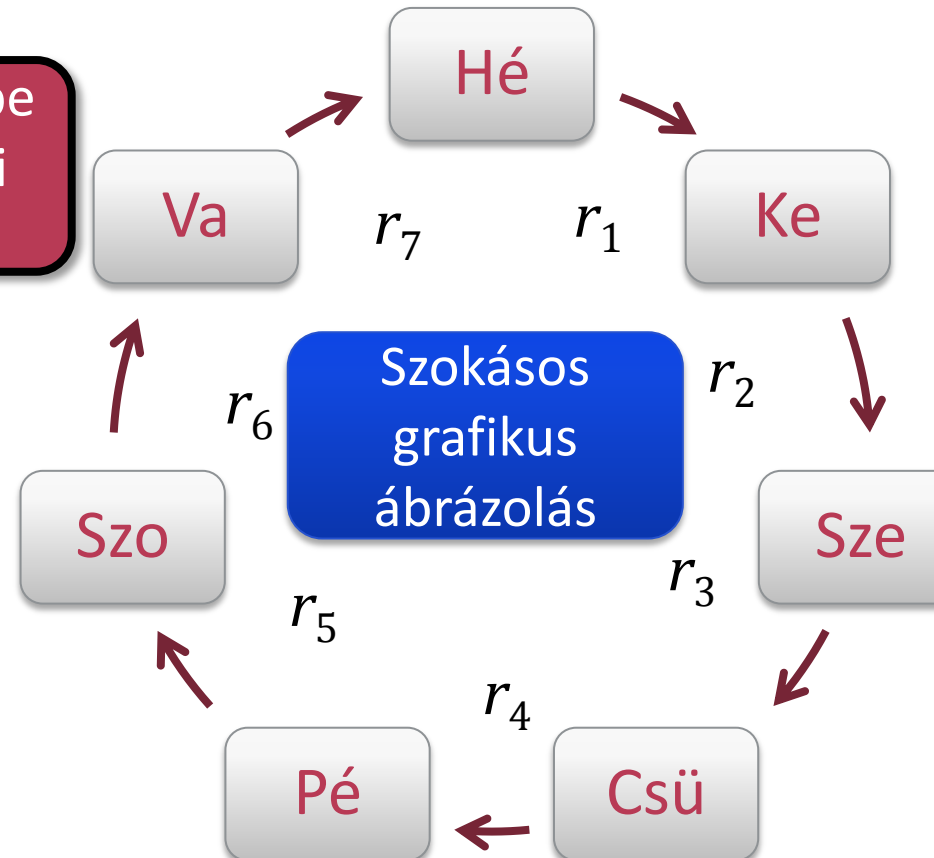
- Mely állapotokat mely állapotok követhetnek?

- Állapottér $S = \{Hé, Ke, Sze, Csü, Pé, Szo, Va\}$

- Eseménytér: **állapotátmeneti reláció** $R \subseteq S \times S$

- Állapotátmeneti szabályok $r \in R$
- $r_1 = (Hé, Ke)$
 - $r_2 = (Ke, Sze)$
 - $r_3 = (Sze, Csü)$
 - $r_4 = (Csü, Pé)$
 - $r_5 = (Pé, Szo)$
 - $r_6 = (Szo, Va)$
 - $r_7 = (Va, Hé)$

Csü-ből Pé-be
át tud lépni
a rendszer



- Más néven: **állapotgráf**

Észrevételek az állapotgráfról

■ Lehetséges, hogy...

- ... teljes gráf $\rightarrow$ minden átmenet engedélyezett
 - Pl. $S_{\text{kismacska}} = \{\text{alszik}, \text{játszik}, \text{iszik}\}$
- ... nem minden állapotból érhető el az összes többi
 - Pl. $S_{\text{pohár}} = \{\text{üres}, \text{teli}, \text{törött}\} \rightarrow$ nincs $\text{törött} \leadsto \text{üres}$ út
- ... bizonyos állapotoknak több rákövetkezője is van

kikapcsolva és
nyitva

kikapcsolva és
zárva

olvasztó és zárva

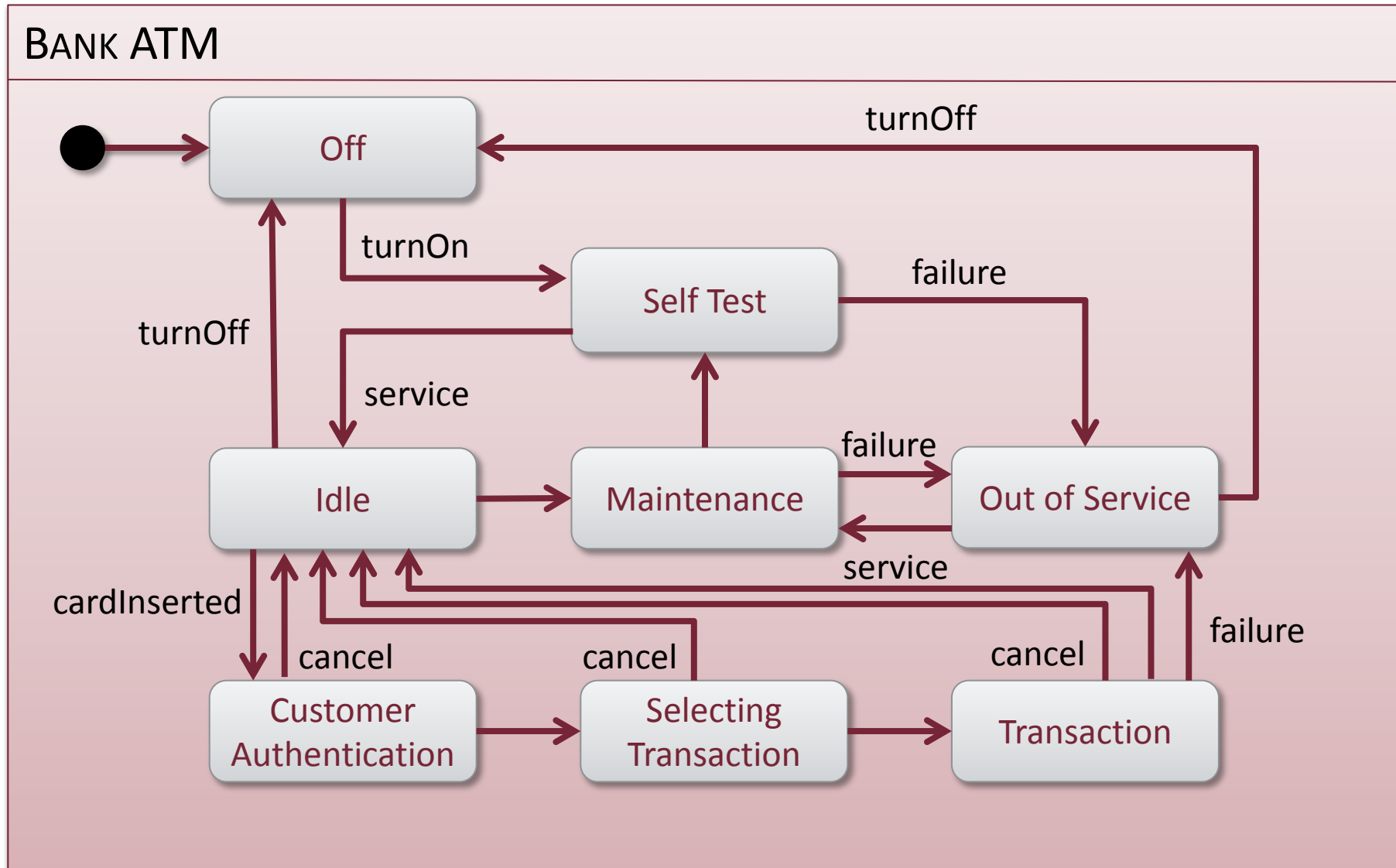
Nemdeterminizmus

○ Lehetséges eredetei:

- A modellezett rendszer működése
- Modellalkotás során bevezetett absztrakció

Pl. figyelembe nem
vett belső változó,
vezérlő input

Állapotgráf példa: ATM



Állapotátmenet címkézése eseménnyel

- Átmeneti szabály címkéje

- Pillanatszerű esemény

- Az átmenet csak az eseménnyel együtt következhet be



- Különféle értelmezés: lehet az esemény...

- ... az állapotátmenet következménye (posztkondíció/utófeltétel)



- ... az állapotátmenet oka (prekondíció/előfeltétel)



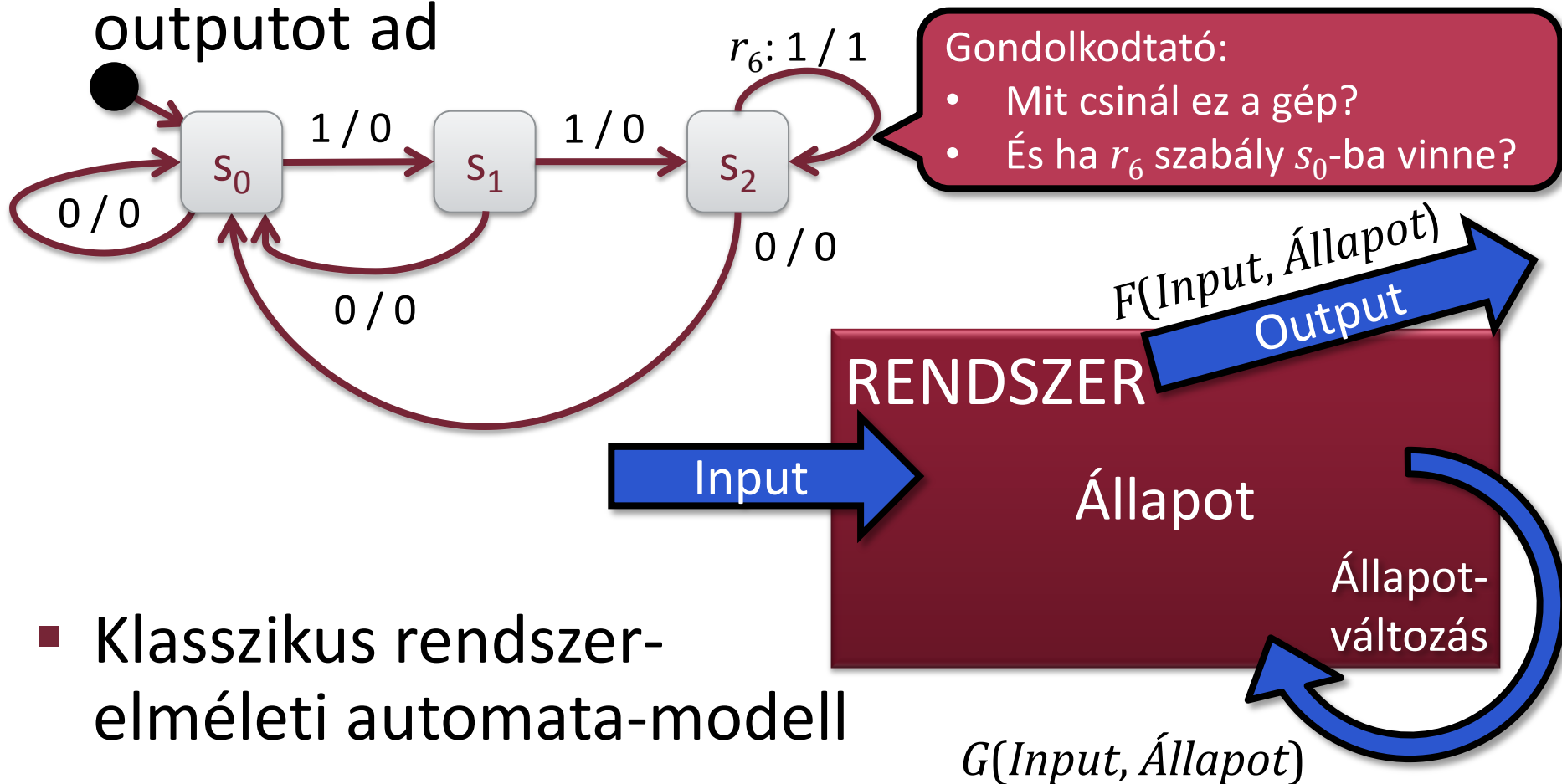
- Egy szabály címkézhető több eseménnyel is

- input beolvasás / output kiírás



Emlékeztető: Mealy véges automata

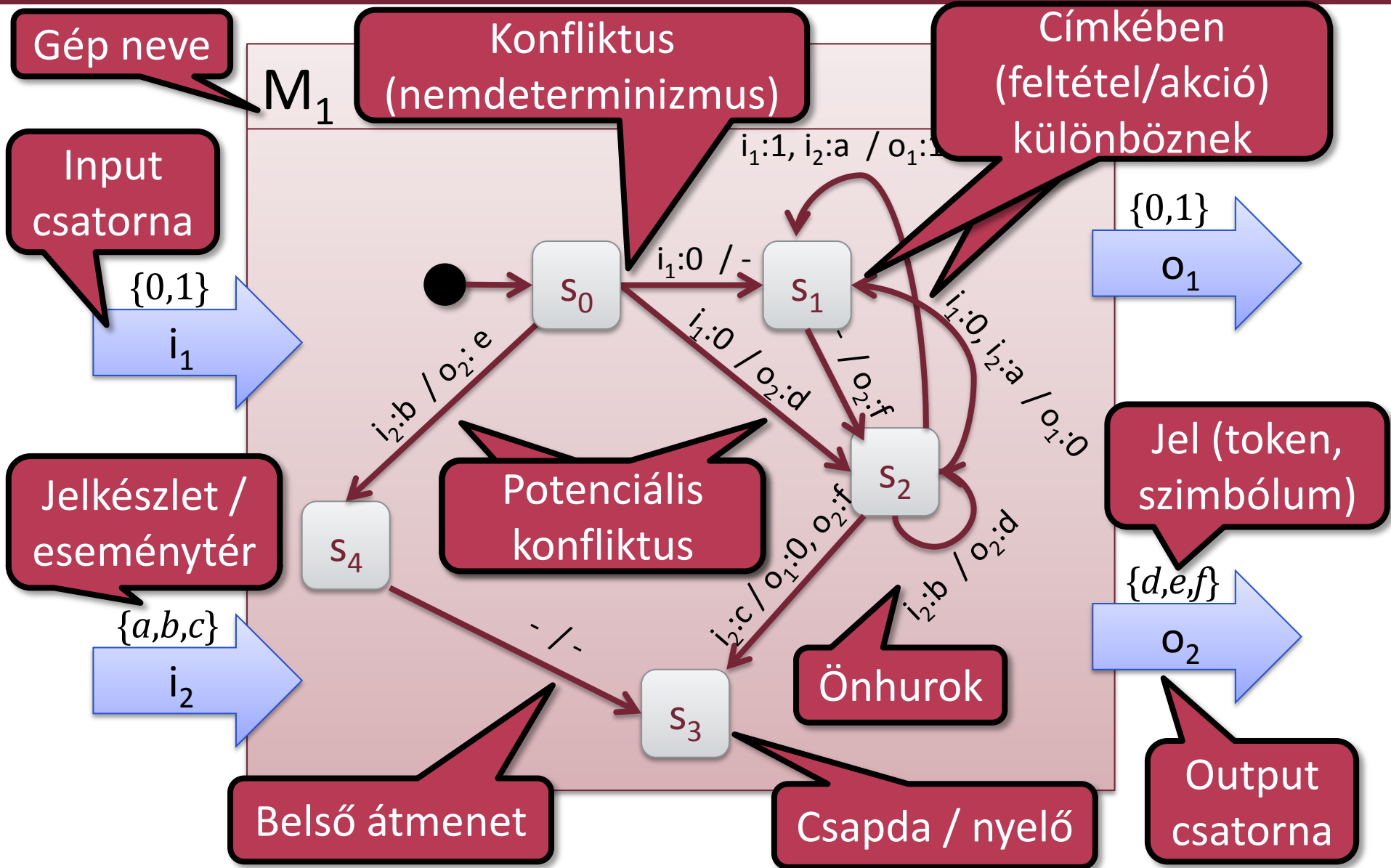
- Kijelölt kezdőállapot $\rightarrow s_0 = s(t=0)$
- Minden lépés determinisztikus, inputot olvas és outputot ad



