Állapot alapú modellezés

Rendszermodellezés 2020.

Budapesti Műszaki és Gazdaságtudományi Egyetem
Méréstechnika és Információs Rendszerek Tanszék

Tartalom

- Előismeretek
- Állapotterek
- Egyszerű (Mealy) állapotgépek
- Összetett (Harel) állapotgépek
- Kitekintés, szoftverek

ELŐISMERETEK

Felépítési és viselkedési modellezés

- Felépítési (*structural*)
  - Statikus
  - Rész és egész, összetevők
  - Kapcsolatok, összeköttetések
- Viselkedési (*behavioral*)
  - Dinamikus
  - Időbeli lefolyás
  - Állapot, folyamat
  - Reakciók a külvilágra

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

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

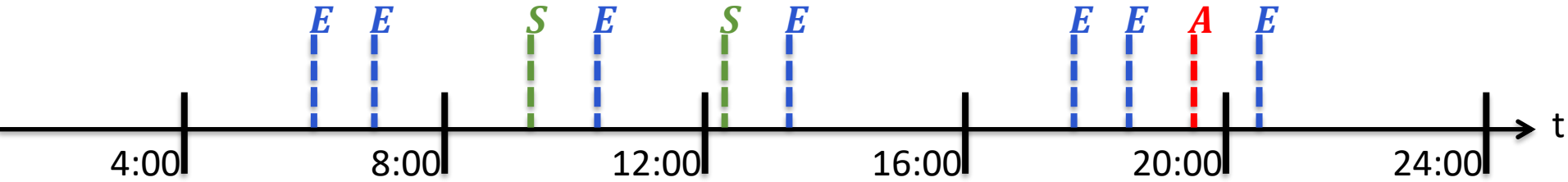
Alapozás: diszkrét eseménysor

- **Esemény**
  - pillanatszerű változás (a rendszerben vagy ki/bemeneten)
  - (Később: folyamatbeli esemény, erőforrás-esemény)
- **Eseményfolyam**
  - Pl. input/output adatforrás
- **Eseménytér** – {megengedett események}
  - Beolvasható input értékek, kibocsátható output értékek

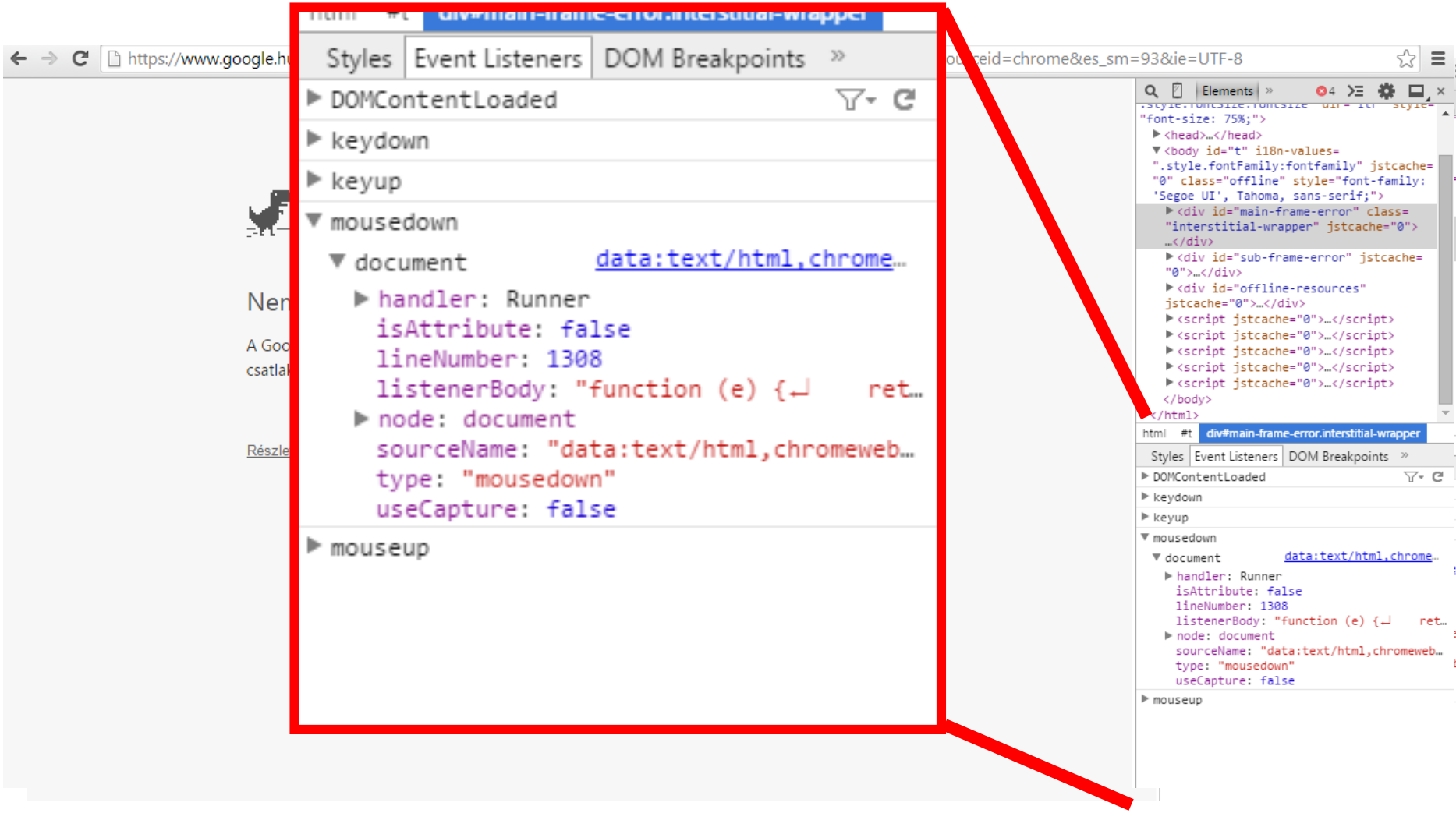
- Pillanatszerű események sora, egyszerre ≤ 1

Eseményfolyam: telefon értesítési jelzései

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



Eseményorientált programozás



ÁLLAPOTTEREK

Definíció: Állapottér

Az **állapottér**

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

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!)
    - {*hétfő*, *kedd*, *csütörtök*, *szombat*} nem teljes

- Kölcsönös **kizárólagosság**
  - Egyszerre legfeljebb egy állapot állhat fent
  - Ellenpéldák (nem alkalmasak állapottérnek!)
    - {*hétköznap*, *hétvége*, *délután*} nem kizárólagos (holott teljes)
    - mikró {*ajtaja tárva*, *kikapcsolva*}

Miért fontosak ezek a tulajdonságok?

- Szökőnap a Düsseldorfi Reptéren

https://www.dailymail.co.uk/travel/travel_news/article-3472837/Hundreds-passengers-arrive-destinations-without-luggage-airport-sorting-device-REFUSES-work-didn-t-recognise-leap-year.html

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…)

Finomított halmaz: B

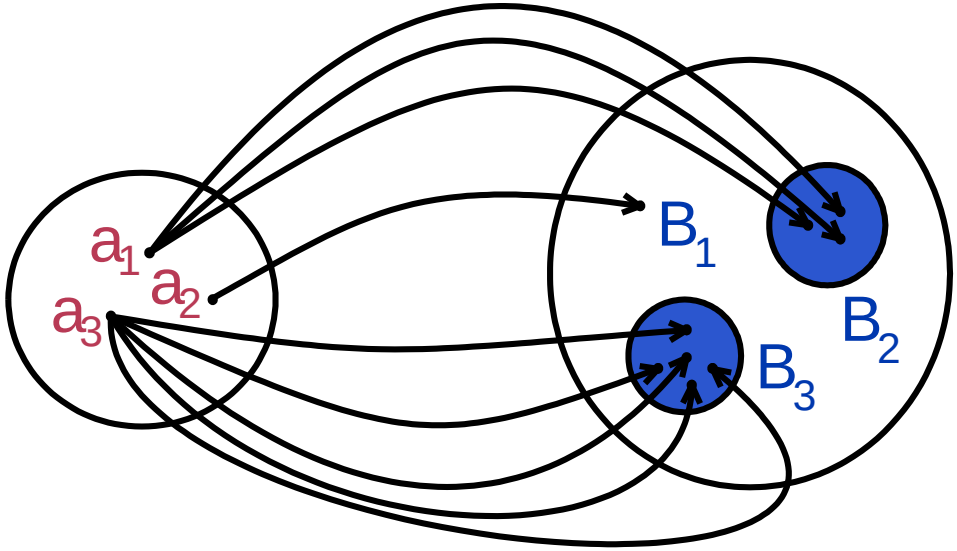
Absztrakt halmaz: A

Absztrakció: $f_a: B \to A$

- f_a szürjektív függvény
- Pl. $f_a(b_i) = a_3$ (minden $b_i \in B_3$)

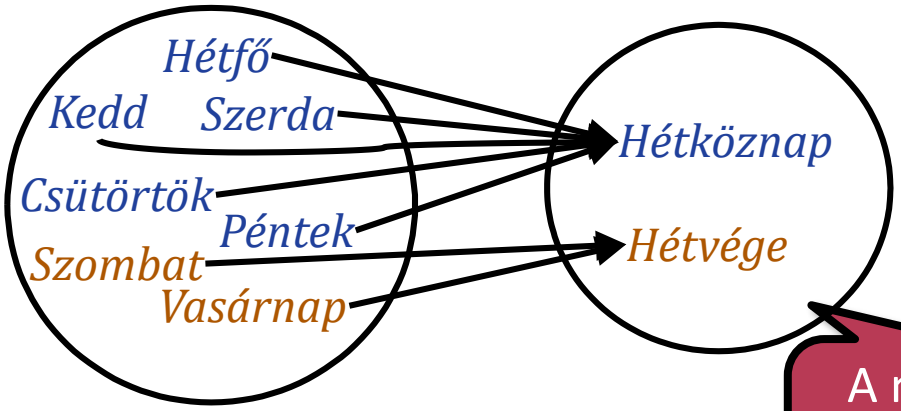
Finomítás: $f_f: A \to 2^B$

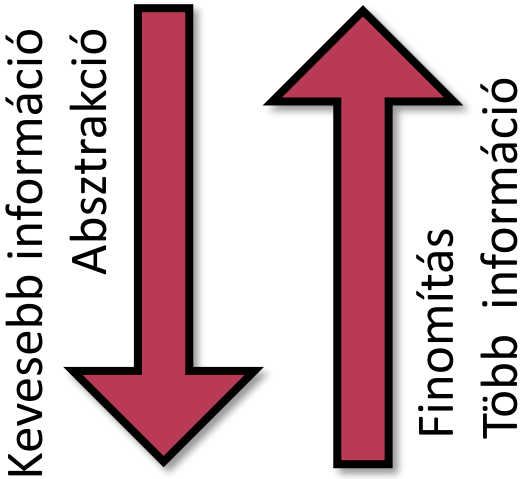
- f_f minden elemet halmazra képez
- Pl. $f_f(a_1) = B_2$