A Mealy-gép kiterjesztései

- Nemdeterminisztikus modell (vs. input)
- Input olvasás nélküli (**spontán**) lépés
 - Belső, nem modellezett esemény hatására
 - Pl. mikro elkészül, leáll → időzítőt nem modellezzük
- Több output csatorna (külön eseményfolyam!)
 - Külön-külön jelkészlettel
 - A szabály egy részhalmazra küld ki oda illő jeleket
- Több input csatorna (külön eseményfolyam!)
 - A szabály egy részhalmazról olvas jeleket
 - Ebből is adódhat nemdeterminizmus

Kiterjesztett állapotgép

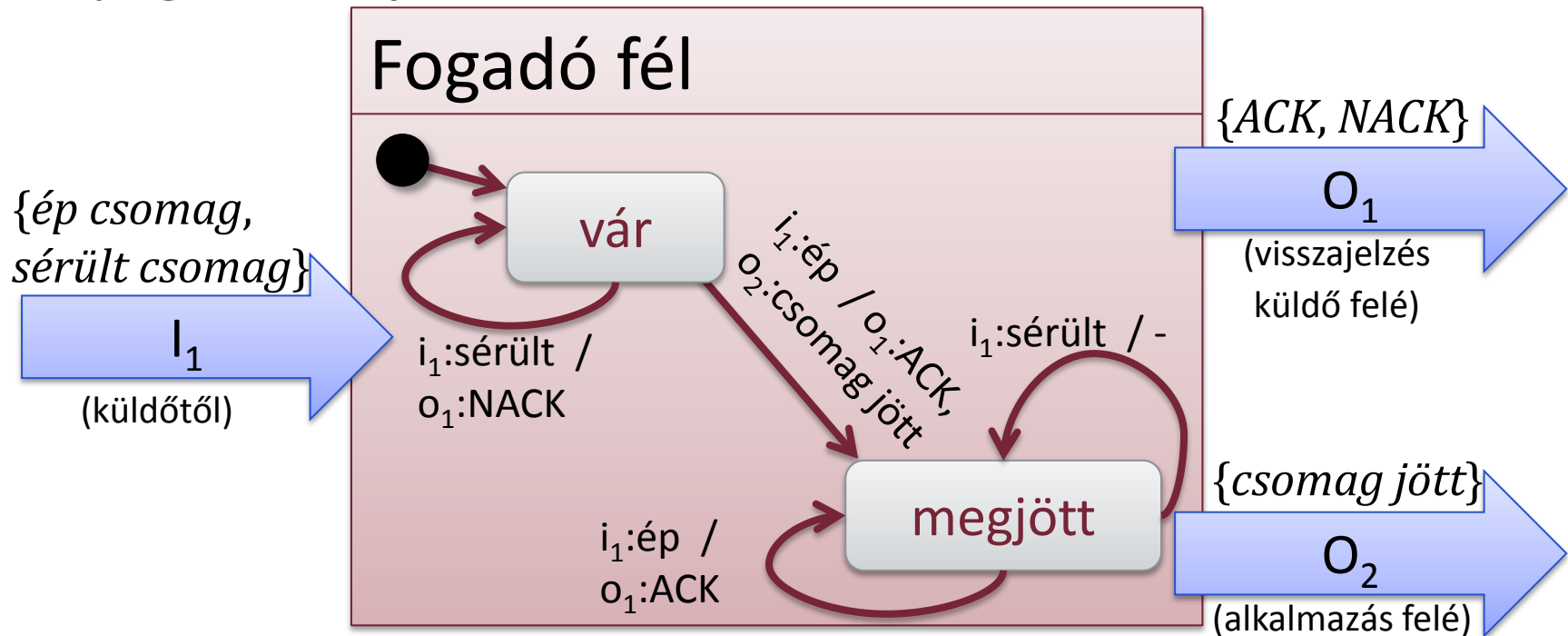


Specifikálatlan input kezelése

- Minden állapotban minden inputra kell szabály?
 - Ha nincs szabály → esemény nem történhet meg
 - Matematikailag így van, de **input**okra nem reális feltételezés
 - Mi van, ha a gyakorlatban mégis megtörténik?
 - Ha nincs szabály → láthatatlan hurokél
 - „Minden egyéb” input beolvasva, figyelmen kívül hagyva
 - Ha nincs szabály → érvénytelen a modell
 - Mindenképp kell “visszajelzés” az eseményre
 - Pl. kritikus beágyazott rendszereknél
 - Az is érvénytelen, ha több szabály van (determinizmus kell)
- Ha mindig van szabály → **teljesen specifikált**

Kitekintés: szakmai példák

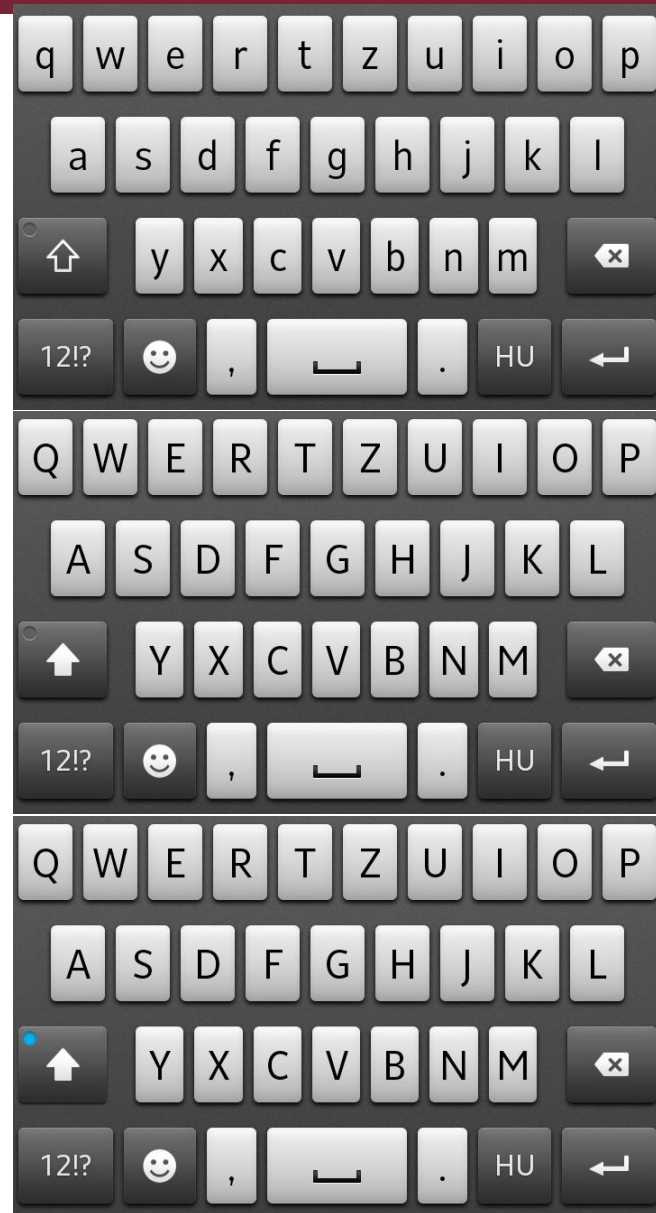
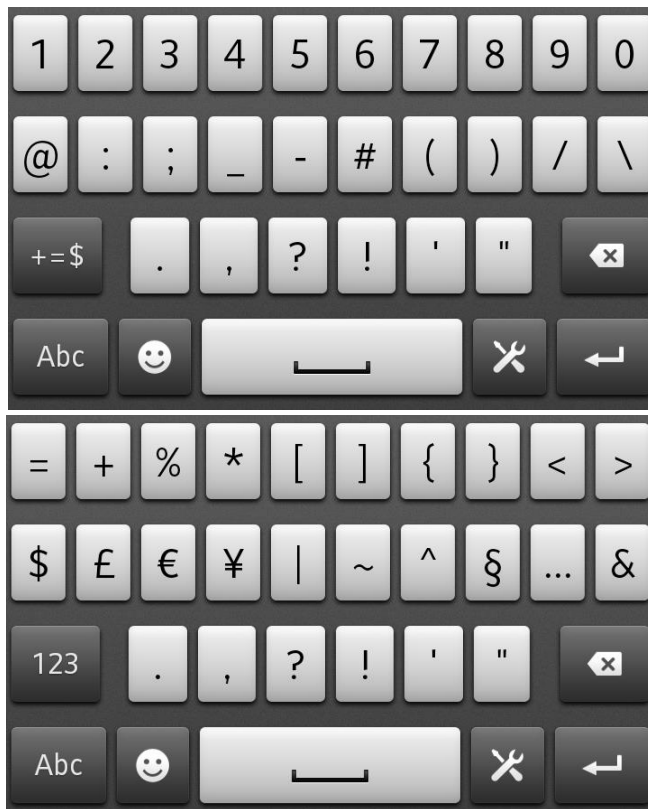
- Csomag alapú adatátviteli protokoll
 - Csomagok elveszhetnek, megsérülhetnek
 - Nyugtázás, újraküldés



- (Bővebben Id. Kommunikációs hálózatok 1. tárgy)

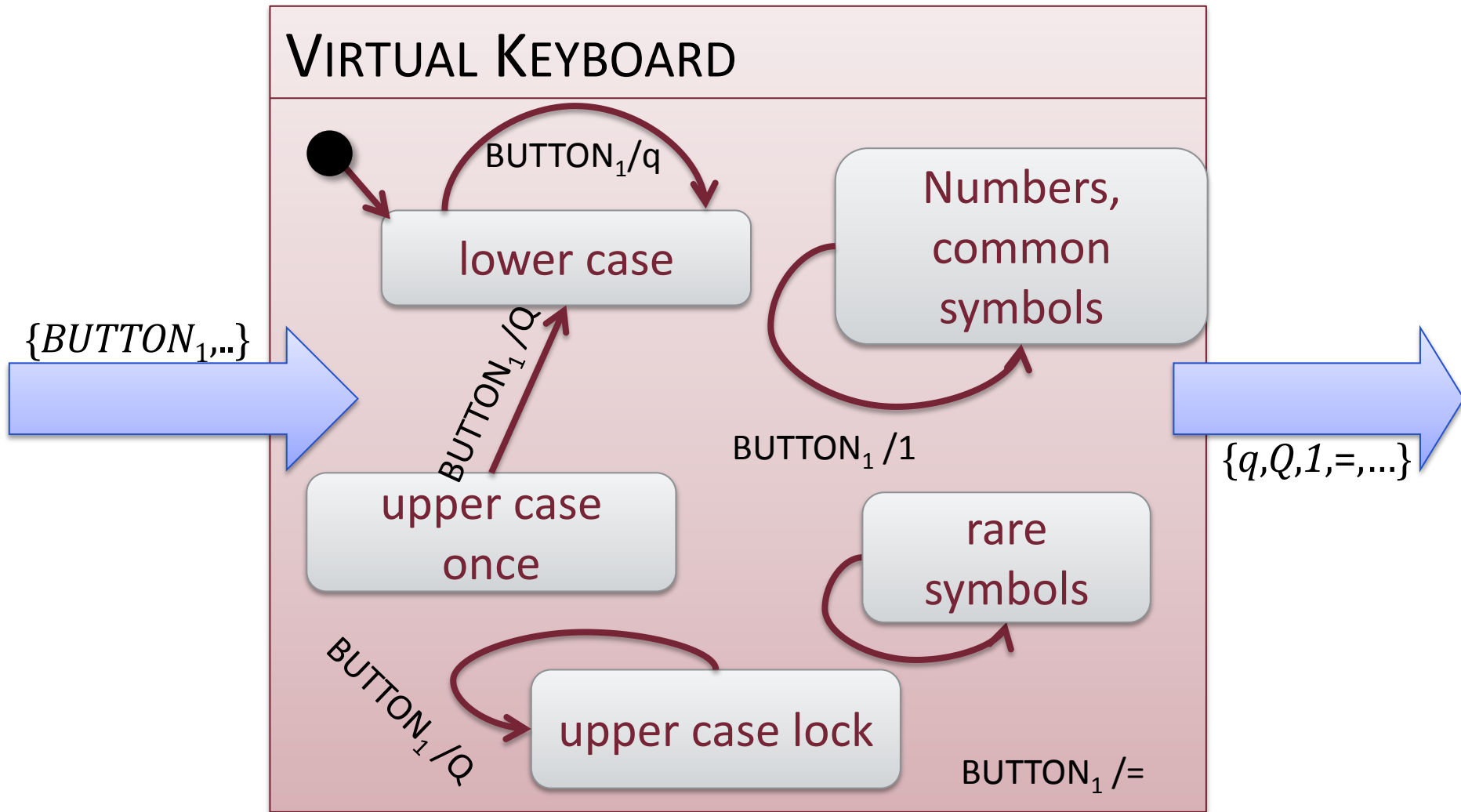
Emlékeztető

- Virtuális billentyűzet érintőképernyőre



Kitekintés: szakmai példák

- Virtuális billentyűzet (részleges) állapotgépe



Kitekintés: szakmai példák

■ Programozás: állapotgép megvalósítása

→ Prog1 6. ea

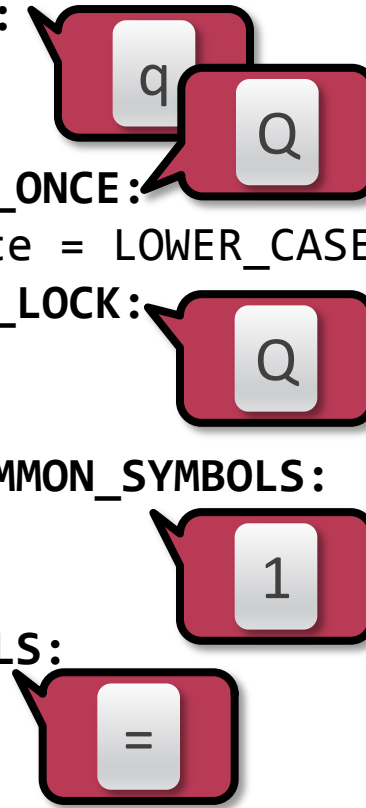
○ Elágazási feltétel:

- állapotváltozók és
- input alapján

○ Minden ágon:

- output kiadás (ha van)
- állapotváltás (ha kell)

```
void handleKey(KeyCode input) {  
    switch(input) {  
        case BUTTON_1:  
            switch(keyboardState) {  
                case LOWER_CASE:  
                    emit('q');  
                    break;  
                case UPPER_CASE_ONCE:  
                    keyboardState = LOWER_CASE;  
                case UPPER_CASE_LOCK:  
                    emit('Q');  
                    break;  
                case NUMBERS_COMMON_SYMBOLS:  
                    emit('1');  
                    break;  
                case RARE_SYMBOLS:  
                    emit('=');  
            }  
            break;  
        case SWITCH_1:  
            // ...  
    }  
}
```



Kitekintés: szakmai példák

- Ha az (állapotgép)modell...
 - ... kellően részletes (determinisztikus), és
 - ... olyan formában van, amit fel lehet dolgozni
 - Ld. szakterület-specifikus célnyelvek (pl. protokolltervezésre)
 - Ld. szabványos modellezési formátumok (pl. UML)
- ... akkor automatikusan programkóddá fordítható
 - Pl. kódgenerálás kommunikációs protokollokhoz
 - Pl. beágyazott vezérlőeszközök modell alapú fejlesztése
- ... vagy általános interpreterrel értelmezhető
 - Pl. IT rendszerüzemeltetés automatizálása

Előismeretek

Állapottér

Mealy gépek

Harel gépek

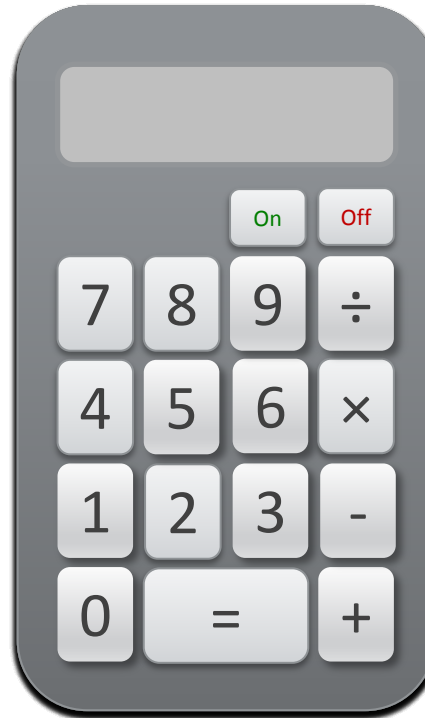
Kitekintés

ÖSSZETETT (HAREL) ÁLLAPOTGÉPEK

Statechart nyelvek

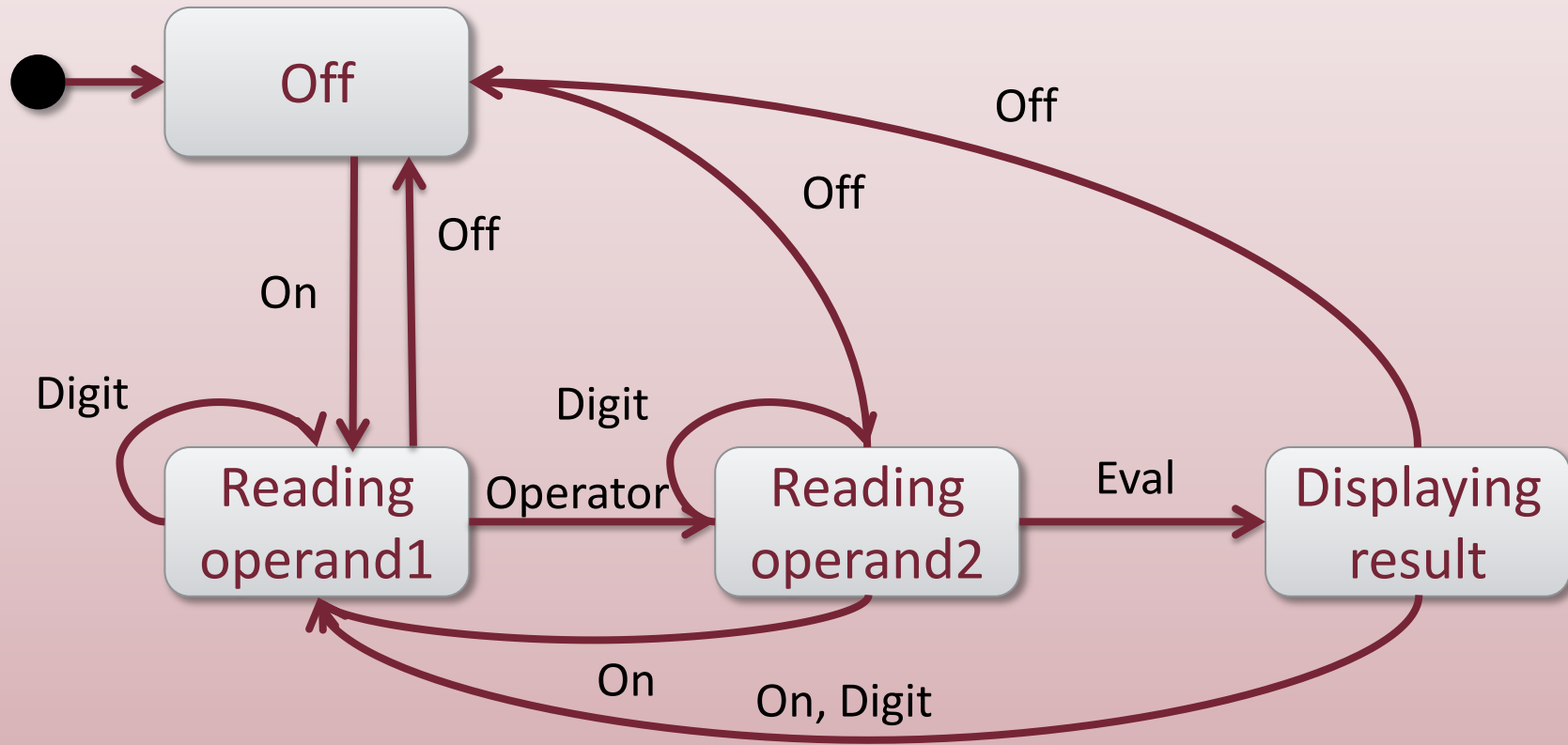
- **Statechart** (vagy Harel-féle összetett állapotgép)
 - Mealy-féle egyszerű állapotgép +
 - Állapothierarchia
 - Ortogonalitás
 - Változók
 - Pszeudoállapotok
 - ...
- Pl.
 - Yakindu (HF)
 - UML (SzoftTech)

Számológép



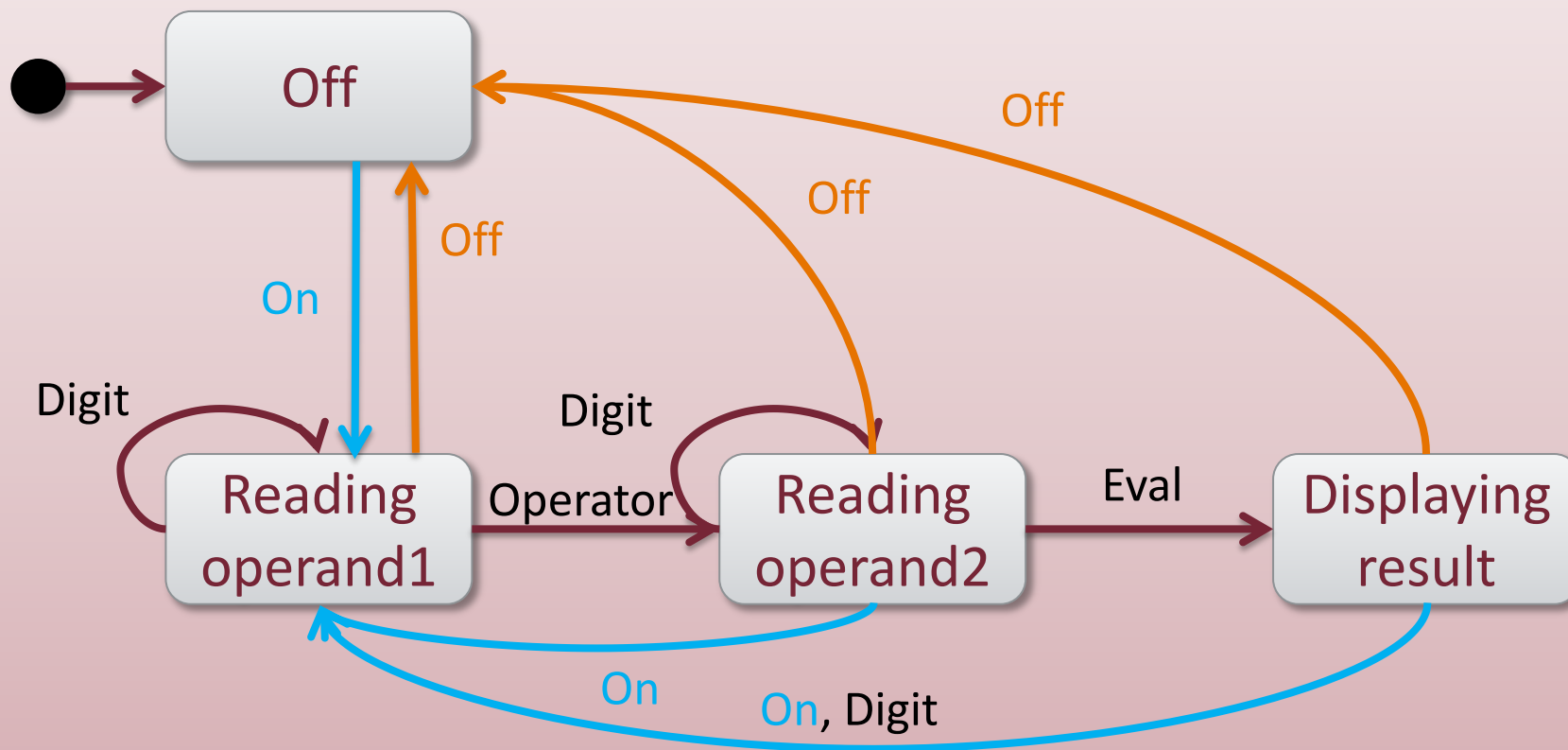
Állapothierarchia

CALCULATOR



Állapothierarchia

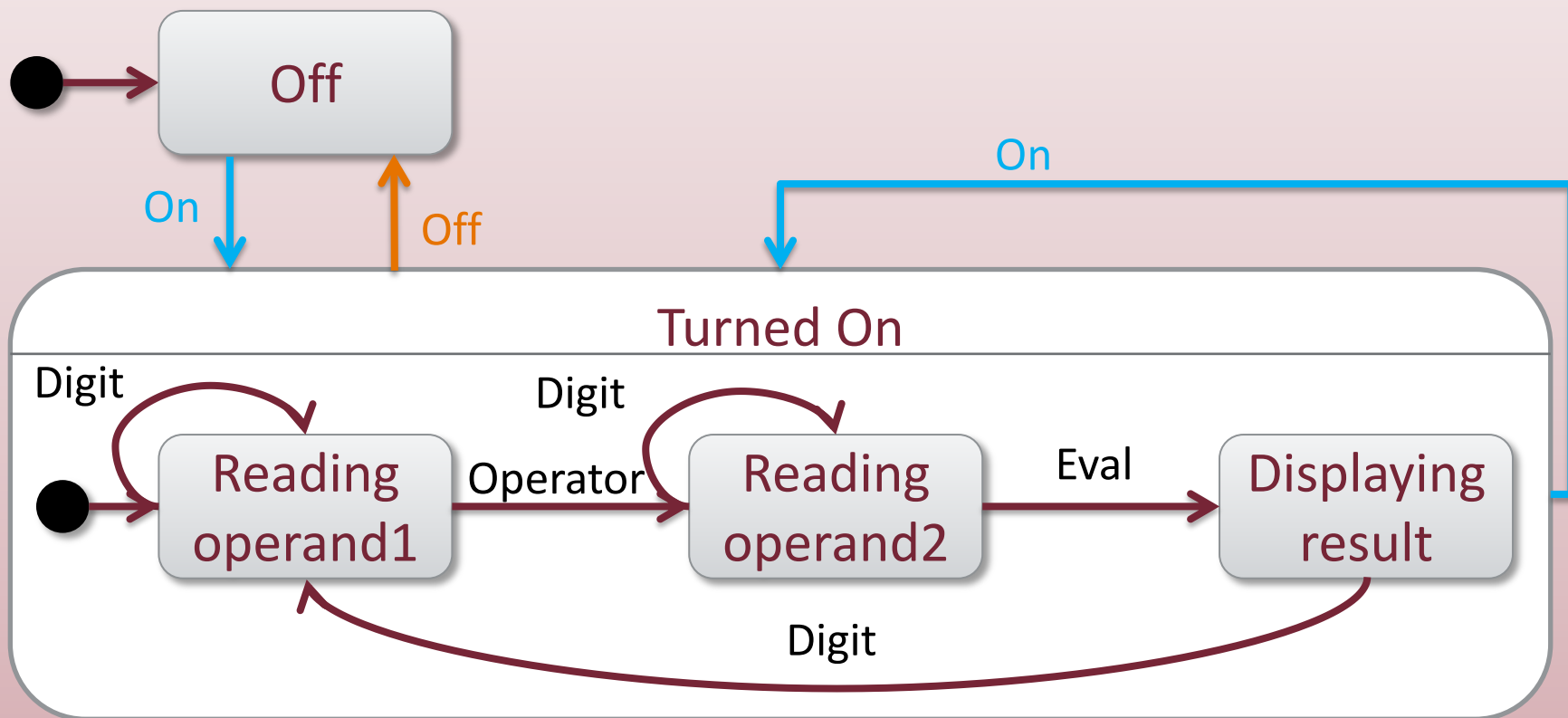
CALCULATOR



- Bemenet: {on, off, digit, operand, eval}
- Feltételezés: kétoperandusú műveleteink vannak