Állapotfinomítás, állapotabsztrakció

- Állapotfinomítás/absztrakció: az állapottéren végzett halmazfinomítás/absztrakció





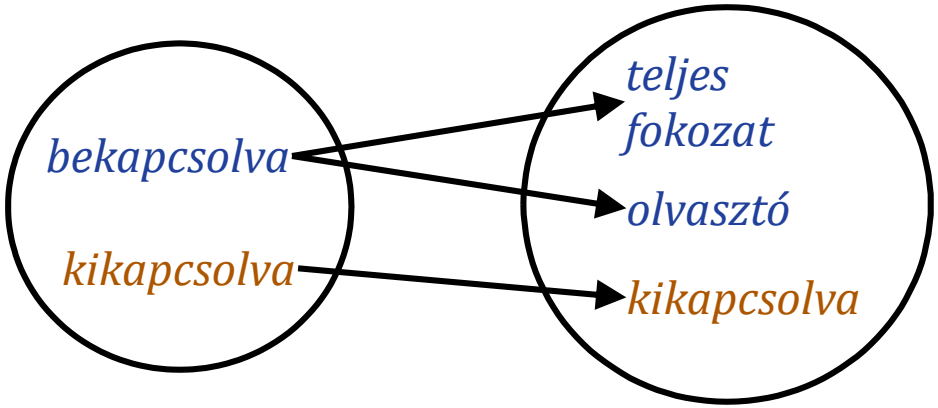
A menetrend tervezőjét csak ennyi érdekli

IGAZ-E ?

  - Mikró 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?

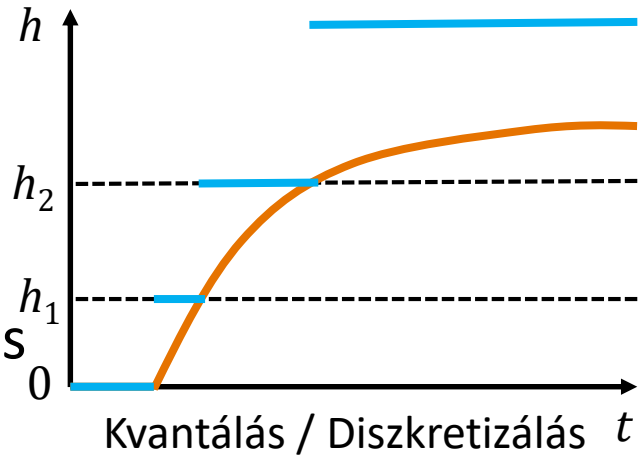


Absztrakció motivációja

- 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
    - Szélsőséges eset: **állapotmentes** modell ($|S| = 1$)
  - Néha csak kevesebb információt szabad felfedni
- Gyakori formája: **dekompozíció** (ld. később)

Folytonos változók

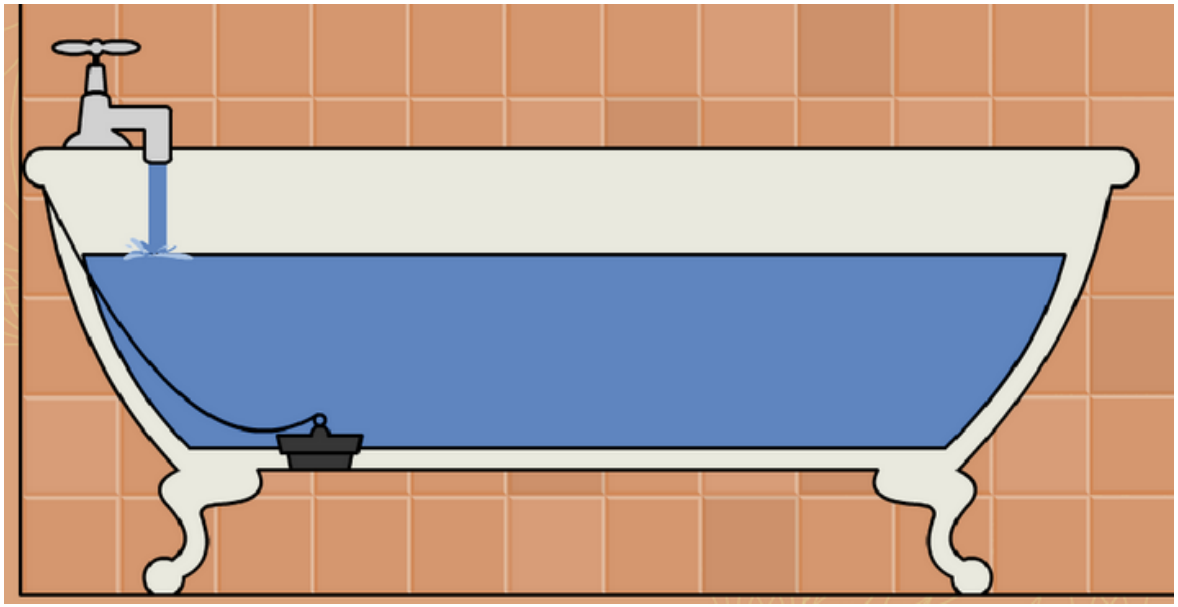
- Potenciálisan végtelen (akár kontinuum) állapottér
  - Pl. a repülő állapotváltozói
    - $v \in \mathbb{R}$ sebesség
    - $h \in \mathbb{R}$ magasság
    - $\alpha \in [-\pi/2, \pi/2]$ emelkedési szög
  - Az állapot időbeli változása lehet folytonos
    - Pl. a repülő emelkedése: $\partial h / \partial t = v \sin \alpha$



- Ezzel szemben tipikus IT rendszermodellekben
  - **Diszkrét állapotok** (nincs köztük folytonos átmenet)
  - Gyakran **véges állapottér** (ellenpélda: számláló $\in \mathbb{N}$)
  - Pillanatszerű **állapotátmenetek,** köztük állandó állapot

Hibrid rendszerek

- Folytonos és diszkrét változások is
- Pl: fürdőkád
  - Diszkrét változók: csap {nyitva;zárva}, dugó {van;nincs}
  - Folytonos változó: vízszint - $0 \leq h \leq 60\ cm$
  - Diszkrét változók értéke hatással van a folytonos változó értékére



Predikátumabsztrakció

- (Folytonos) változó értékkészletének absztrakciója predikátumok szerint
- Pl: vízszint
  - 0 – kád alja
  - 40 – gyári „max” jelzés
  - 60 – kád pereme

Predikátumok

$h = 0, h > 0$

$h \leq 40, h > 40$

$h < 60, h = 60$

- Absztrakt állapottér: predikátumok kombinációi
  - $\{h = 0, h > 0 \wedge h \leq 40, h > 40 \wedge h \leq 60, h = 60\}$

Kád üres

Vízszint rendben

Vízszint kritikus

Kád túlcsordult

Predikátumabsztrakció

- Predikátumabsztrakció:
  - Adott állításhalmazból (predikátumhalmazból)
    - melyik állapotra
    - melyik állítás igaz
  - Azonos állítások igazak → egy absztrakt állapot

Teljes: minden állapot benne lesz egy csoportban

Kizárólagos: egy állítás nem lehet egyszerre igaz és hamis

A **predikátumabsztrakció** olyan absztrakció, amely

- a konkrét állapotokat
- logikai állításokkal (**predikátumokkal**) halmazokba csoportosítja
- és a halmazokat az elemeiken teljesülő állításokkal jellemzi.

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 → a teljes rendszerállapot ismerete nélkül eldönthető, melyik áll fenn

Állapotterek (direkt) szorzata

- Két állapottér együttes figyelembevétele
  - S_1 = {*teljes fokozat, olvasztó mód, kikapcsolva*}
  - S_2 = {*nyitva, zárva*}

Állapotváltozó: vektor

$S_1 \times S_2$	*nyitva*	*zárva*
teljes	*teljes* és *nyitva*	*teljes* és *zárva*
olvasztó	*olvasztó* és *nyitva*	*olvasztó* és *zárva*
kikapcsolva	*kikapcsolva* és *nyitva*	*kikapcsolva* és *zárva*

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.

Mi történik 24 órás számláló esetén?

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

$$\Downarrow$$

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

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.

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

$$\Downarrow$$

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

$$\Downarrow$$

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

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.

Eredeti halmaz: $B \subseteq D_1 \times \cdots \times D_n$ (n állapotváltozó, D_j ért. tartománnyal)

Megtartott változók: $V \subseteq \{x_{i_1}, \ldots, x_{i_m}\}$ ($m < n$ változó)

Absztrakt halmaz: $A \subseteq D_{i_1} \times \cdots \times D_{i_m}$ (m állapotváltozó)

Vetítés: $f_V : B \to A$

$$f_V([v_1, \ldots, v_n]) = [v_{i_1}, \ldots, v_{i_m}]$$

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.

Eredeti halmaz: $B \subseteq \mathbb{N} \times \mathbb{N}$ $(x_1 \in \mathbb{N}, x_2 \in \mathbb{N})$

Megtartott változók: $V \subseteq \{x_1\}$

Absztrakt halmaz: $A \subseteq \mathbb{N}$

Vetítés: $f_V : B \to A$

$f_V([v_1, v_2]) = [v_1]$ *Pl.* $f_V([42, 5]) = [42]$

Állapotváltozók finomított kompozíciója

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

$S \subseteq S_1 \times S_2$	*nyitva*	*zárva*
teljes	*teljes* és *nyitva*	*teljes* és *zárva*
olvasztó	*olvasztó* és *nyitva*	*olvasztó* és *zárva*
kikapcsolva	*kikapcsolva* és *nyitva*	*kikapcsolva* és *zárva*

állapotabsztrakció

állapotabsztrakció (vetítés)

Észrevétel: a vetítés mint absztrakciós viszony megmarad

Állapotváltozók finomított kompozíciója

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

A **potenciális állapottér** olyan állapottér, amely a **valódi állapotteret** tartalmazza.

Vegyük észre, hogy a potenciális állapottér a valódi állapottér **absztrakciója:**

- „elfelejtjük”, hogy egy állapot tényleg benne van-e az állapottérben vagy sem
- a direkt szorzat többnyire kompaktabban megadható
  - kisebb információtartalom: *n+m* elem (állapotváltozók értékkészletei) jellemzően kevesebb, mint *n*m* (szorzat értékkészlete)

Még egyszer a finomításról

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

Tanulság:
finomításoktól nőhet is, csökkenhet is az állapottér mérete

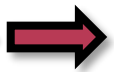
A lényeg, hogy **többet tudunk a rendszerről**

Avagy: **kevesebb rendszer illik a modellre**

$S \subseteq S_1 \times S_2$	*nyitva*	*zárva*
teljes	*teljes* és *nyitva*	*teljes* és *zárva*
olvasztó	*olvasztó* és *nyitva*	*olvasztó* és *zárva*
kikapcsolva	*kikapcsolva* és *nyitva*	*kikapcsolva* és *zárva*

Dekompozíció állapotváltozókra

- Dekompozíció: szorzat / kompozíció megfordítása
  - kikapcsolva és nyitva
  - kikapcsolva és zárva
  - olvasztó és zárva
  - *teljes és zárva*



S_1 = {*teljes*, *olvasztó*, kikapcsolva}
S_2 = {nyitva, zárva}

- A vetített állapotváltozók absztrakciók
- A direkt szorzatuk csak potenciális állapottér
- Miért dekomponálunk?
  - Állapotváltozók külön kezelhetőek
  - Állapotváltozók külön tárolhatóak

Predikátumabsztrakció állapotvektorokkal

- A vetítés és kompozíció speciális műveletek
  - Egy komponens „elfelejtése” vagy „figyelembe vétele”
- Predikátumabsztrakció (állapotvektorokkal):

Eredeti halmaz: $B \subseteq D_1 \times \cdots \times D_n$ (n állapotváltozó, D_j ért. tartománnyal)

Predikátumhalmaz: $P = \{p_1, \ldots, p_m\}$ (m predikátum)

Absztrakt halmaz: $A \subseteq \{\text{igaz}, \text{hamis}\}^m$ (m hosszú bitvektor)

Predikátumabsztrakció: $f_P : B \to A$

Állapotvektor a konkrét értékekkel

Az állítás igazságértéke, ha a változók helyére behelyettesítjük az állapotot

$$f_P([v_1, \ldots, v_n]) = [p_1\langle x_1 = v_1, \ldots, x_n = v_n\rangle, \ldots, p_m\langle x_1 = v_1, \ldots, x_n = v_n\rangle]$$

Állapotváltozó

Állapot

Az állítások igazságértékeinek vektora

Predikátumabsztrakció állapotvektorokkal

- A vetítés és kompozíció speciális műveletek
  - Egy komponens „elfelejtése” vagy „figyelembe vétele”
- Predikátumabsztrakció (állapotvektorokkal):

Eredeti halmaz: $B \subseteq \mathbb{R} \times \mathbb{R} \quad (x_1 \in \mathbb{R}, x_2 \in \mathbb{R})$

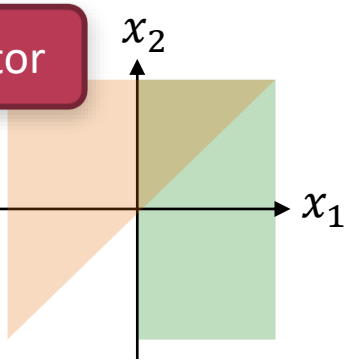
Kontinuum végtelen koordináta

Predikátumhalmaz: $P = \{x_1 < x_2, x_1 > 0\}$

Absztrakt halmaz: $A \subseteq \{\text{igaz}, \text{hamis}\} \times \{\text{igaz}, \text{hamis}\}$

Predikátumabsztrakció: $f_P : B \to A$

4 lehetséges bitvektor



Lehet olyan állapot, ami biztosan nem releváns?

$f_P([v_1, v_2]) = [v_1 < v_2, v_1 > 0]$

$f_P([\pi, 3]) = [\pi < 3, \pi > 0] = [\text{hamis}, \text{igaz}] \sim [0,1]$

Ez is egy állapotvektor…

Predikátumabsztrakció állapotvektorokkal

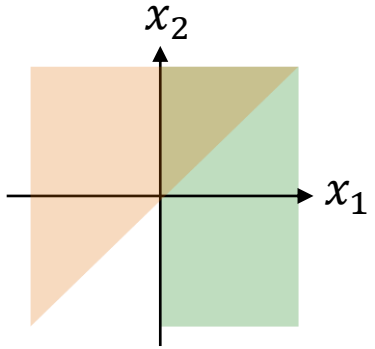
- A vetítés és kompozíció speciális műveletek
  - Egy komponens „elfelejtése” vagy „figyelembe vétele”
- Predikátumabsztrakció (állapotvektorokkal):

Eredeti halmaz: $B \subseteq \mathbb{R} \times \mathbb{R} \quad (x_1 \in \mathbb{R}, x_2 \in \mathbb{R})$

Predikátumhalmaz: $P = \{x_1 < x_2, x_1 > 0\}$

Absztrakt halmaz: $A \subseteq \{\text{igaz}, \text{hamis}\} \times \{\text{igaz}, \text{hamis}\}$

Predikátumabsztrakció: $f_P : B \to A$



$$f_P([v_1, v_2]) = [v_1 < v_2, v_1 > 0]$$

$$f_P([\pi, 3]) = [\pi < 3, \pi > 0] = [\text{hamis}, \text{igaz}] \sim [0,1]$$

Predikátumabsztrakció állapotvektorokkal

- A vetítés és kompozíció speciális műveletek
  - Egy komponens „elfelejtése” vagy „figyelembe vétele”
- Predikátumabsztrakció (állapotvektorokkal):

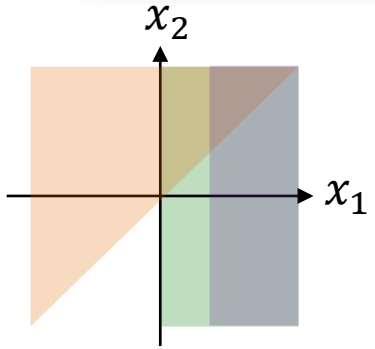
Eredeti halmaz: $B \subseteq \mathbb{R} \times \mathbb{R} \quad (x_1 \in \mathbb{R}, x_2 \in \mathbb{R})$

Predikátumhalmaz: $P = \{x_1 < x_2, x_1 > 0, \boldsymbol{x_1 > 1}\}$

Absztrakt halmaz: $A \subseteq \{\mathrm{i}, \mathrm{h}\} \times \{\mathrm{i}, \mathrm{h}\} \times \{\mathbf{i}, \mathbf{h}\}$

$\neg(x_1 > 0)$ és $x_1 > 1$ sohasem lehet

Predikátumabsztrakció: $f_P : B \to A$



$$f_P([v_1, v_2]) = [v_1 < v_2, v_1 > 0, \boldsymbol{v_1 > 1}]$$

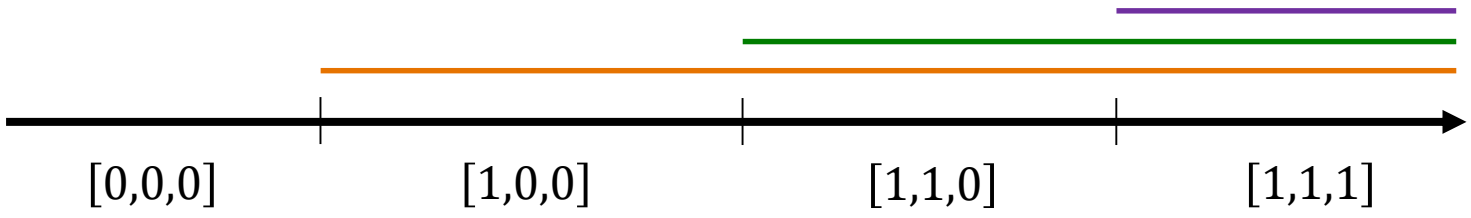
$$f_P([\pi, 3]) = [\pi < 3, \pi > 0, \boldsymbol{\pi > 1}] = [\mathrm{h}, \mathrm{i}, \mathbf{i}] \sim [0,1,\mathbf{1}]$$

Intervallumok és predikátumabsztrakció

- Az intervallumokra bontás felfogható
  - Predikátumabsztrakció és
  - Finomított kompozit állapottér kombinációjaként

Predikátumhalmaz: $P = \{x > 0, x > 5, x > 10\}$

Absztrakt halmaz: $A \subseteq \{[0,0,0], [0,0,1], \dots, [1,1,1]\}$



Nem értelmes: $\{[0,0,1], [0,1,0], [0,1,1], [1,0,1]\}$

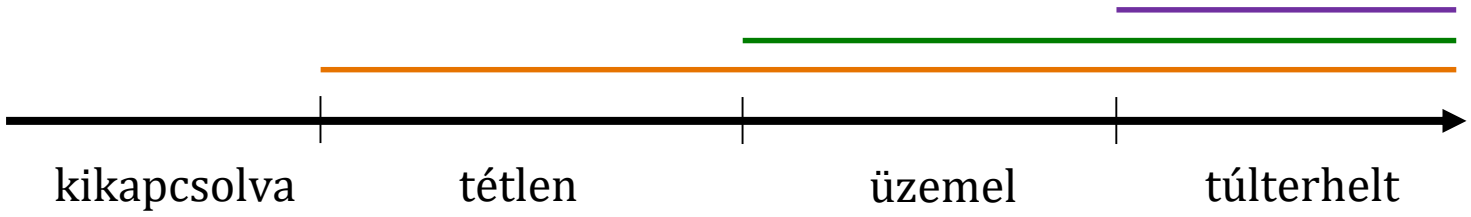
- Gyakori még:
  - Minden predikátum pontosan egy intervallumban igaz

Intervallumok és predikátumabsztrakció

- Az intervallumokra bontás felfogható
  - Predikátumabsztrakció és
  - Finomított kompozit állapottér kombinációjaként

Predikátumhalmaz: $P = \{x > 0, x > 5, x > 10\}$

Absztrakt halmaz: $A = \{\text{kikapcsolva}, \text{tétlen}, \text{üzemel}, \text{túlterhelt}\}$



A predikátumok kombinációit gyakran nevezzük el a hétköznapokban, így gyakran finomítási lépés rájönni a mögöttes állításokra/változókra.

Pl. színek ↔ (mérhető) frekvencia, mint intervallumokra osztott változó

A modell a valóság absztrakciója?