Állapothierarchia

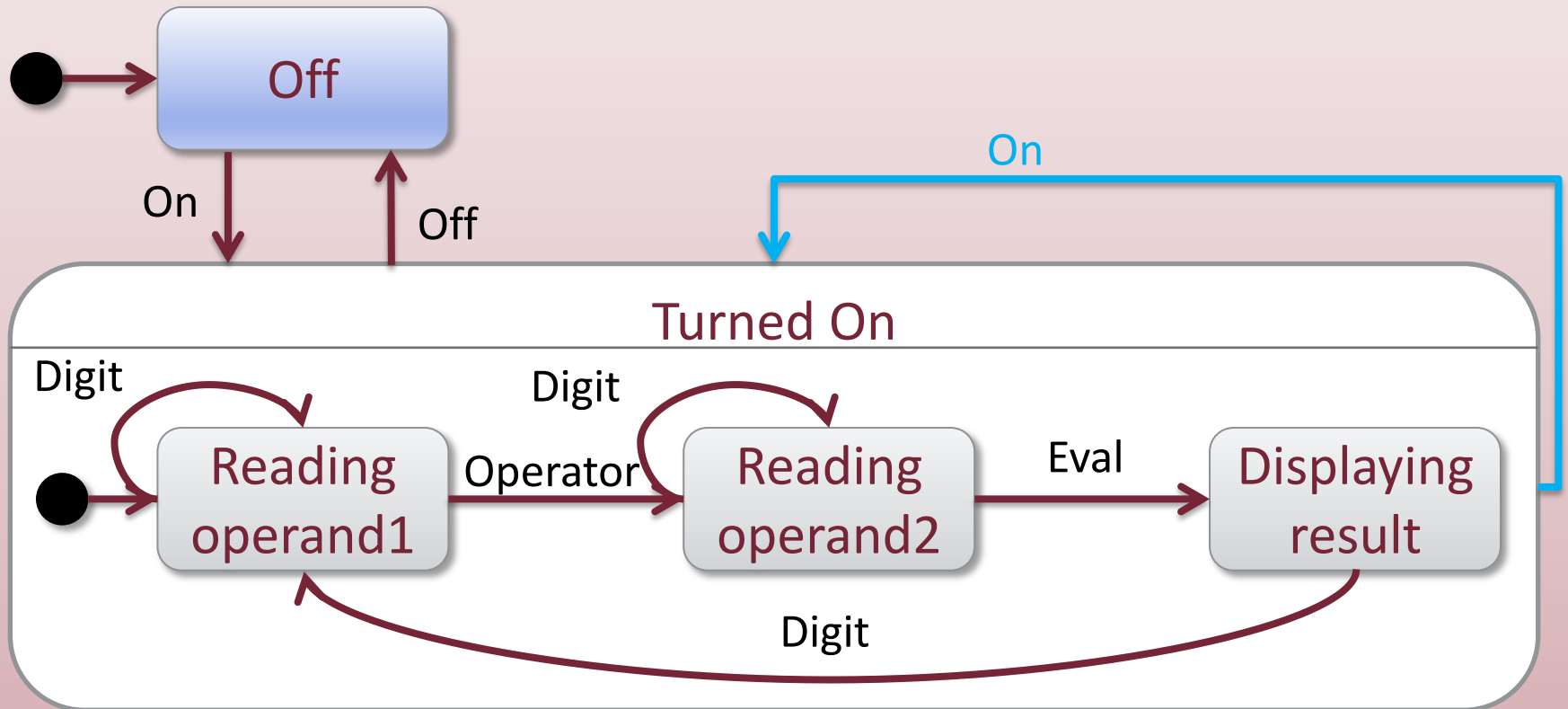
CALCULATOR



- Közös viselkedés kiemelése közös absztrakcióba

Állapothierarchia

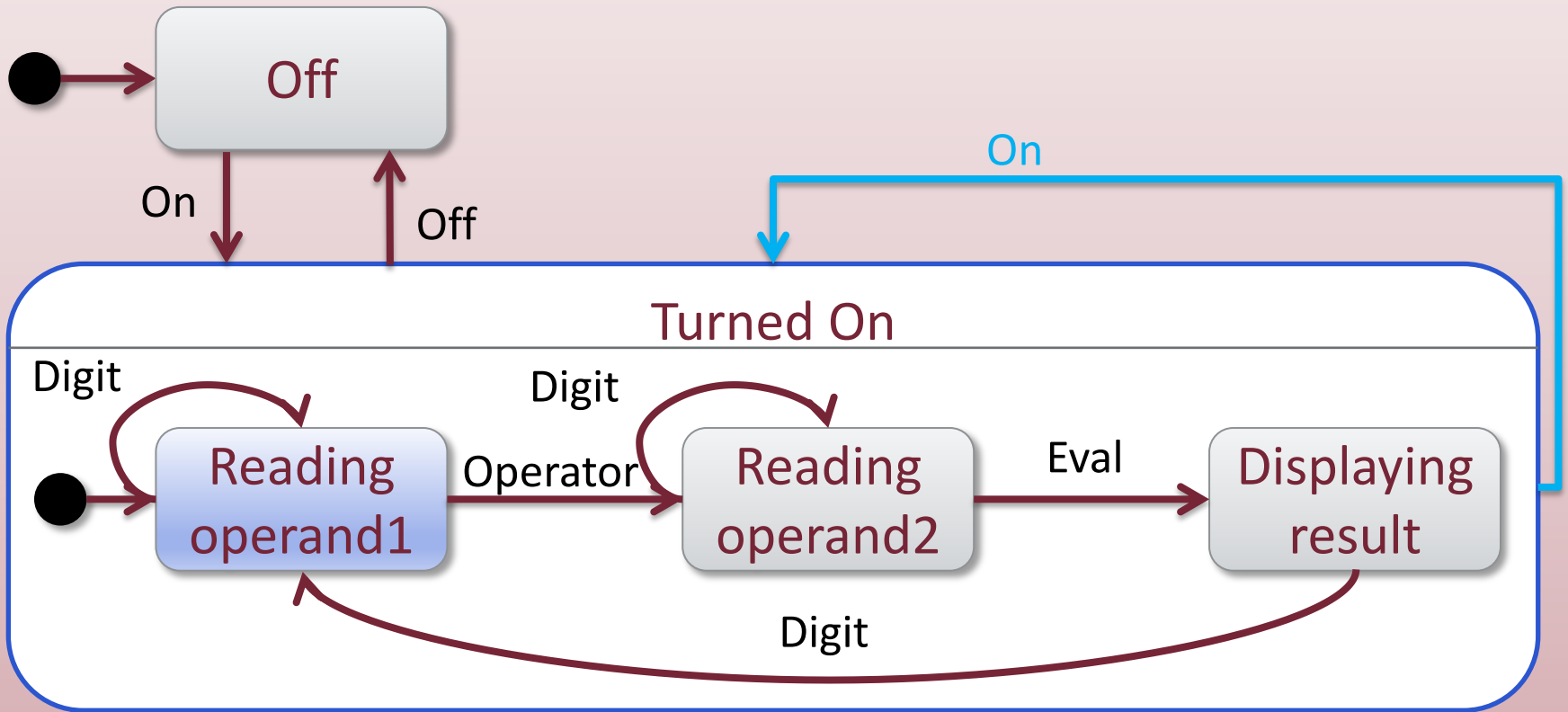
CALCULATOR



- Állapotkonfiguráció: $\{Off\}$

Állapothierarchia

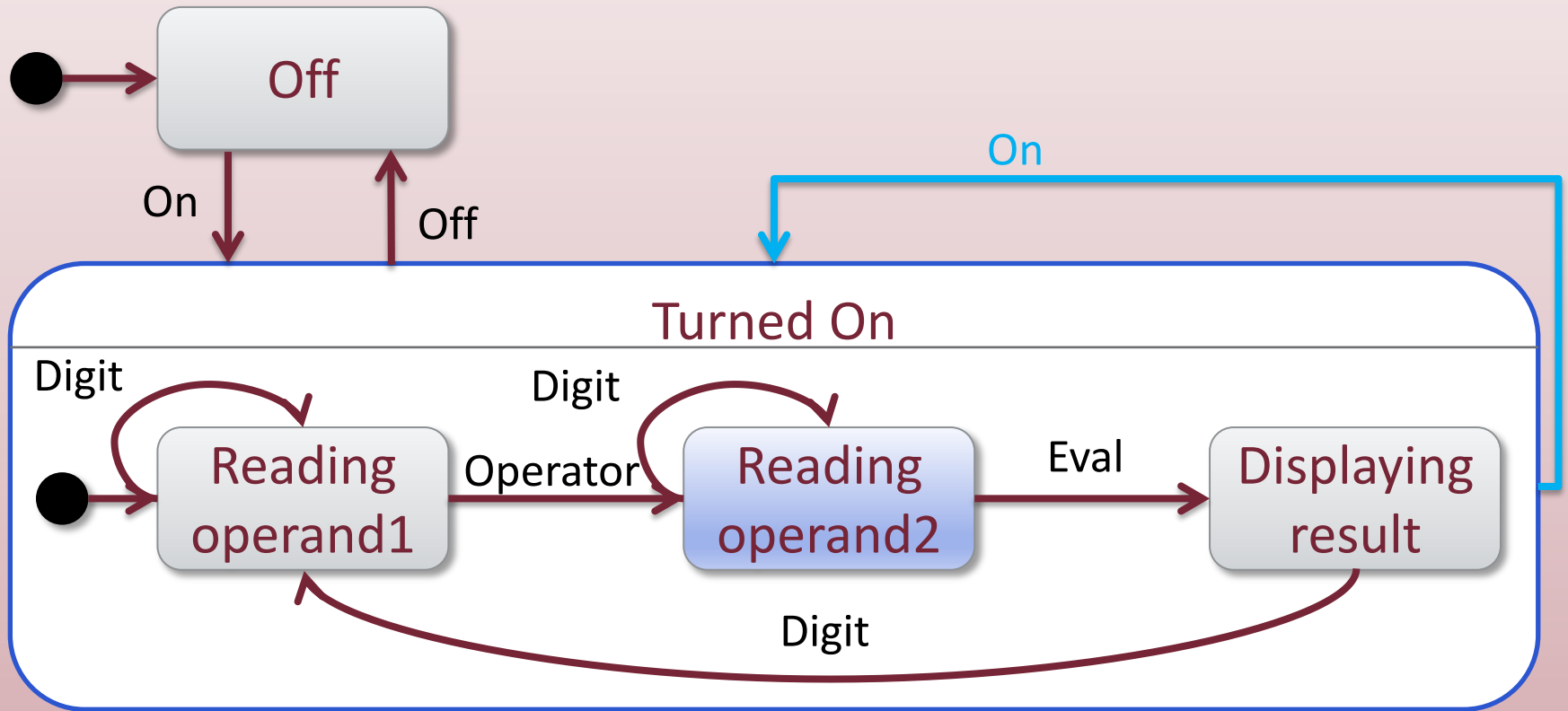
CALCULATOR



- Állapotkonfiguráció: $\{On, Reading\ operand1\}$

Állapothierarchia

CALCULATOR



- Állapotkonfiguráció: $\{On, Reading\ operand2\}$

Állapothierarchia

CALCULATOR

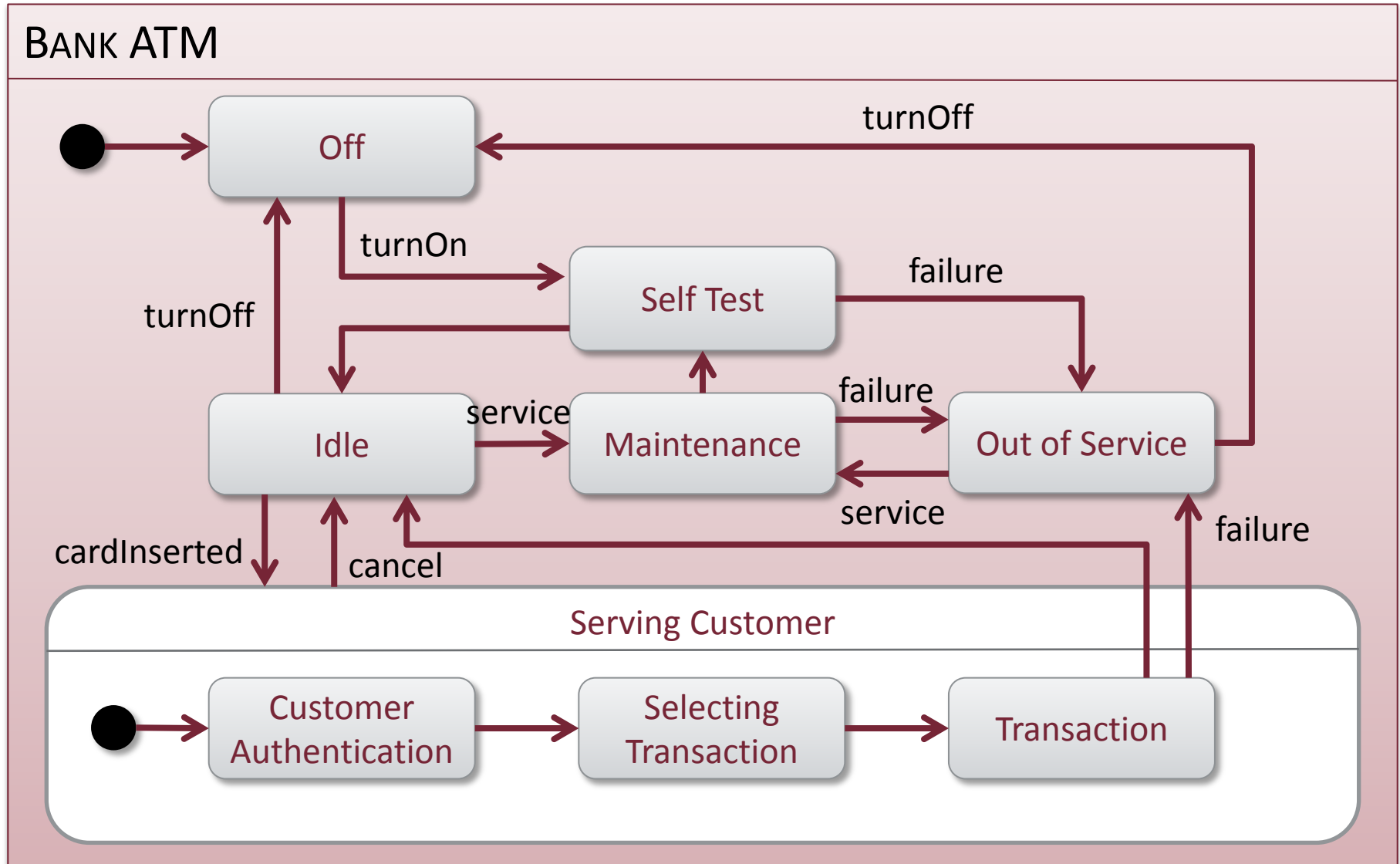


Két állapot egyszerre → sérti a kizárólagosságot?

- Nem, a statechart nem állapottér
- Csak az egy szinten lévő állapotok alkotnak kizárólagos állapothalmazt

- Állapotkonfiguráció: $\{On, Reading\ operand2\}$

Példa: ATM

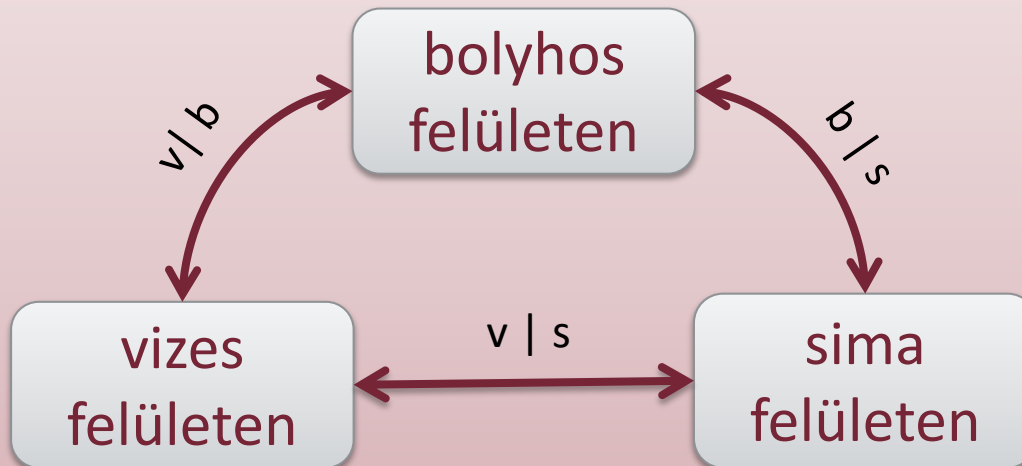


Példa: robotporszívó



Ortogonalitás

ROBOTPORSZÍVÓ HELYE



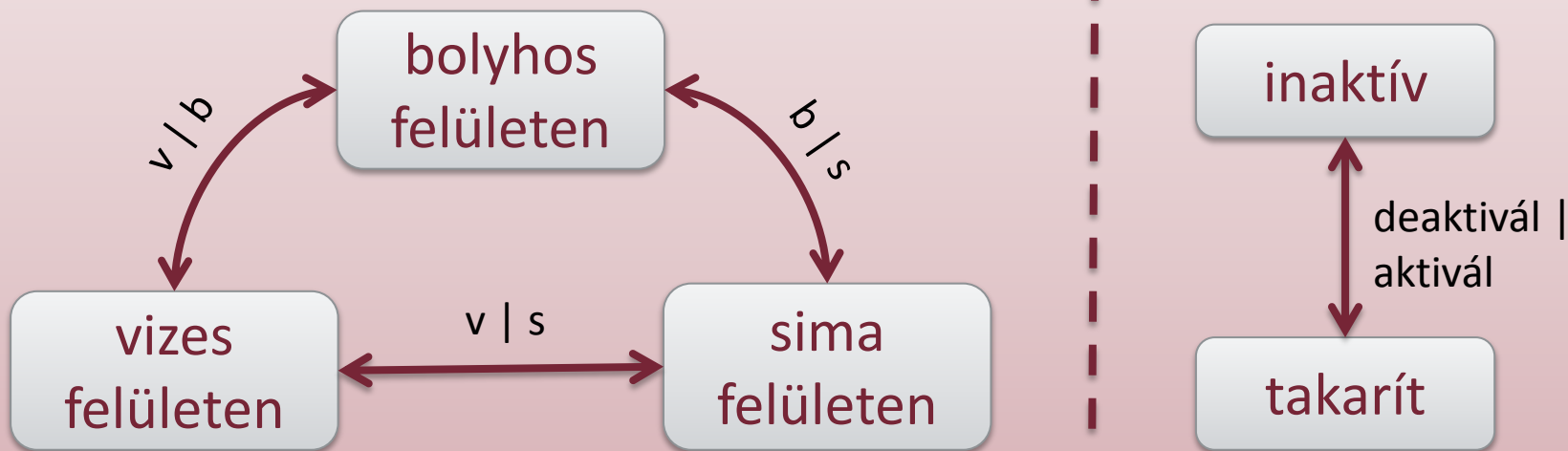
ROBOTPORSZÍVÓ ÜZEMMÓDJA



Megj: kétirányú nyíl nem szokásos, csak a tömör ábrázolás miatt használjuk...

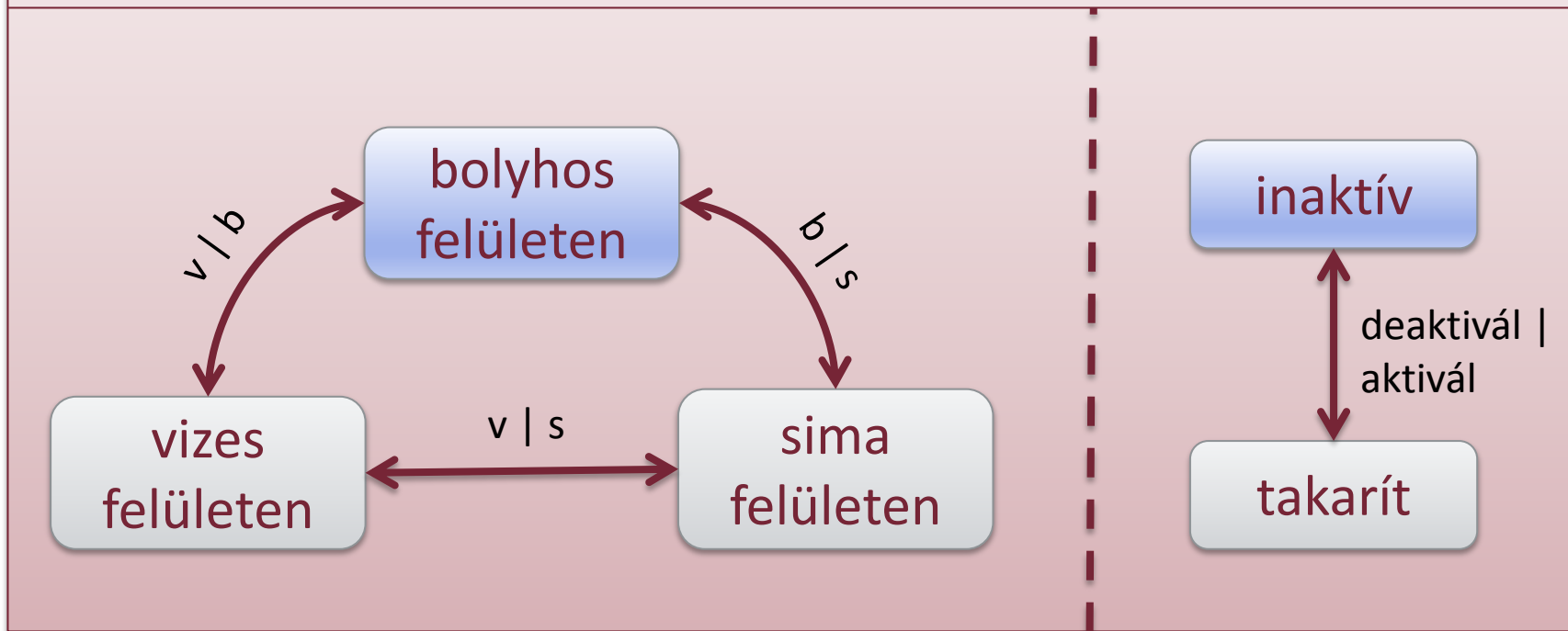
Ortogonalitás

ROBOTPORSZÍVÓ HELYE ÉS ÜZEMMÓDJJA



Ortogonalitás

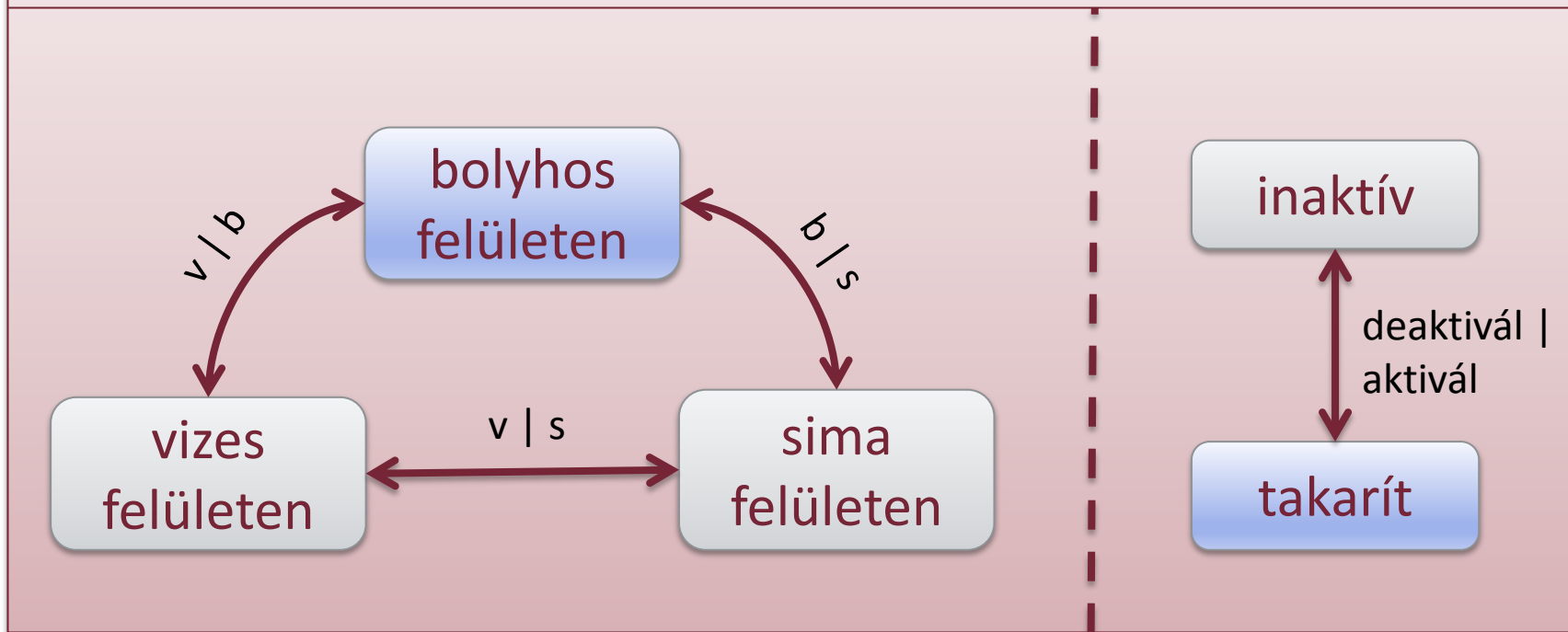
ROBOTPORSZÍVÓ HELYE ÉS ÜZEMMÓDJJA



- Állapotkonfiguráció: $\{bolyhos\ felületen, inaktív\}$

Ortogonalitás

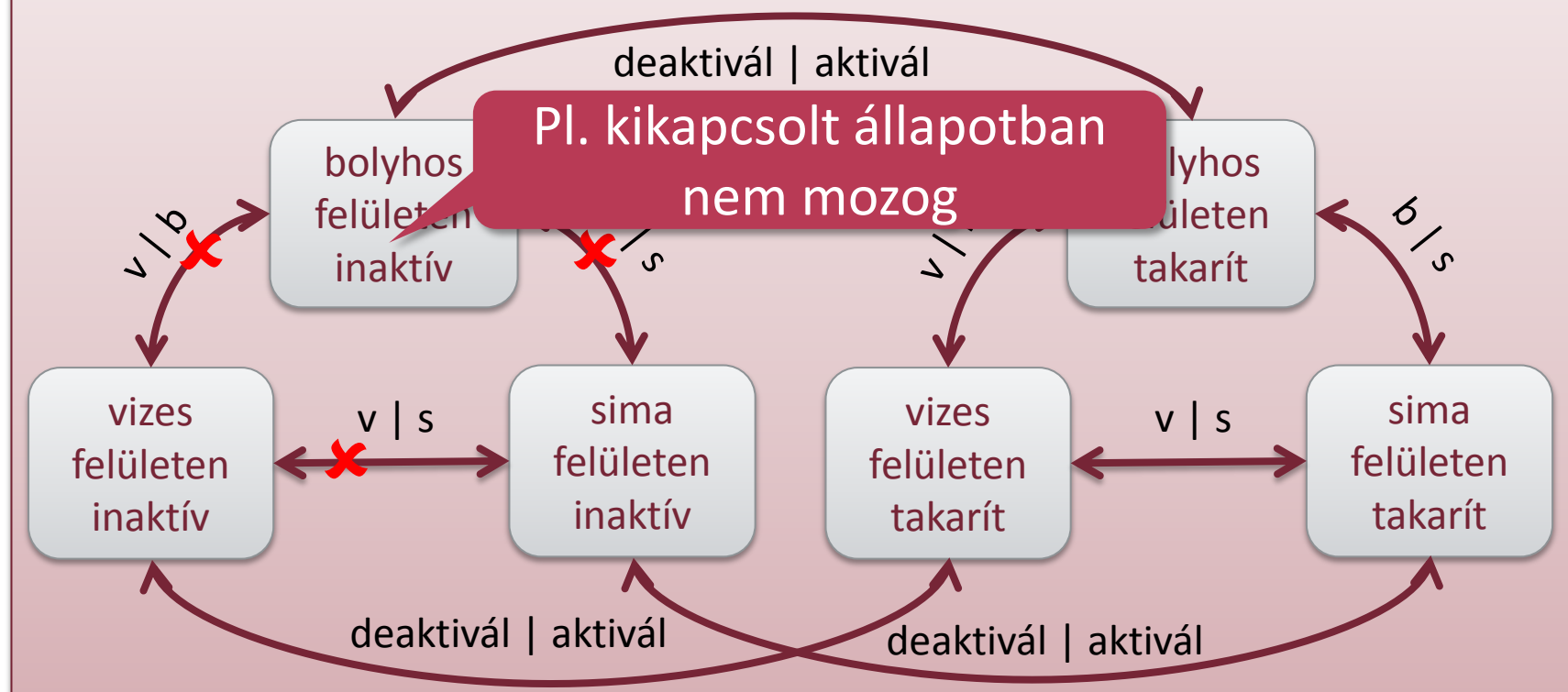
ROBOTPORSZÍVÓ HELYE ÉS ÜZEMMÓDJJA



- Állapotkonfiguráció: $\{bolyhos\ felületen, takarít\}$

Régiók aszinkron szorzata

ROBOTPORSZÍVÓ HELYE ÉS ÜZEMMÓDJA

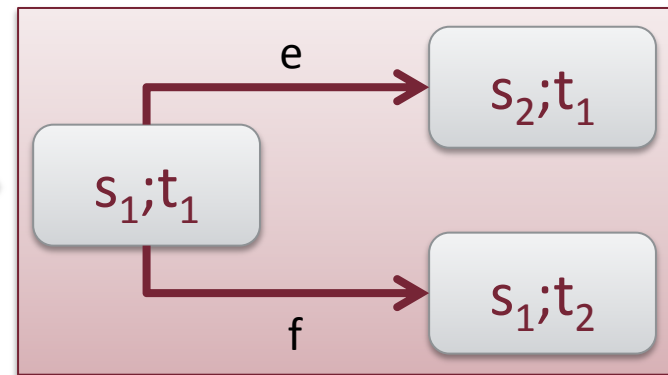
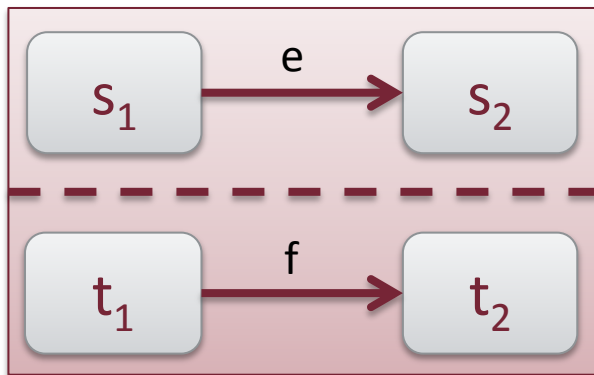


- További finomítást igényel: átmenetek kieshetnek
→ Állapotok így elérhetetlenné válhatnak

Definíció: Aszinkron szorzat

A (Mealy-) állapotgépek **aszinkron szorzata** (régióknak is nevezett) komponens állapotgépeken végzett **kompozíciós művelet**. A szorzat **eredménye** egy (Mealy) állapotgép, melynek

- állapottere a régiók állapottereinek direkt szorzata,
- kezdőállapotában az összes régió kezdőállapotban van,
- és átmeneti szabályait az összes olyan lépés alkotja, amelyben
 - **pontosan egy régió** állapotátmenetet végez,
 - míg a többi régió állapota nem változik.



Változók

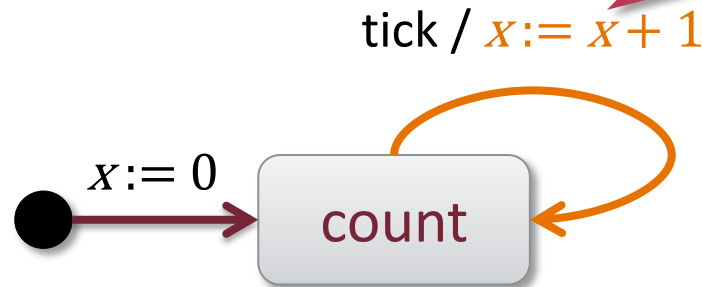
- Végtelen számláló

- $S = \mathbb{N}$



Valójában x odaképezhető egy másik állapotrégióba...

- Változó bevezetése: x

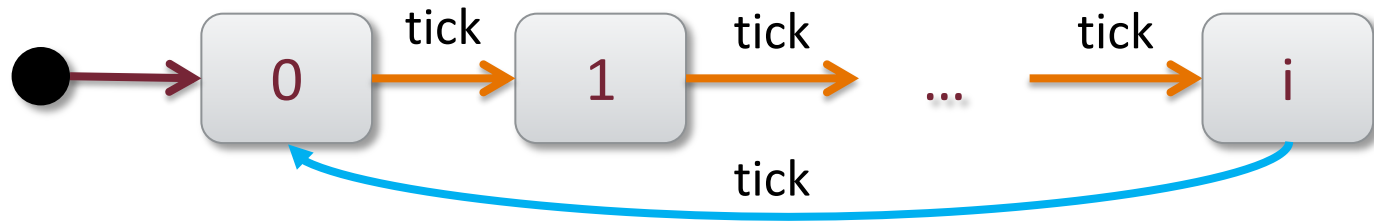


$$(count, \{x \mapsto 0\}) \rightarrow (count, \{x \mapsto 1\}) \rightarrow \dots$$

Változók + őrfeltételek

- Ciklikus számláló

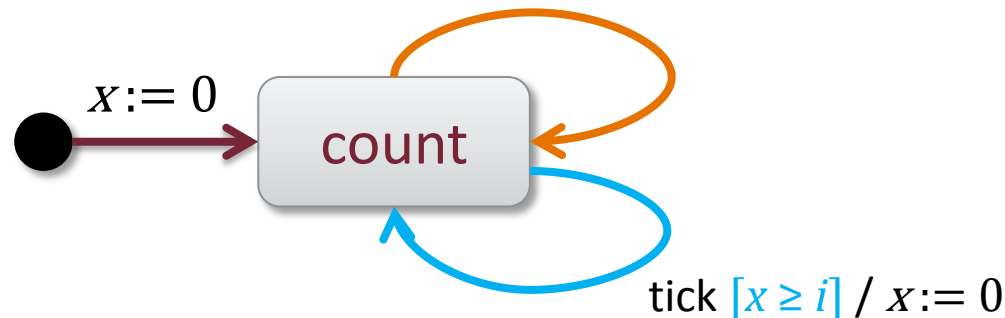
- $S = \{0, 1, \dots, i\}$



- Őrfeltételekkel:

Változó vagy állapotrégió
figyelembe vehető

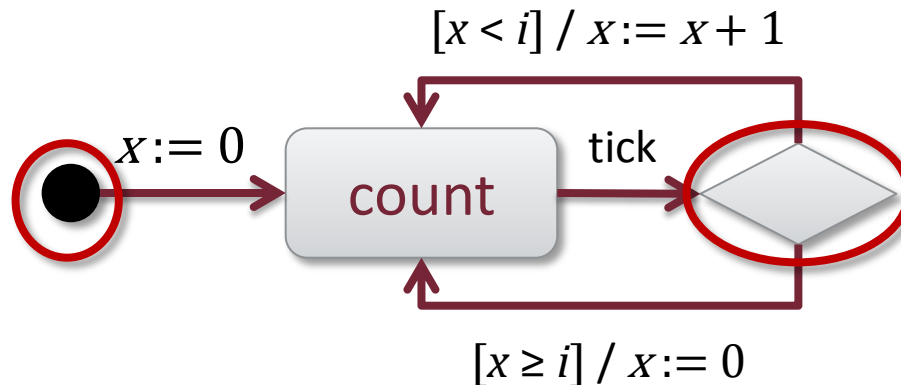
tick $[x < i] / x := x + 1$



Pseudoállapotok

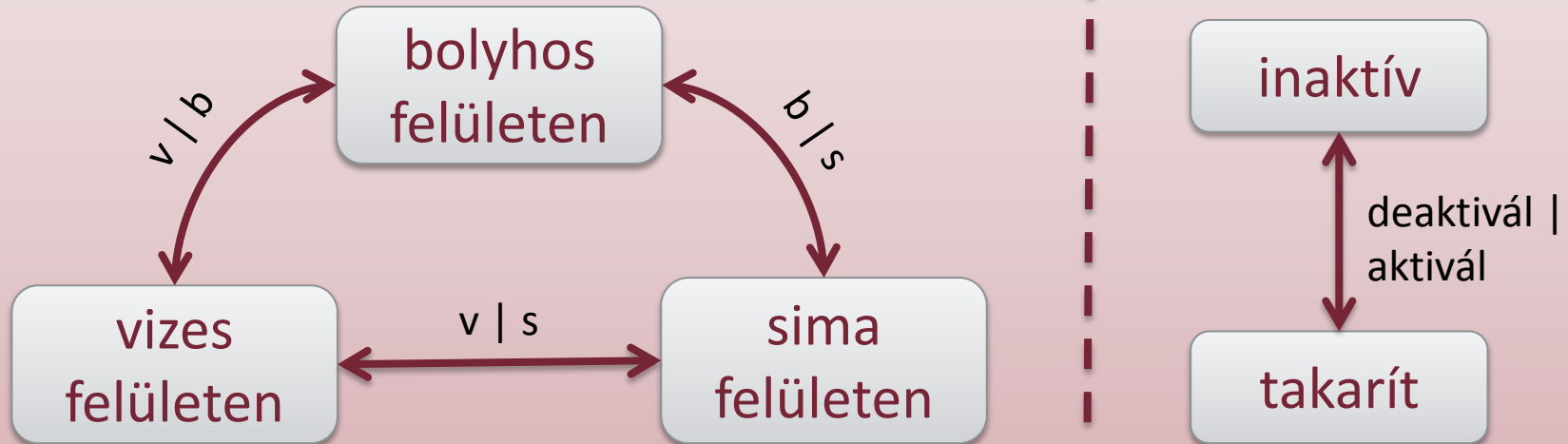
■ Pseudoállapot:

- Szemantikailag nem állapot:
 - Nincs olyan időpillanat, amikor a rendszert jellemezné
- Szintaktikailag állapot:
 - Lehet tranzíció kezdő- vagy célállapota



Kooperáció aszinkron szorzat esetén

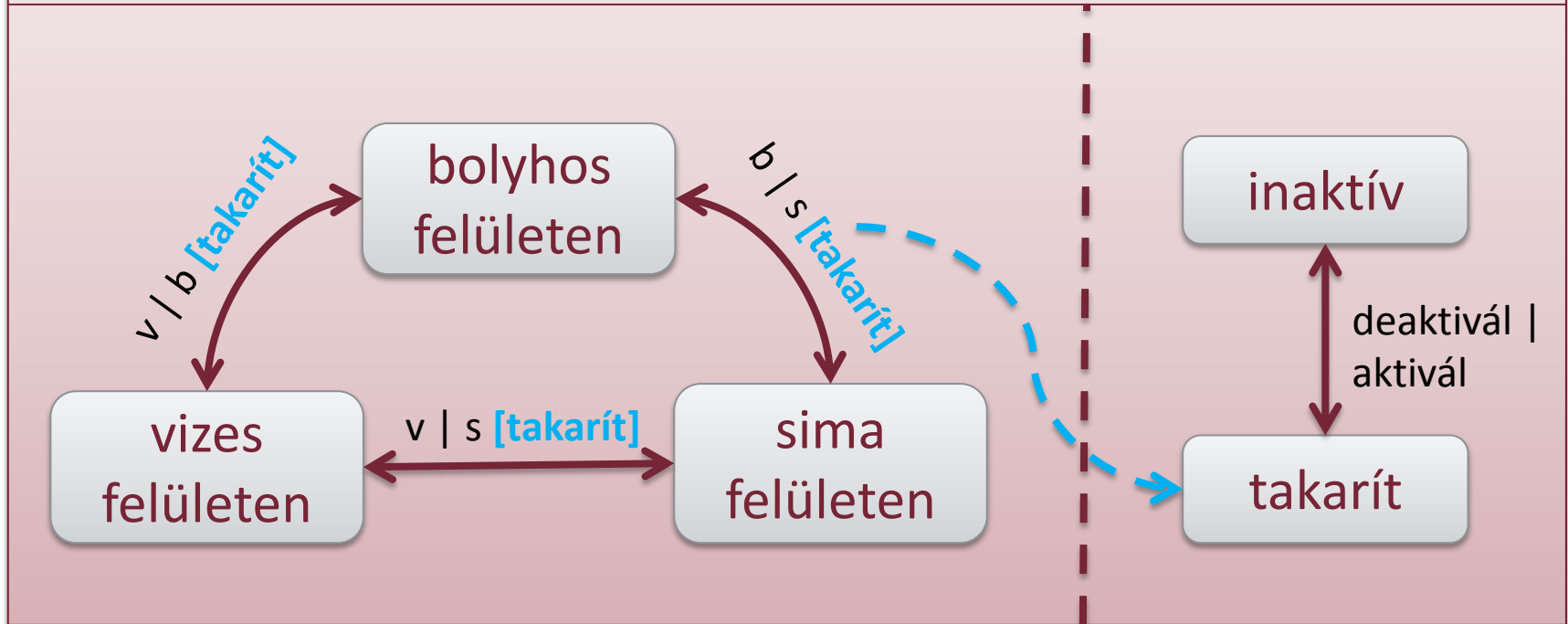
ROBOTPORSZÍVÓ HELYE ÉS ÜZEMMÓDJJA



- Hogyan modellezzük a kooperációt?
 - Egészen pontosan hogyan **nem független** a két régió?

Kooperáció aszinkron szorzat esetén

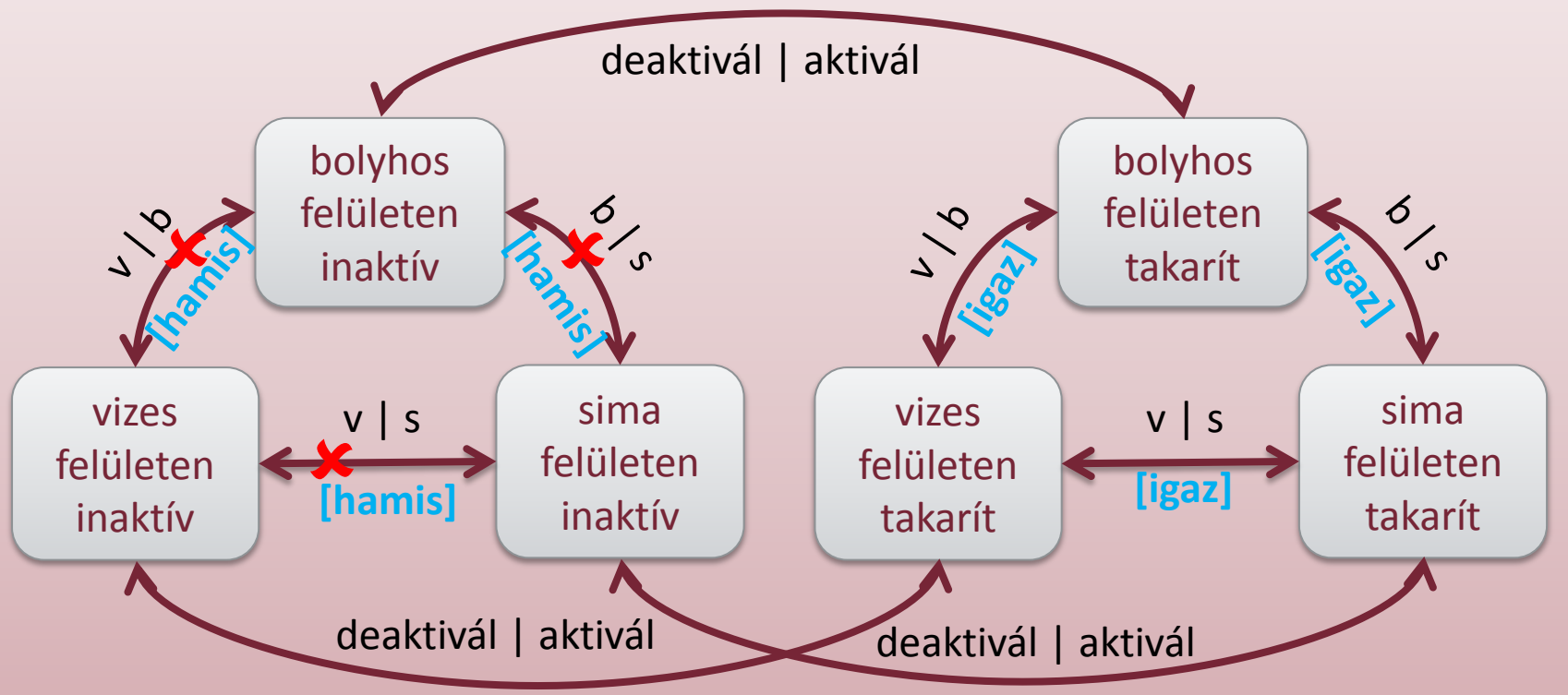
ROBOTPORSZÍVÓ HELYE ÉS ÜZEMMÓDJA



- Kooperáció **őrfeltétellel**
 - Egyik régióban átmenet feltétele másik régió állapota

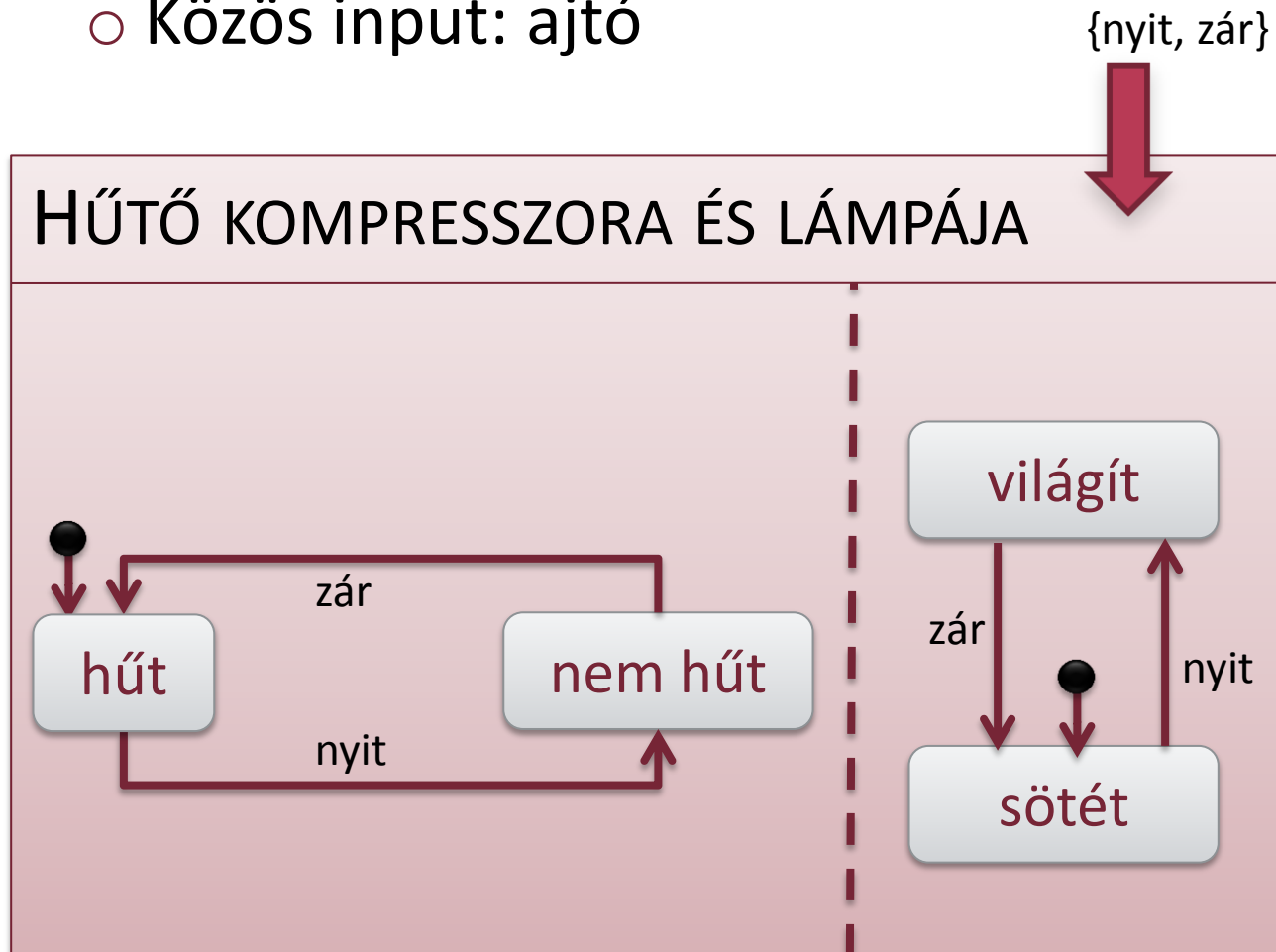
Kooperáció aszinkron szorzat esetén

ROBOTPORSZÍVÓ HELYE ÉS ÜZEMMÓDJA



Szinkron szorzat példa

- Hűtőszekrény kompresszor vs. lámpa
 - Közös input: ajtó



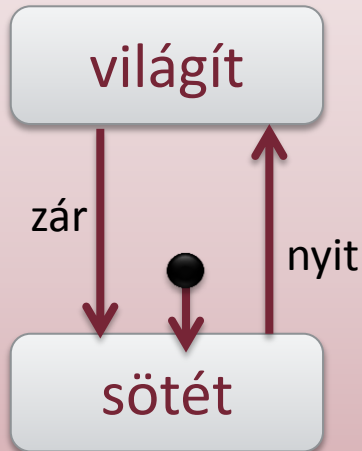
Szinkron szorzat példa

- Közös inputolvasás
→ szinkron lépés
 - (A kezdőállapotból elérhetetlenné válhatnak állapotok)