1. A világ nagyrészének állapota leírható véges sok valós számmal → **Dekomponálás**
2. Ezek közül nem mind fontos → **Vetítés**
   - *Pl. (általában) nem érdekel egy processzor súlya*
3. Gyakran nem számít a pontos érték → **Predikátumabsztrakció**
   - *Pl. nem érdekel, mennyit fogyaszt, csak hogy be van-e kapcsolva*
4. Az állapotváltozók direkt szorzata csak a **Potenciális állapottér**
   - *Pl. ha magas a terhelése mindig sokat fogyaszt*

Kitekintés: szakmai példák

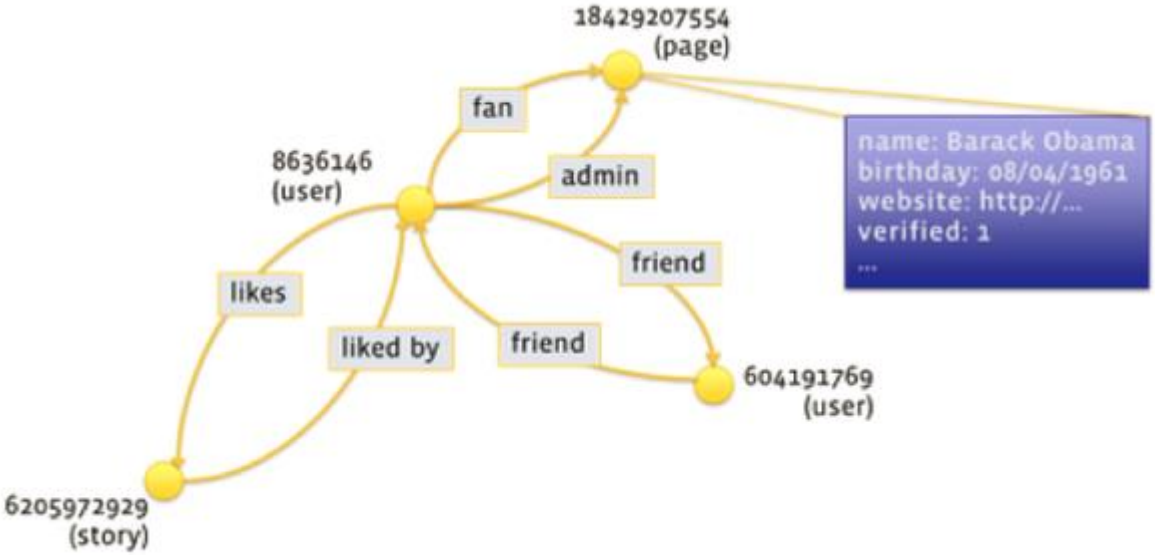
- Hol jön elő az állapot alapú modellezés az IT-ban?
- Közösségi háló – Jancsi és Juliska ismeretsége
  - (ettől függ néhány funkció, pl. milyen képek látszanak)
  - Állapotváltozók:
    - Jancsi bejelölte-e Juliskát
    - Vice versa

Tárolandó:
- Memóriában
- Adatbázisban (hosszabb távra)

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

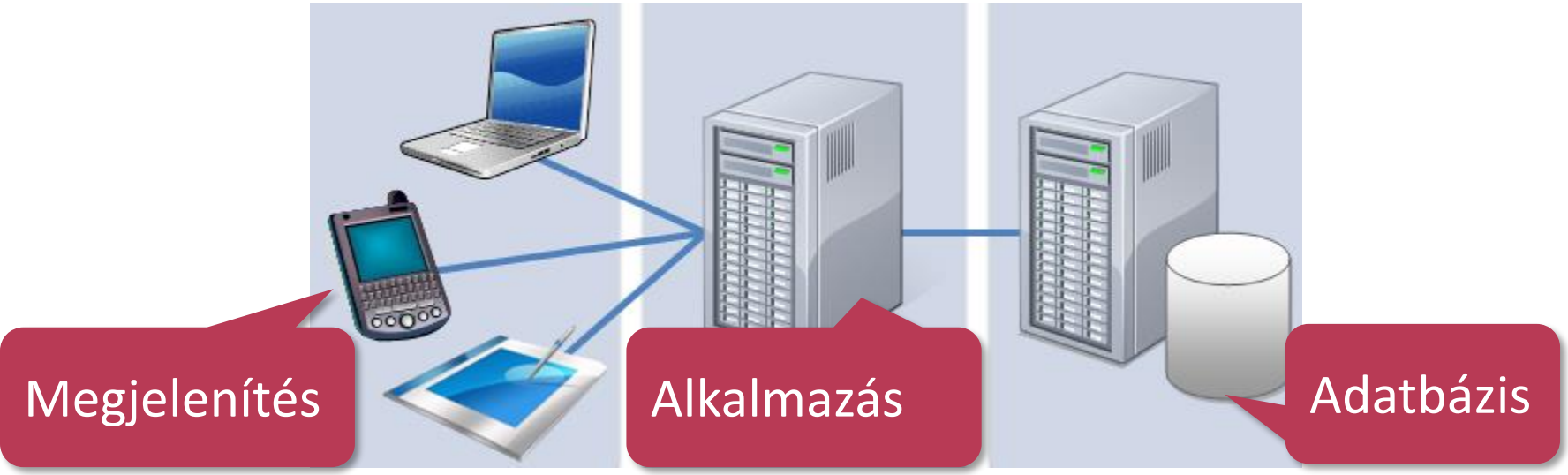
Kitekintés: szakmai példák

- Hol jön elő állapotabsztrakció?
- Közösségi háló: az adatbázis egy részét látom csak
  - Nincs szükségem arra, Jancsi ismeri-e Juliskát
    - Nincs is jogom tudni!
  - Jóval kisebb adatforgalom



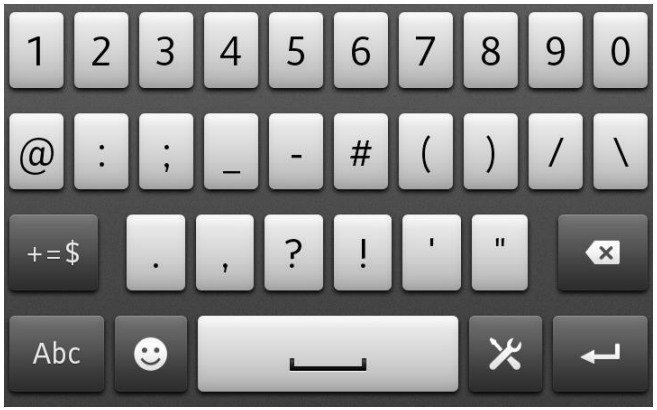
Kitekintés: szakmai példák

- Hol jön elő állapotabsztrakció?
- Szoftverrendszer dekompozíciója
  - Egyszerűbb a *megjelenítő réteg* (HTML + CSS + JavaScript), ha csak a nekem szánt információval dolgozik
  - Utána a különféle mobilklienseket is egyszerű lesz elkészíteni
  - A közösségi oldal mérnökei egyszer, egy helyen valósították meg a számomra releváns adatok kinyerését



Kitekintés: szakmai példák

- Virtuális billentyűzet érintőképernyőre
  - állapotváltozók?



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

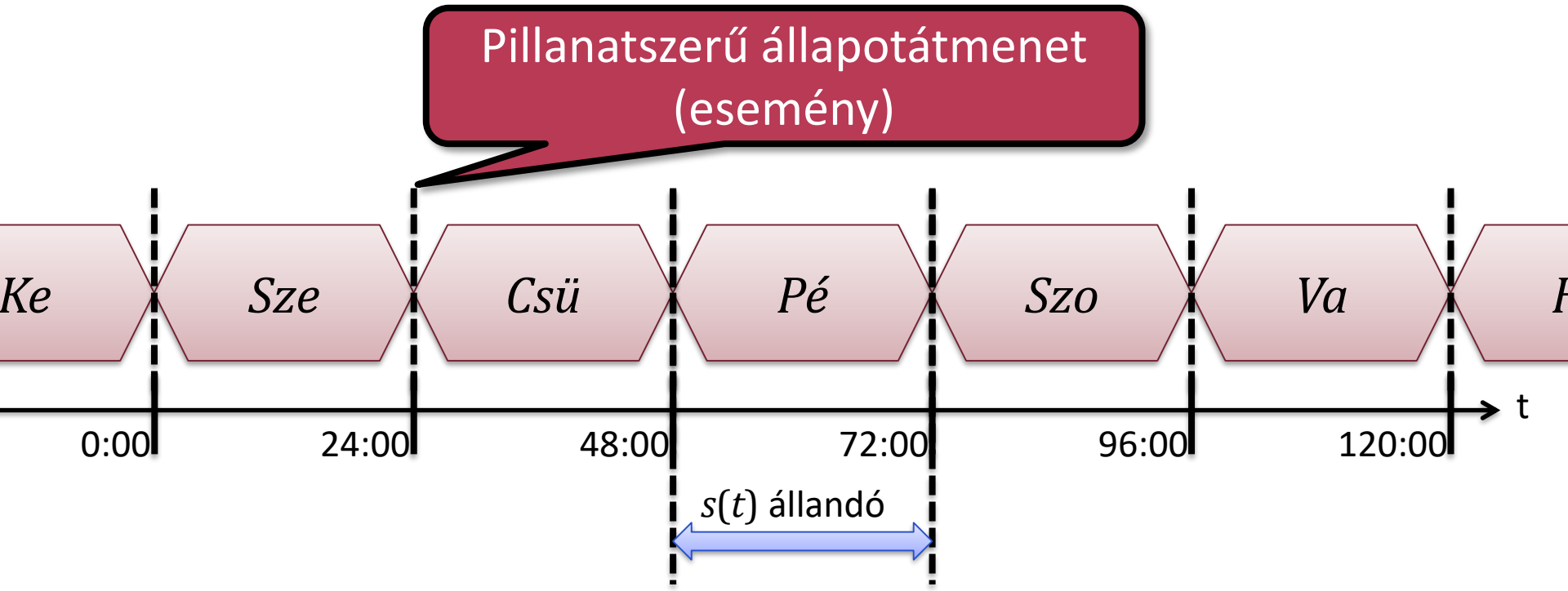
- Programozás: hogy tároljuk az állapotot?
  - Tömör megoldás: több állapotváltozóval

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

EGYSZERŰ (MEALY) ÁLLAPOTGÉPEK

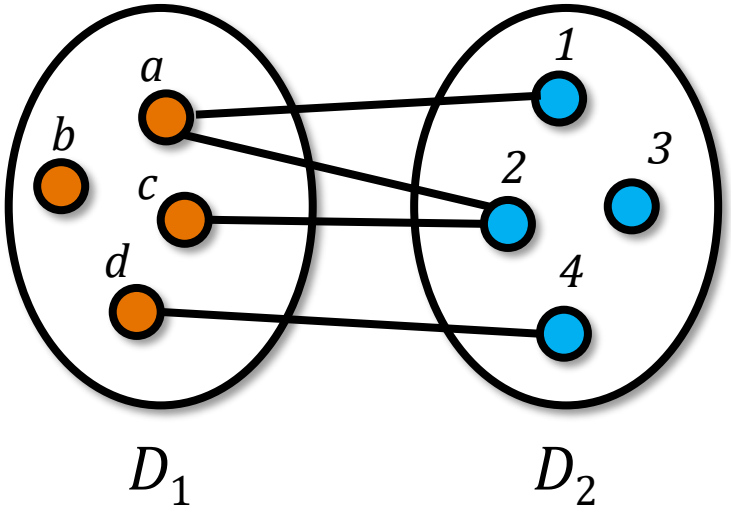
Állapotátmenetek

- 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
  - $R \subseteq D_1 \times D_2$