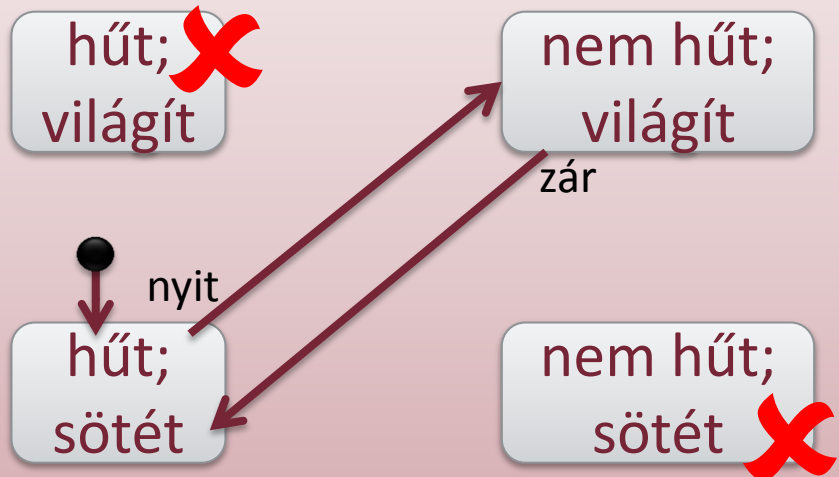
KOMPRESSZOR



LÁMPA



HŰTŐ KOMPRESSZORA ÉS LÁMPÁJA



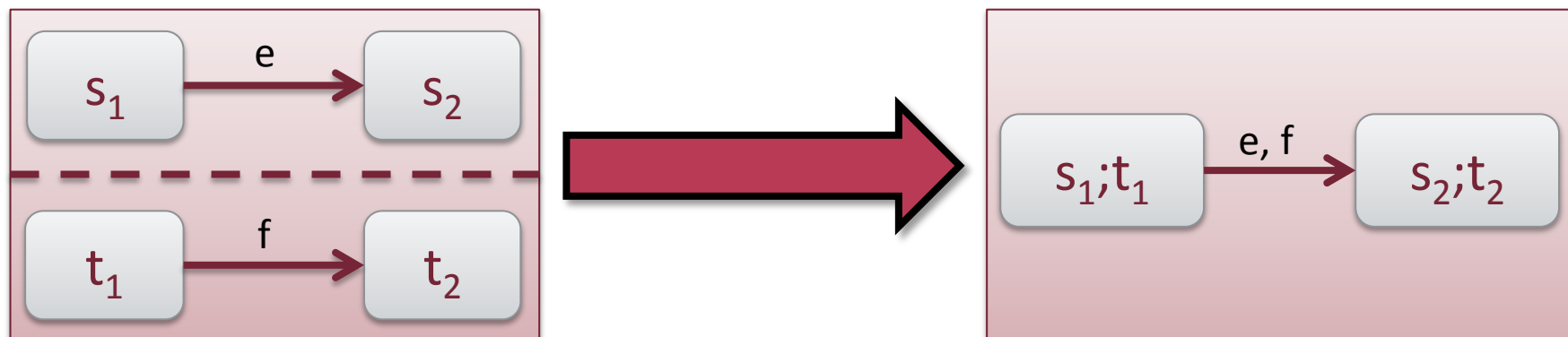
Szinkron szorzat

■ Állapotgépek szorzata

- Modell felépítése állapotrégiókból (komponensekből)
- Állapothalmaz: a régiók állapottereinek **direkt szorzata**
- Kezdőállapot: a régiók kezdőállapotaiból álló n-es

■ Átmenetek

- **Szinkron szorzat:** mindkét állapotváltozó egyszerre lép
 - Egy-egy átmenet „összeragasztása”, címkék uniója



Vegyes szorzat

- Állapotgépek szorzata
 - Modell felépítése állapotrégiókból (komponensekből)
 - Állapothalmaz: a régiók állapottereinek **direkt szorzata**
 - Kezdőállapot: a régiók kezdőállapotaiból álló n -es
- Átmenetek
 - **Vegyes szorzat**: helyenként szinkronizáció történik
 - Alapvetően aszinkron kompozíció...
 - ...de bizonyos esetekben együtt lépnek a régiók

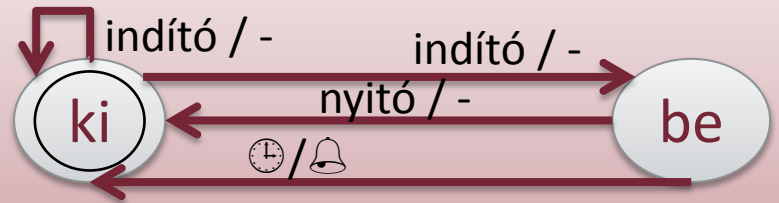
Szinkronizáció egyszerű esete:
van közös input (is)

Vegyes szorzat példa

- További finomítást igényel

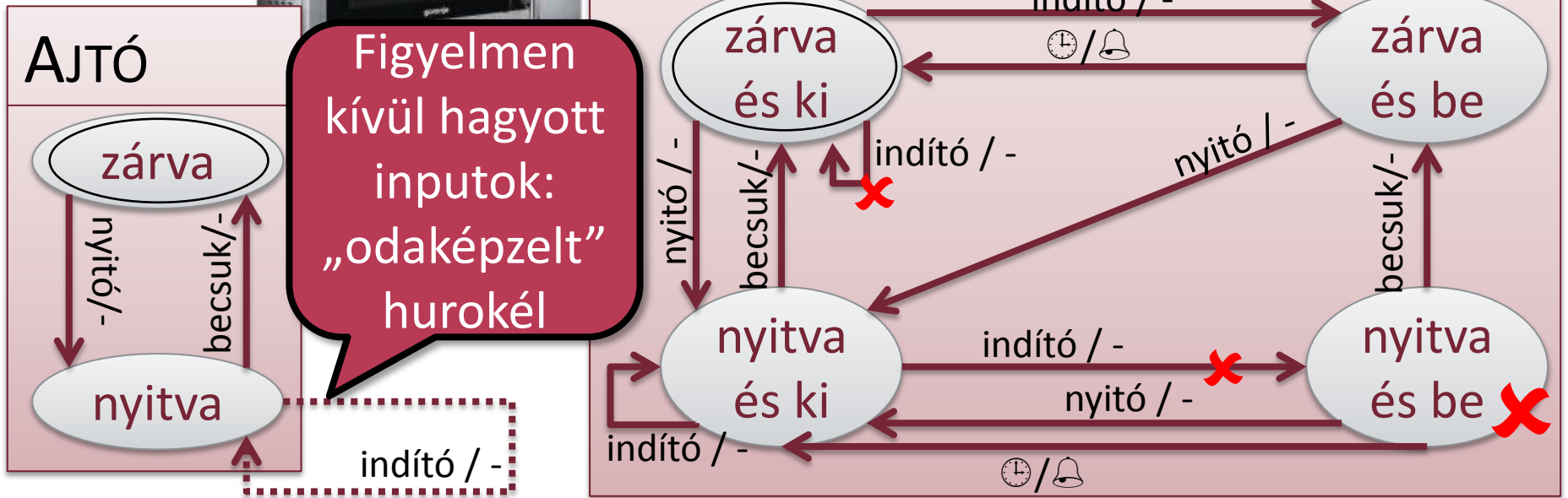
- Átmenetek kieshetnek
- (A kezdőállapotból így elérhetetlenné válhatnak állapotok)

MAGNETRON



Figyelman kívül hagyott inputok: „odaképzelt” hurokél

AJTÓ ÉS MAGNETRON



Vegyes szorzat

- Állapotgépek szorzata
 - Modell felépítése állapotrégiókból (komponensekből)
 - Állapothalmaz: a régiók állapottereinek **direkt szorzata**
 - Kezdőállapot: a régiók kezdőállapotaiból álló n -es
- Átmenetek
 - **Vegyes szorzat**: helyenként szinkronizáció történik
 - Alapvetően aszinkron kompozíció...
 - ...de bizonyos esetekben együtt lépnek a régiók

Szinkronizáció egyszerű esete:
van közös input (is)

Fejlett kooperáció: **randevú**
(belső szinkronizáló esemény)

Szorzatok áttekintése

■ Állapotterek szorzata

○ **Direkt szorzat:** $S_1 \times S_2 \times \dots \times S_n$

- Összetett állapot: komponensek állapotaiból képzett n-es

■ Állapotgépek szorzata

○ Az állapottér mindig a direkt szorzat (finomítása)

○ **Szinkron szorzat**

- Mindig egyszerre lépnek a komponensek / régiók

○ **Aszinkron szorzat**

- Mindig csak egy komponens / régió lép

○ **Vegyes szorzat**

- Helyenként egy, helyenként több komponens / régió lép

Szorzatok áttekintése

■ Állapotterek szorzata

- **Direkt szorzat:** $S_1 \times S_2 \times \dots \times S_n$

- Összetett állapot: komponensek állapotaiból képzett n-es

■ Állapotgépek szorzata

- Az állapottér mindig a direkt szorzat (finomítása)

- **Szinkron szorzat**

- Mindig egyszerre lépnek a komponensek / régiók

- **Aszinkron szorzat**

- Mindig csak egy komponens / régió lép

- **Vegyes szorzat**

- Helyenként egy, helyenként több komponens / régió lép

Kooperáció módjai

Örfeltétel

Randevú

Előismeretek

Állapottér

Mealy gépek

Harel gépek

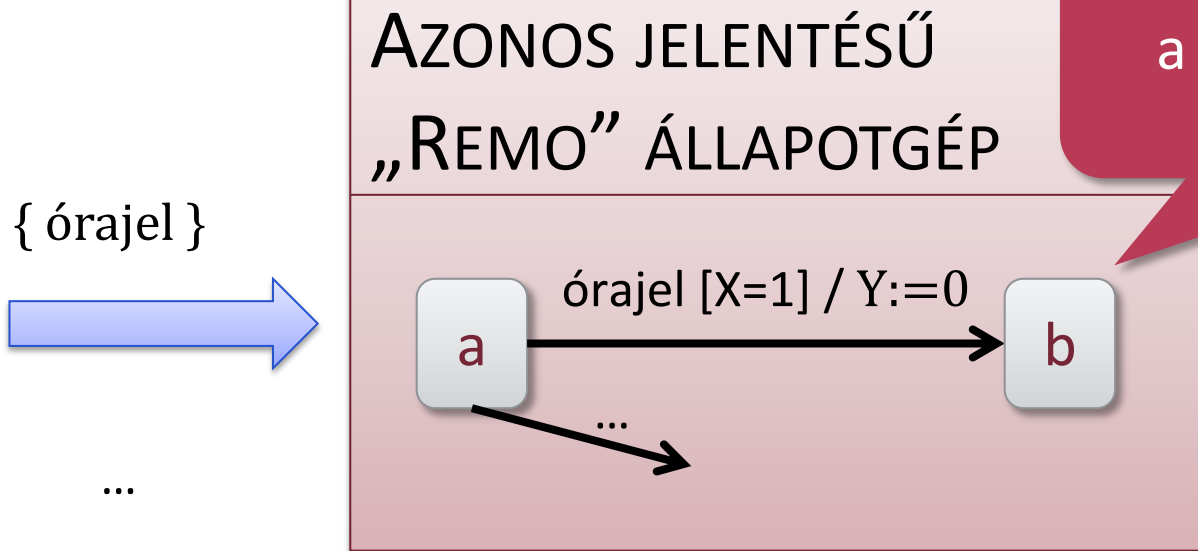
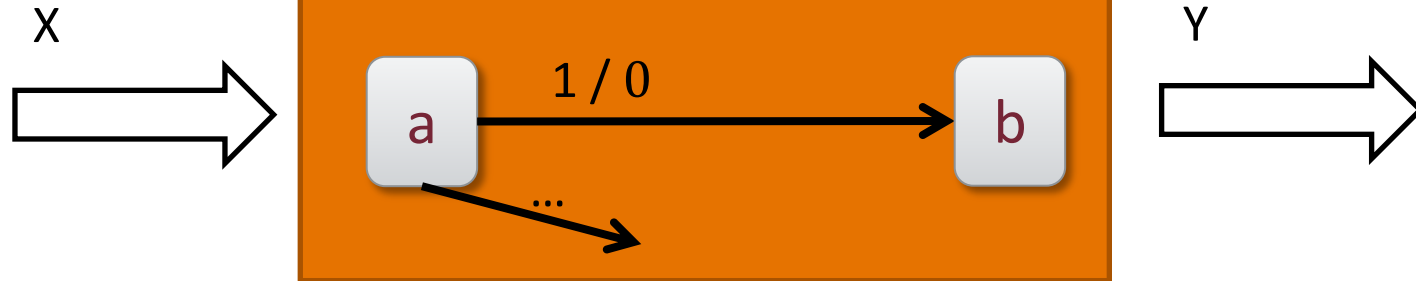
Kitekintés

KITEKINTÉS, SZOFTVERTÁMOGATÁS

Az állapotátmenet értelmezése

- Digitális áramkörtervezésnél
 - Szinkron sorrendi hálózat (ld. *Digitális Technika*)
 - Egyféle input esemény (órajel)
 - Állapotátmeneten feltüntetett input feltételek
 - Mindig órajelre lép → fel se tüntetjük
 - Input feltétel: a bemenő jelvezetékek *állapotára* vonatkozik
- Ezzel szemben **általános rendszermodellezésnél**
 - Sokféle modellezett input esemény (jel)
 - Állapotátmeneten feltüntetett input feltételek
 - Elsődleges feltétel: kiváltó **input esemény**
 - Opcionálisan más részrendszerek állapotától függ: **őrfeltétel**

Az állapotátmenet értelmezése



Az órajel input esemény,
a vezetékeken lévő jel
állapotváltozó

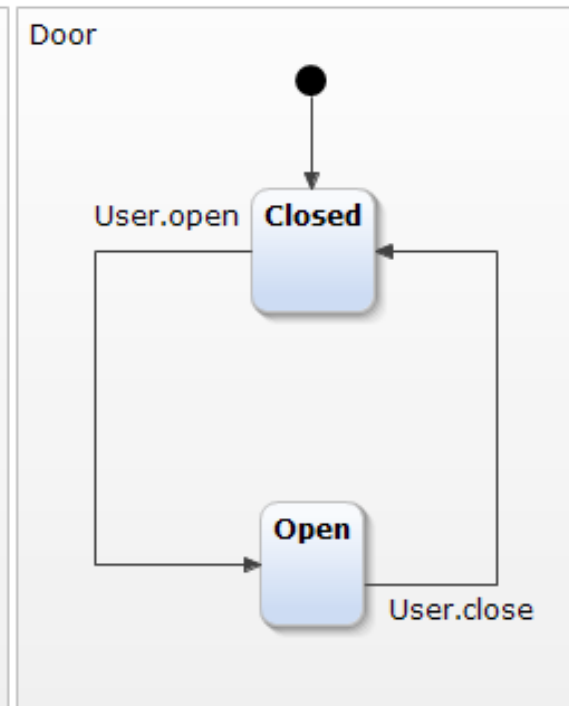
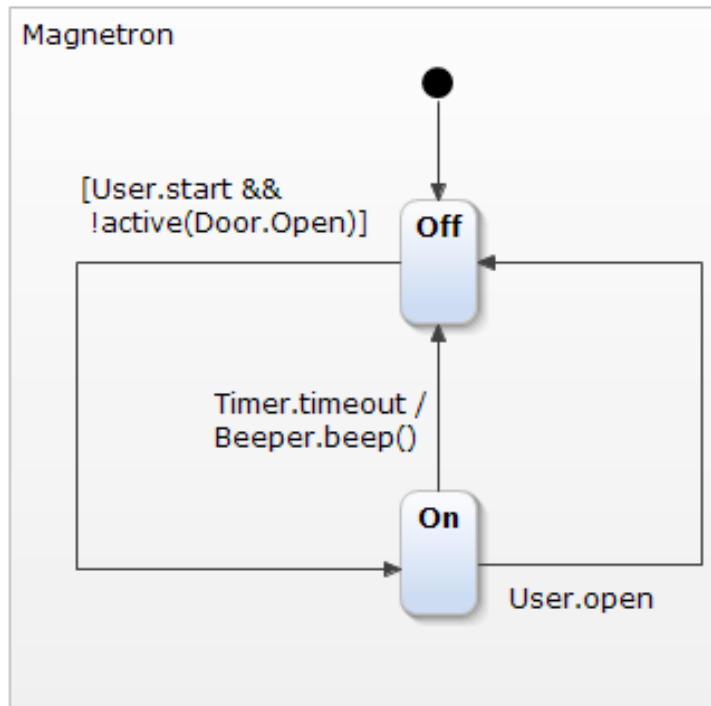
Yakindu Statechart Tools