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

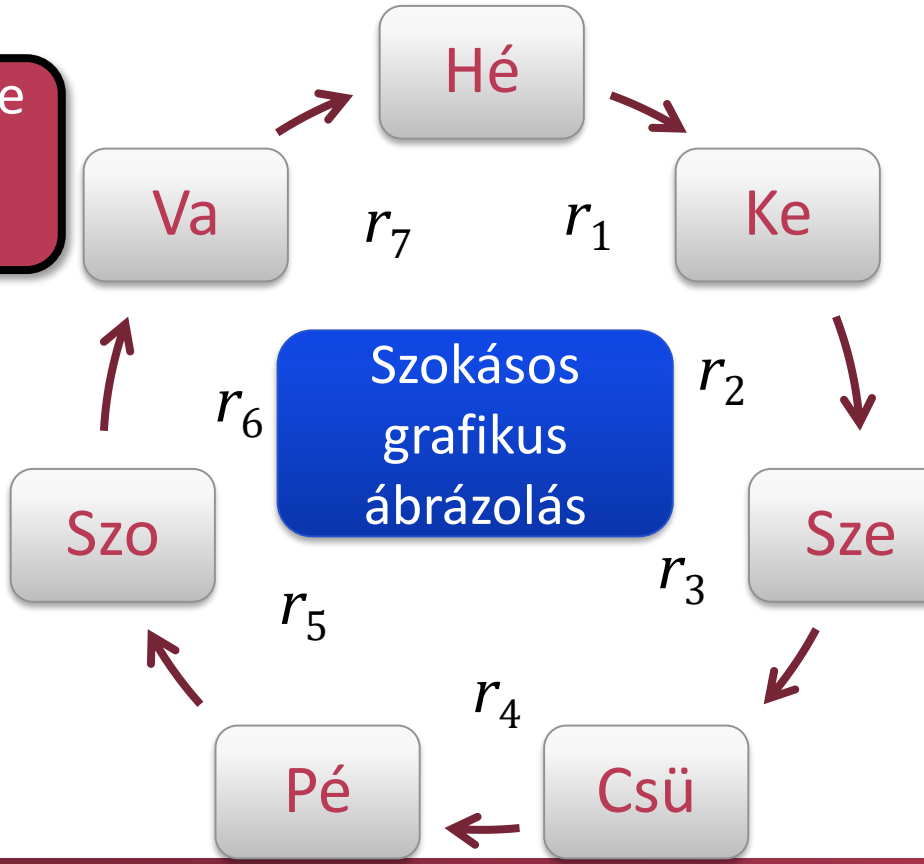
Állapotátmenetek

- Mely állapotokat mely állapotok követhetnek?
  - Állapottér $S = \{Hé, Ke, Sze, Csü, Pé, Szo, Va\}$
  - Eseménytér: **állapotátmeneti reláció** $R \subseteq S \times S$

Állapotátmeneti **szabályok** $r \in R$

- $r_1 = (Hé, Ke)$
- $r_2 = (Ke, Sze)$
- $r_3 = (Sze, Csü)$
- $r_4 = (Csü, Pé)$
- $r_5 = (Pé, Szo)$
- $r_6 = (Szo, Va)$
- $r_7 = (Va, Hé)$

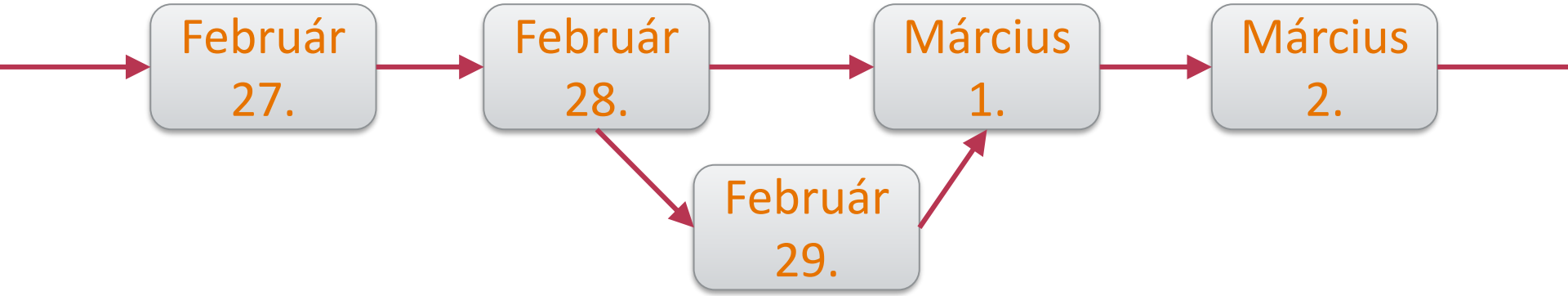
Csü-ből Pé-be át tud lépni a rendszer



  - Más néven: **állapotgráf**

Állapotátmenetek

- Mely állapotokat mely állapotok követhetnek?
  - Állapottér: az év napjai
  - Állapotgráf:



- Absztrakció: nem modellezzük, hogy mi az állapotátmenet kiválasztásának feltétele

Észrevételek az állapotgráfról

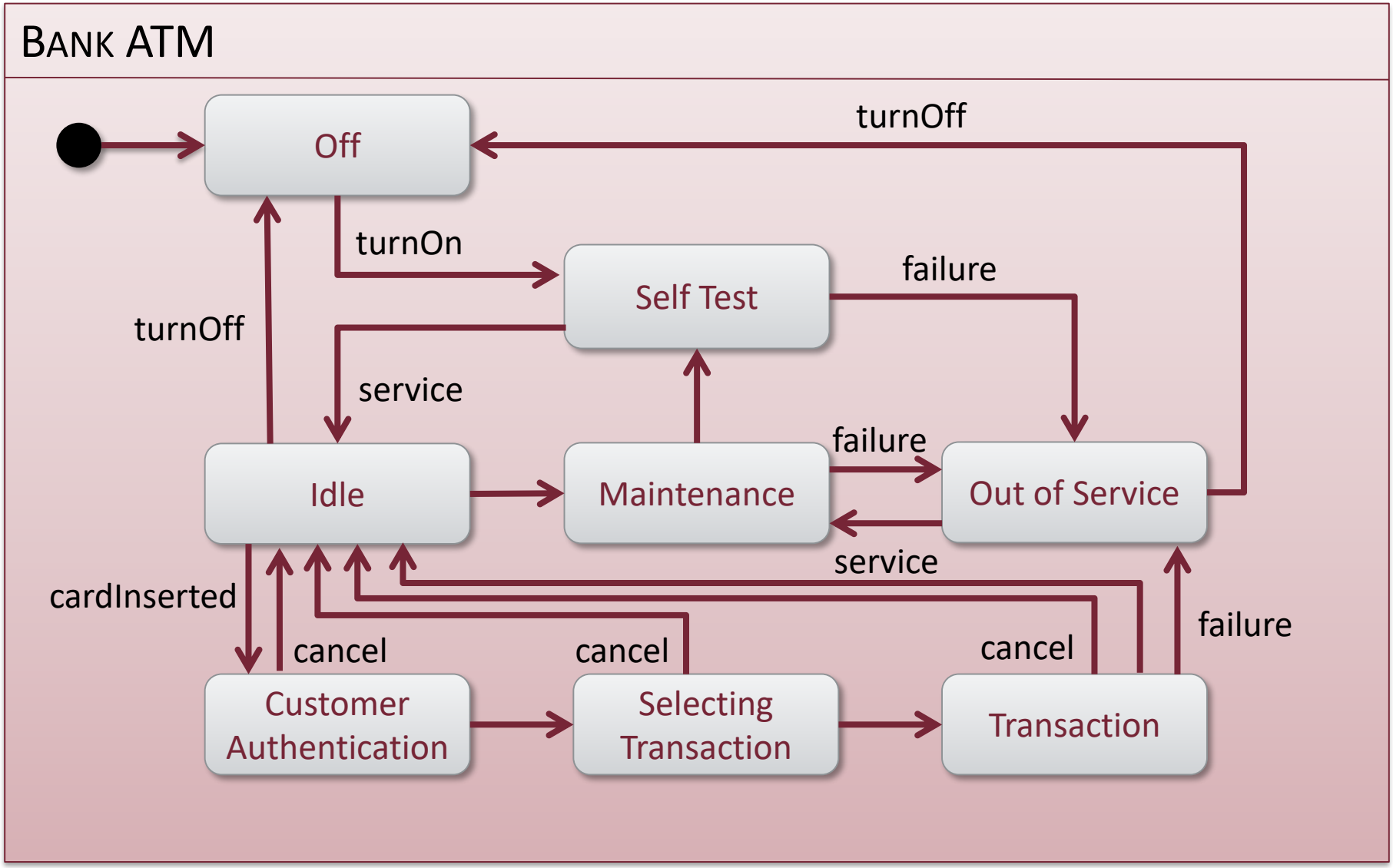
- Lehetséges, hogy…
  - … teljes gráf → minden átmenet engedélyezett
    - Pl. $S_{\text{mikró}}=\{kikapcsolva, bekapcsolva\}$
  - … nem minden állapotból érhető el az összes többi
    - Pl. $S_{\text{pohár}}= \{üres, teli, törött\}$ → nincs $törött \rightsquigarrow üres$ út
  - … bizonyos állapotoknak több rákövetkezője is van

Nemdeterminizmus

- Lehetséges eredetei:
  - A modellezett rendszer működése
  - Modellalkotás során bevezetett absztrakció

Pl. figyelembe nem vett belső változó, vezérlő input

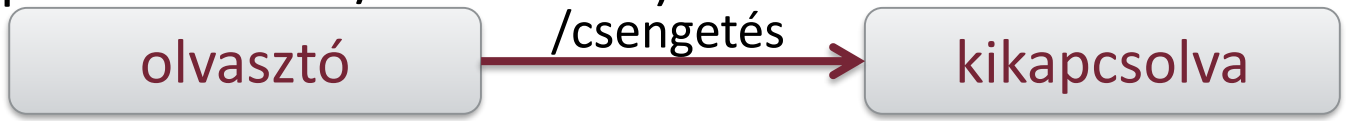
Állapotgráf példa: ATM



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)

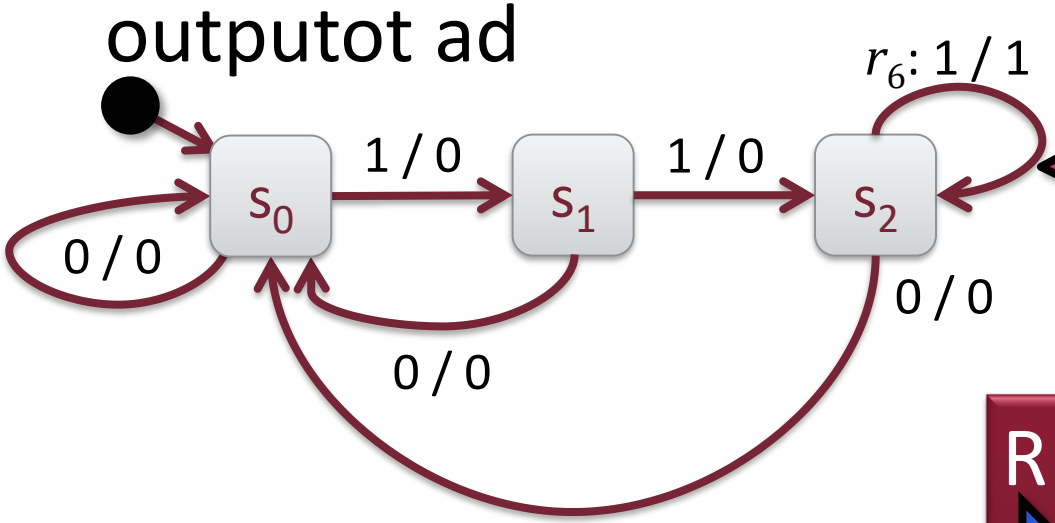


  - … az állapotátmenet oka (prekondíció/előfeltétel)

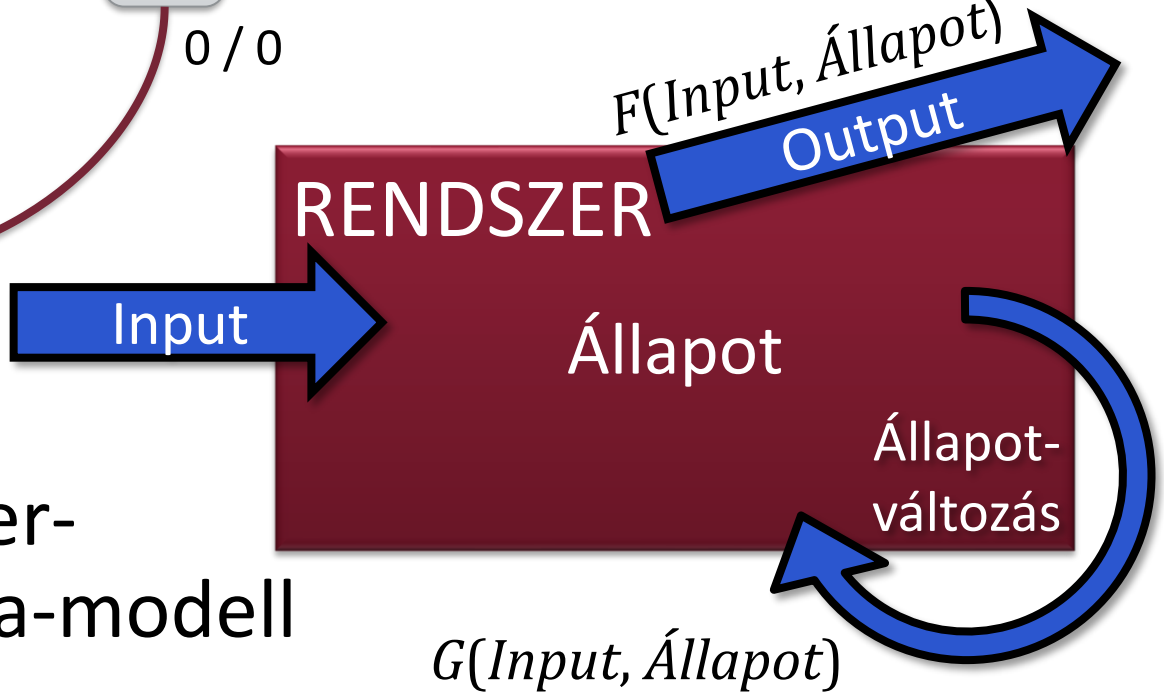
- Egy szabály címkézhető több eseménnyel is
  - input beolvasás / output kiírás

Mealy véges automata

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



Gondolkodtató:
- Mit csinál ez a gép?
- És ha r_6 szabály s_0-ba vinne?

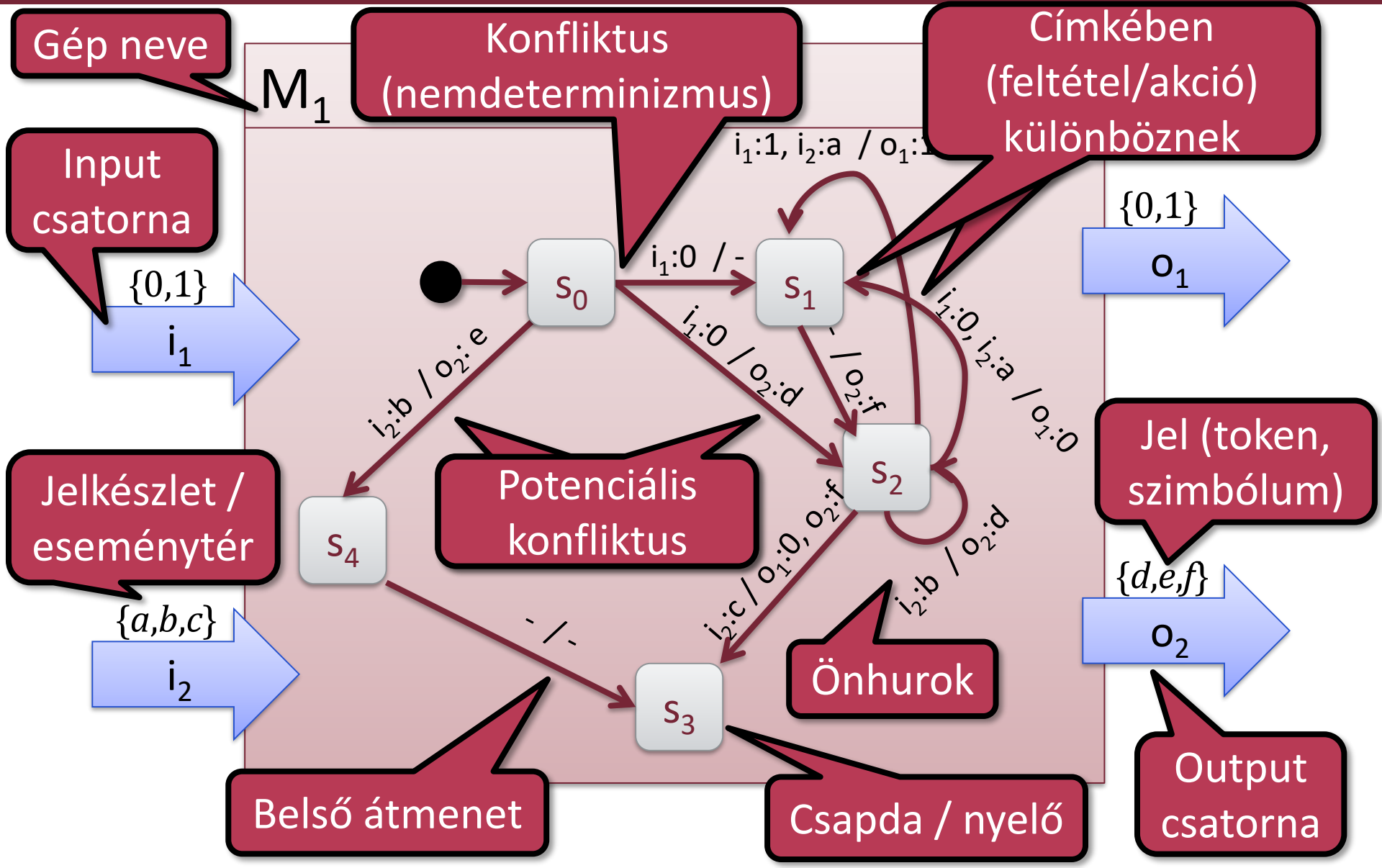


- Klasszikus rendszer-elméleti automata-modell

A Mealy-gép kiterjesztései

- Nemdeterminisztikus modell (vs. input)
- Input olvasás nélküli (**spontán**) lépés
  - Belső, nem modellezett esemény hatására
  - Pl. mikró 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

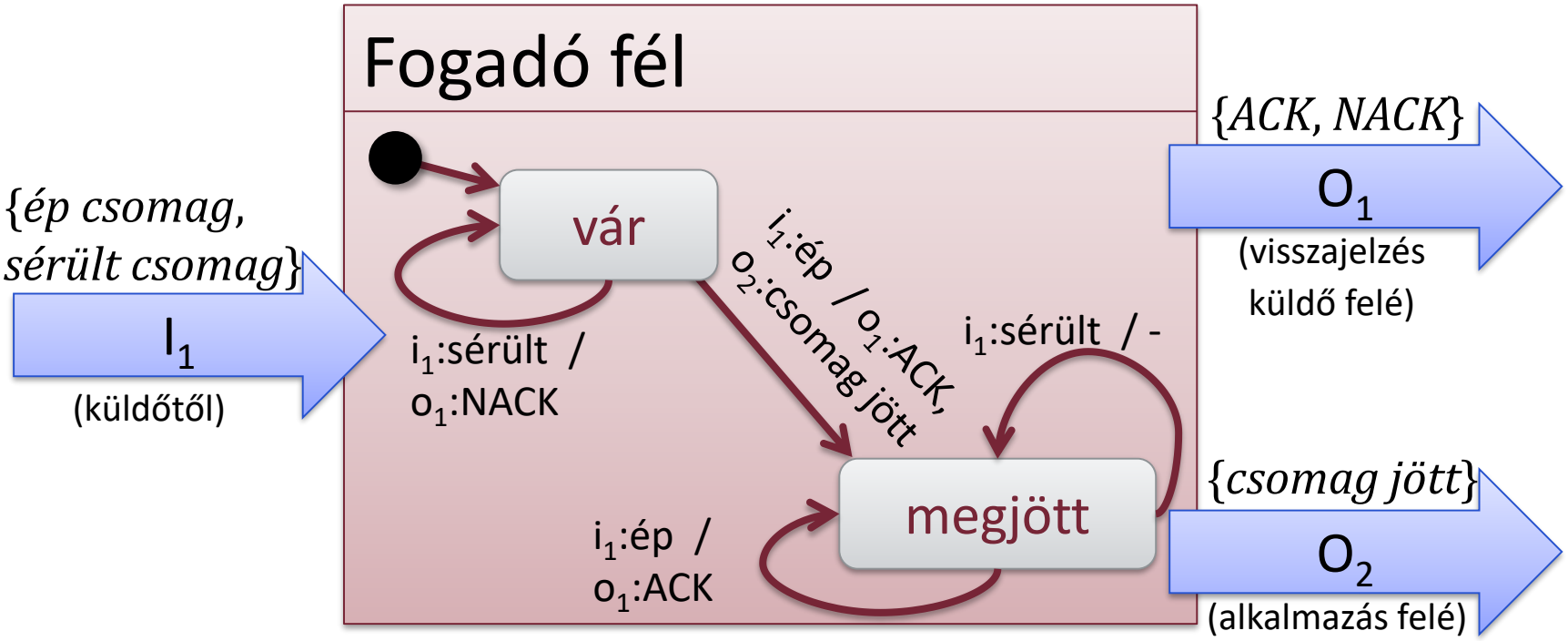


Specifikálatlan input kezelése

- 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

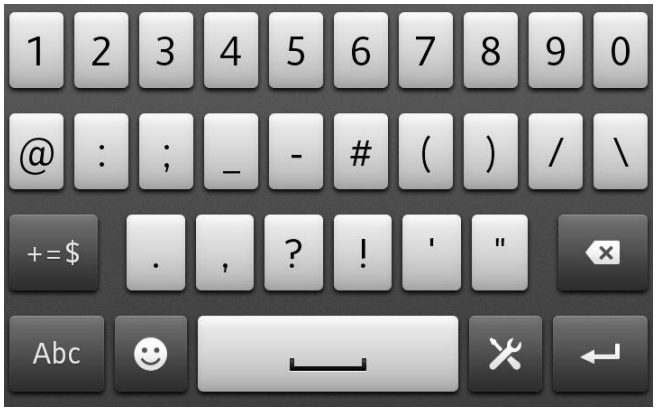
- Csomag alapú adatátviteli protokoll
  - Csomagok elveszhetnek, megsérülhetnek
  - Nyugtázás, újraküldés