- Kompozit állapotgépek szerkeszthetők 
 - (statechart nyelv → tömören kifejezhető kompozíció)

Micro
interface User:
in event open
in event close
in event start

interface Timer:
in event timeout

interface Beeper:
operation beep()

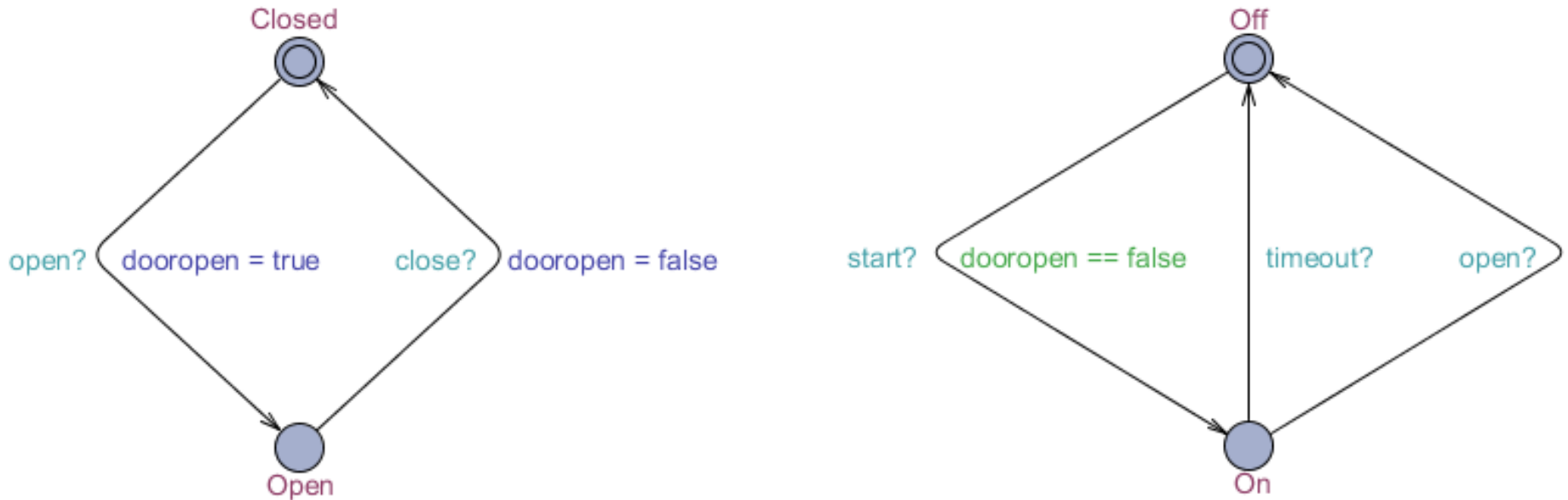


Yakindu Statechart Tools

- Kész állapotgépből C / Java / ... kód generálható
 - *Magnetron* lép *On* állapotban (egyszerűsítve)

```
/* The reactions of state On. */
private void reactMagnetron_On() {
    if (sCITimer.timeout) {
        sCIBeeper.operationCallback.beep();
        stateVector[0] = State.magnetron_Off;
    } else {
        if (sCIUser.open) {
            stateVector[0] = State.magnetron_Off;
        }
    }
}
```

■ Állapotgépek vegyes szorzata



■ Kompozit állapotgép viselkedése...

- ...szimulálható
- ...verifikálható

```
A[] (! (door.Open && magnetron.On))
```



ÖSSZEFOGLALÁS

Definíció: Állapottér

Az **állapottér** egymástól megkülönböztetett **rendszerállapotok** halmaza, amelynek minden időpontban pontosan egy eleme jellemzi a rendszert.

Egy adott időpontban a rendszer **pillanatnyi állapota** az állapottér azon egyetlen eleme, amelyik abban az időpontban jellemző a rendszerre.

○ Állapottér példák

- {*hétfő, kedd, szerda, csütörtök, péntek, szombat, vasárnap*}
- mikro állapotai: {*teljes fokozat, kiolvasztó mód, kikapcsolva*}

○ Pillanatnyi állapot példák

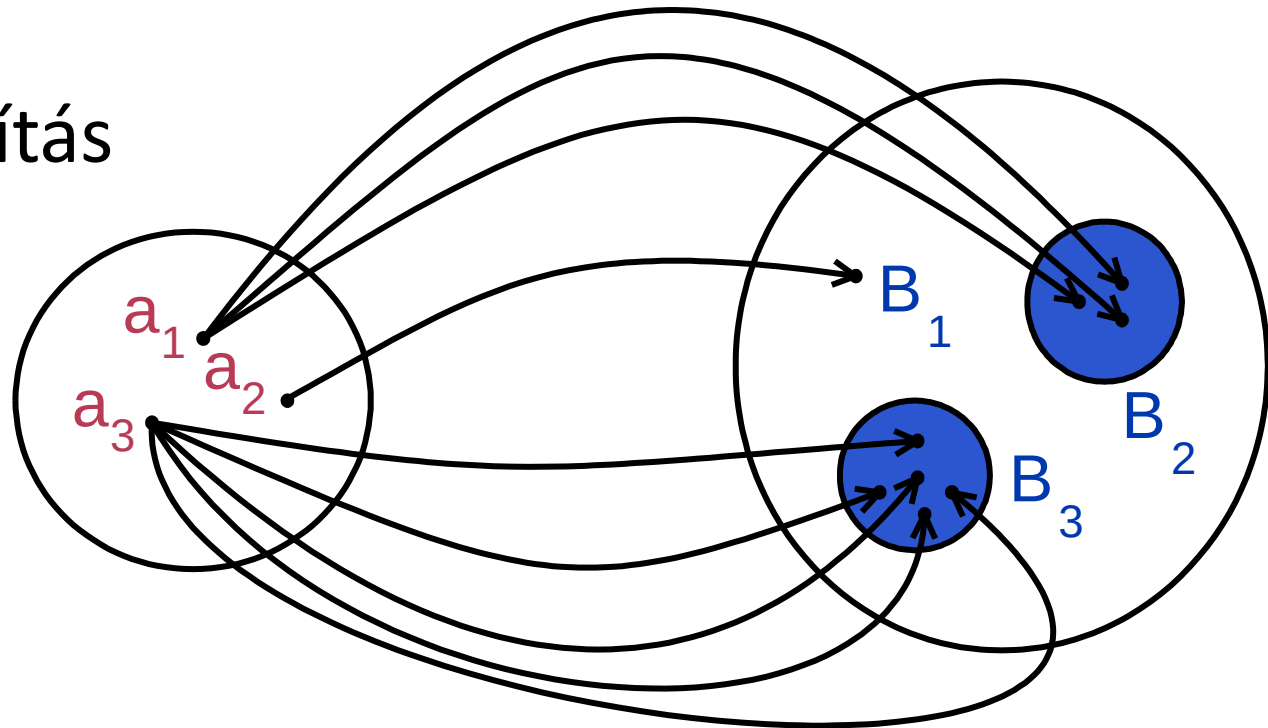
- ma *szerda* van
- a mikro ajtaja *nyitva*

Definíció: Állapotfinomítás és -absztrakció

Az **állapotfinomítás** ill. **állapotabsztrakció** az állapottéren mint halmazon végzett **halmazfinomítás** ill. **halmazabsztrakció**, melynek eredménye egy újabb állapottér.

- (Másfajta absztrakciókkal is találkozunk majd...)

Ismétlés:
halmazfinomítás



Definíció: Állapotterek (direkt) szorzata

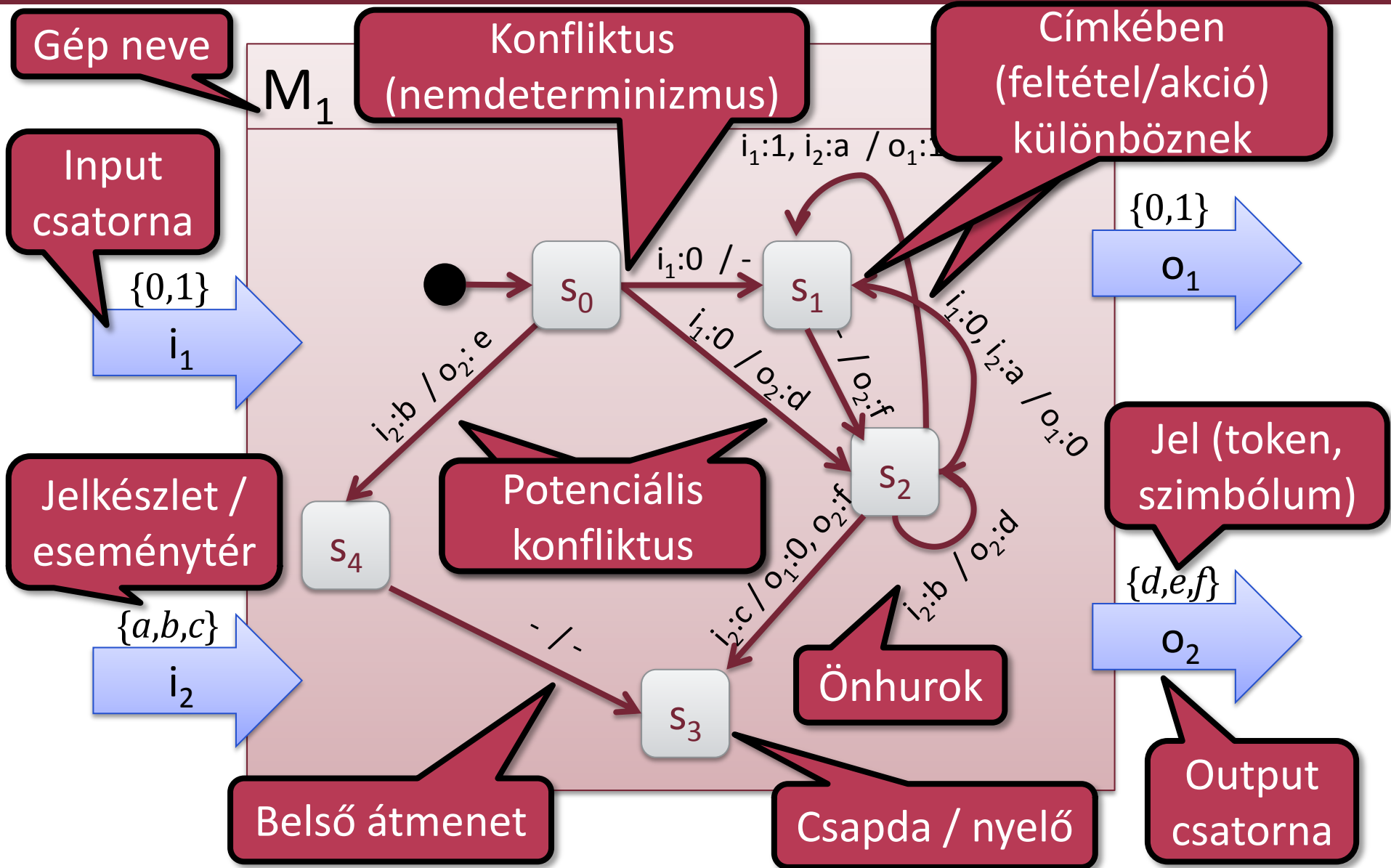
Az állapotterek **direkt szorzata** komponens állapottereken végzett **kompozíciós művelet**.

A szorzat **eredménye** egy újabb állapottér, amely a komponens állapotterek mint halmazok Descartes-szorzataként áll elő

A szorzat állapottérben a komponens állapotterek minden állapot-kombinációjának egy összetett állapot felel meg.

A komponens(ek)re történő **vetítés** egy **állapotabsztrakciós művelet**, amely a szorzat állapottérből egy vagy több komponenst tart meg, a többit elhanyagolja.

Kiterjesztett állapotgép



Állapothierarchia

CALCULATOR



Két állapot egyszerre → sérti a kizárólagosságot?

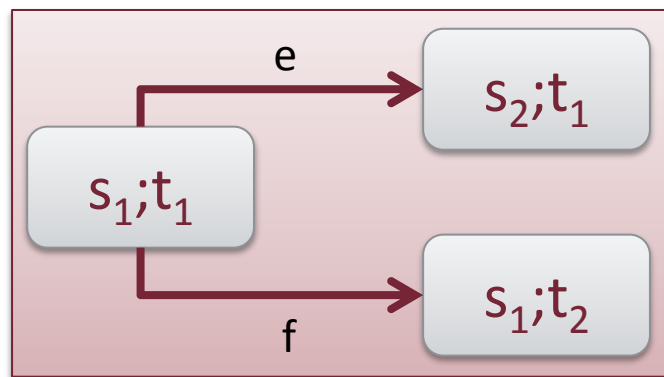
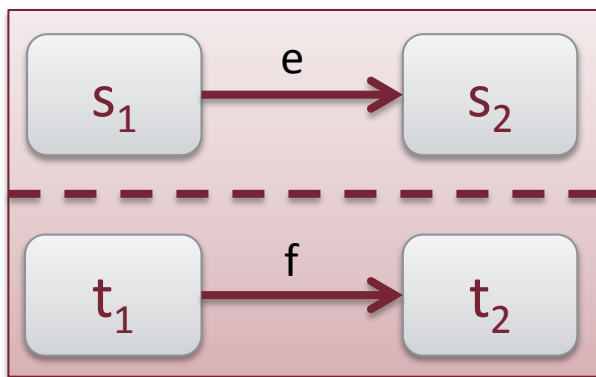
- Nem, a statechart nem állapottér
- Csak az egy szinten lévő állapotok alkotnak kizárólagos állapothalmazt

- Állapotkonfiguráció: $\{On, Reading\ operand2\}$

Definíció: Aszinkron szorzat

A (Mealy-) állapotgépek **aszinkron szorzata** (régióknak is nevezett) komponens állapotgépeken végzett **kompozíciós művelet**. A szorzat **eredménye** egy (Mealy) állapotgép, melynek

- állapottere a régiók állapottereinek direkt szorzata,
- kezdőállapotában az összes régió kezdőállapotban van,
- és átmeneti szabályait az összes olyan lépés alkotja, amelyben
 - **pontosan egy régió** állapotátmenetet végez,
 - míg a többi régió állapota nem változik.



Magyar - English

Felépítési modell	Structural model
Viselkedési modell	Behavioural model
Esemény	Event
Pillanatszerű	Instantaneous
Eseményfolyam	Event stream
Állapot	State
Állapot alapú modell	State based model
Állapottér	State space
Kölcsönösen kizárólagos	Mutually exclusive
Teljes	Complete
Állapotmentes	Stateless

Magyar - English

Állapotterek szorzata	Product of state spaces
Állapotváltozó	State variable
Összetett	Composite
Állapottér-robbanás	State space explosion
Véges	Finite
Diszkrét („szétválasztható”)	Discrete
<i>Diszkrét („nem pletykál”)</i>	<i>Discreet</i> 😊
Állapotátmenet	Transition
Állapotátmeneti szabály	Transition rule
Nemdeterminizmus / Konfliktus	Nondeterminism / Conflict

Magyar - English

Állapotgép / Automata	State machine / Automaton
Kezdőállapot	Initial state
Címke	Label
Ok / Előfeltétel	Cause / Precondition
Következmény / Utófeltétel	Consequence / Postcondition
Hurokél	Loop edge
Csatorna	Channel
Jel / Szimbólum	Signal / Token / Symbol
Nyelő / Csapda	Sink / Trap
Szinkron szorzat	Synchronous product
Aszinkron szorzat	Asynchronous product