- (Bővebben ld. Kommunikációs hálózatok 1. tárgy)

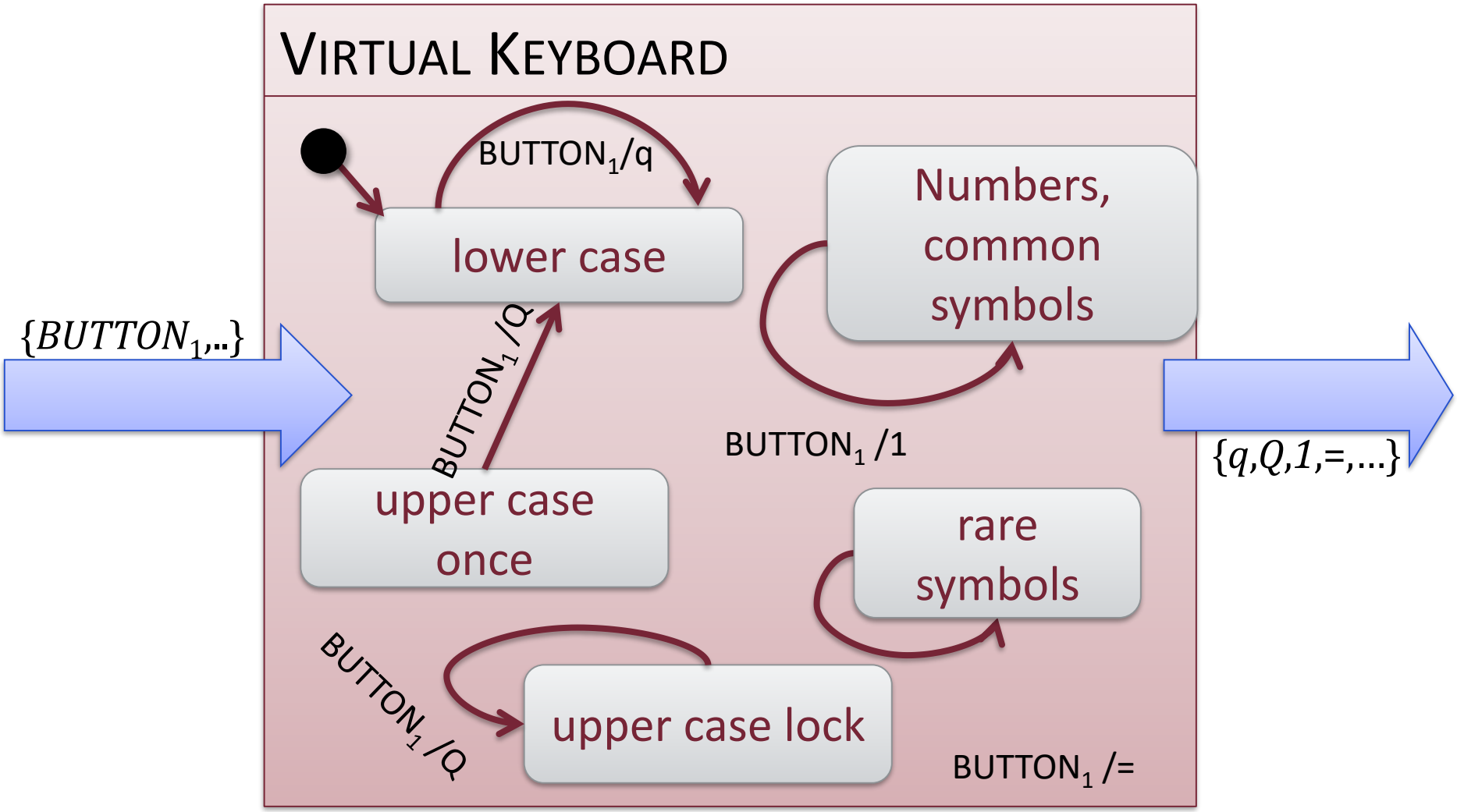
Emlékeztető

- Virtuális billentyűzet érintőképernyőre



Kitekintés: szakmai példák

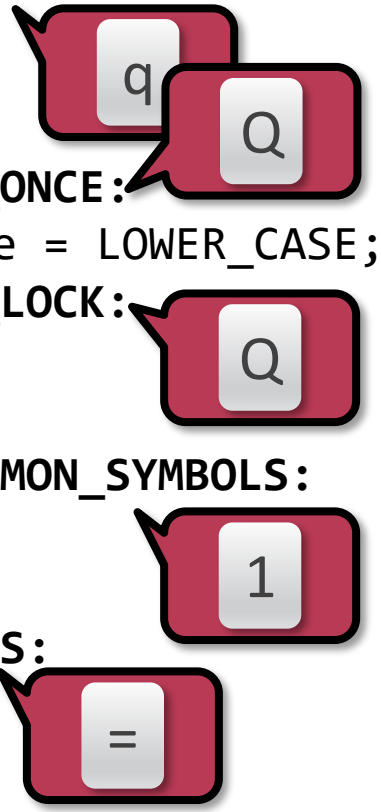
- Virtuális billentyűzet (részleges) állapotgépe



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:
        //...
```



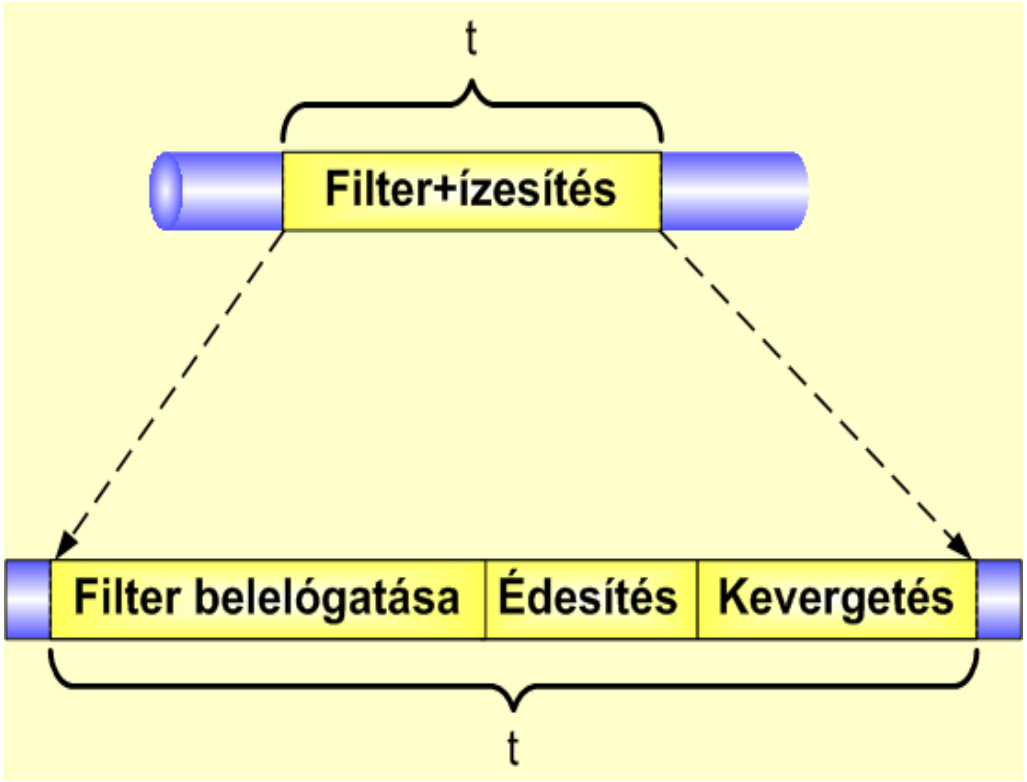
Kitekintés: szakmai példák

- 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

ÖSSZETETT (HAREL) ÁLLAPOTGÉPEK

Bevezetés: Hierarchikus modellfinomítás

- Elemi állapotokat több részállapotra bontunk fel
- Az új állapotokban tartózkodás összideje = a régi állapotban tartózkodás ideje



Kibontás
„egy az egyben”
behelyettesíthető
KOMPOZÍCIONALITÁS

Statechart nyelvek

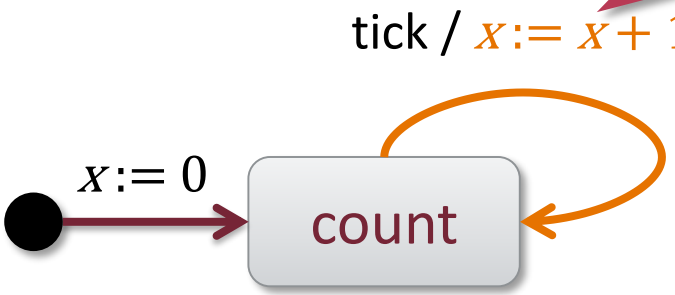
- **Statechart** (vagy Harel-féle összetett állapotgép)
  - Mealy-féle egyszerű állapotgép +
    - Változók
    - Pszeudoállapotok
    - Állapothierarchia
    - Ortogonalitás
    - …
- Pl.
  - Yakindu (HF)
  - UML (SzoftTech)
  - SysML (Informatikai rendszertervezés)

Változók

- Végtelen számláló
  - $S = \mathbb{N}$

- Változó bevezetése: x

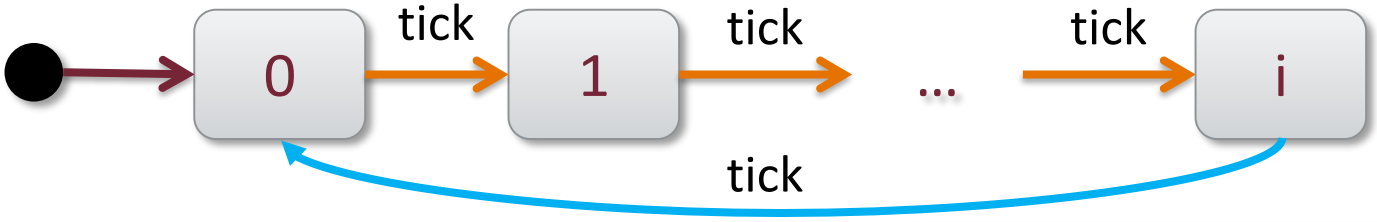
Valójában *x* odaképzelhető egy másik állapotrégióba…



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

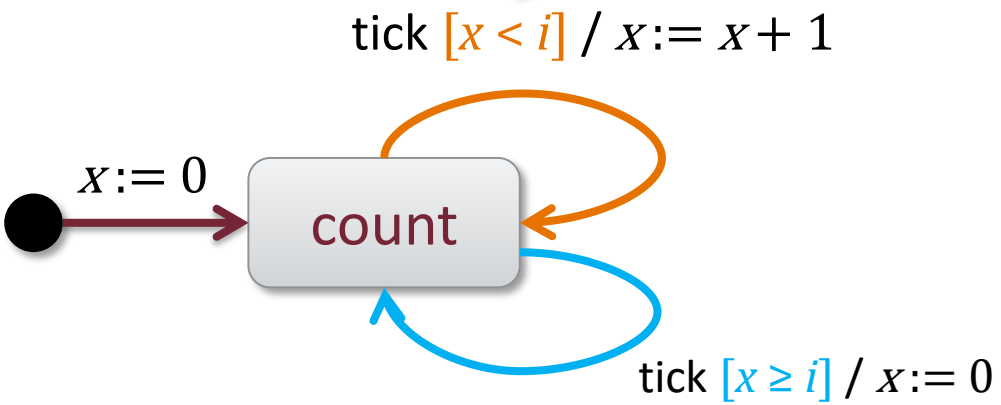
Változók + őrfeltételek

- Ciklikus számláló
  - $S = \{0,1, \ldots, i\}$



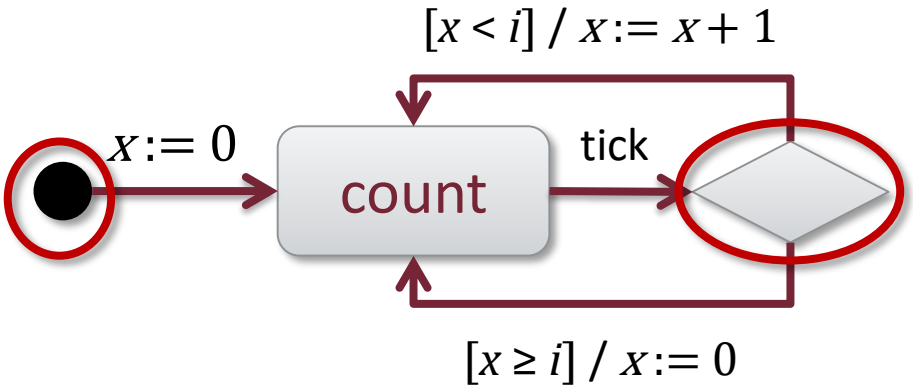
- Őrfeltételekkel:

Változó vagy állapotrégió figyelembe vehető



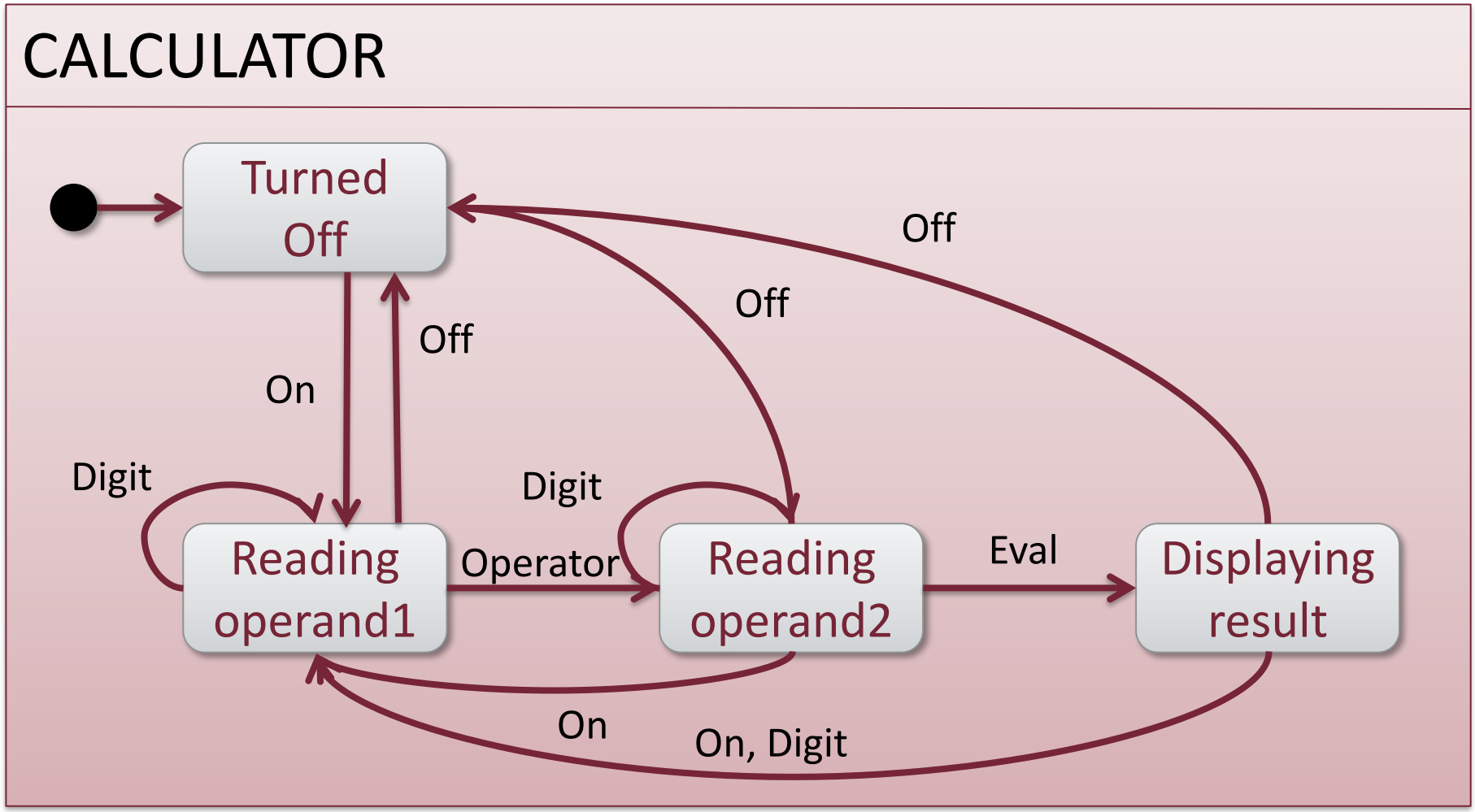
Pszeudoállapotok

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

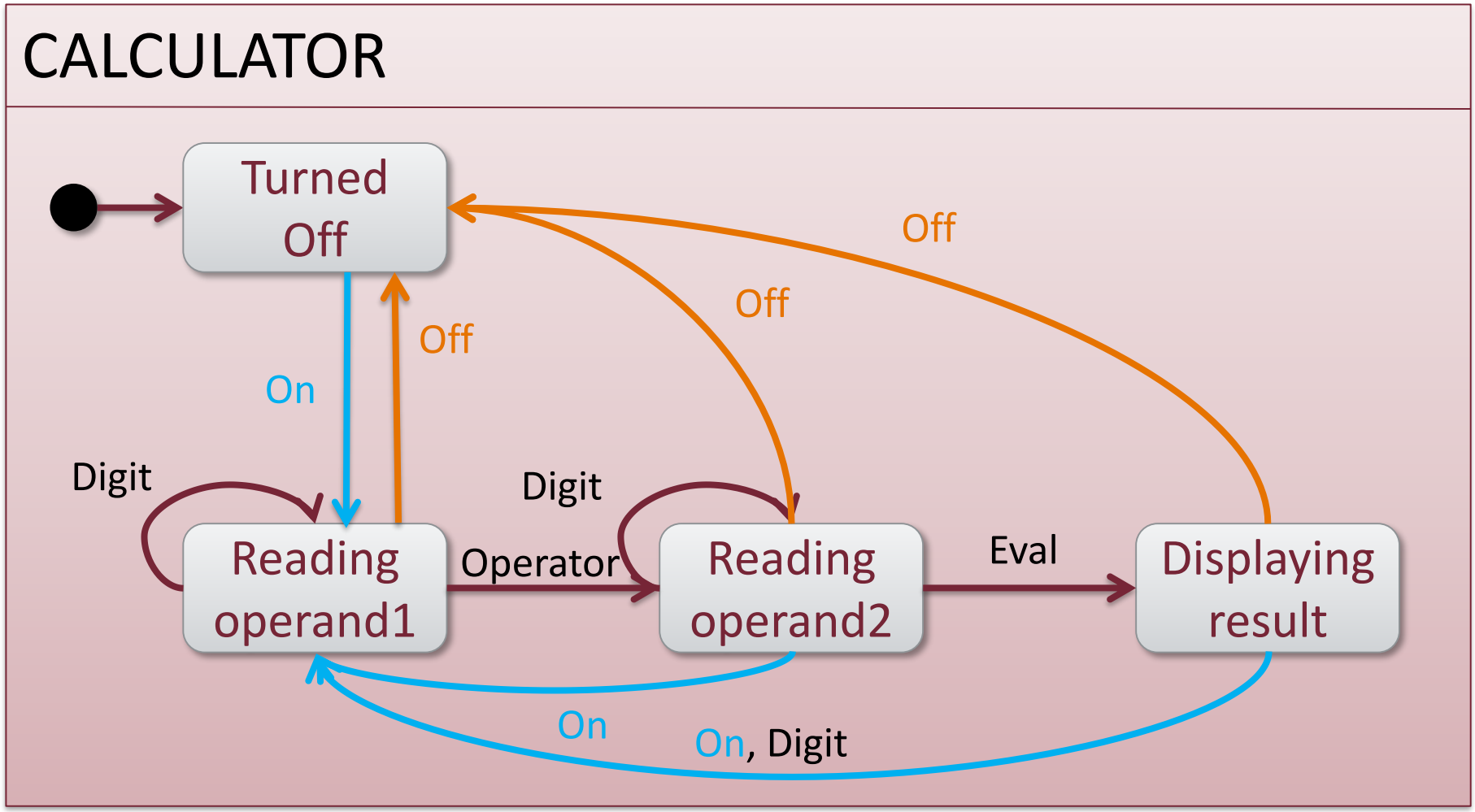


Számológép

Állapothierarchia

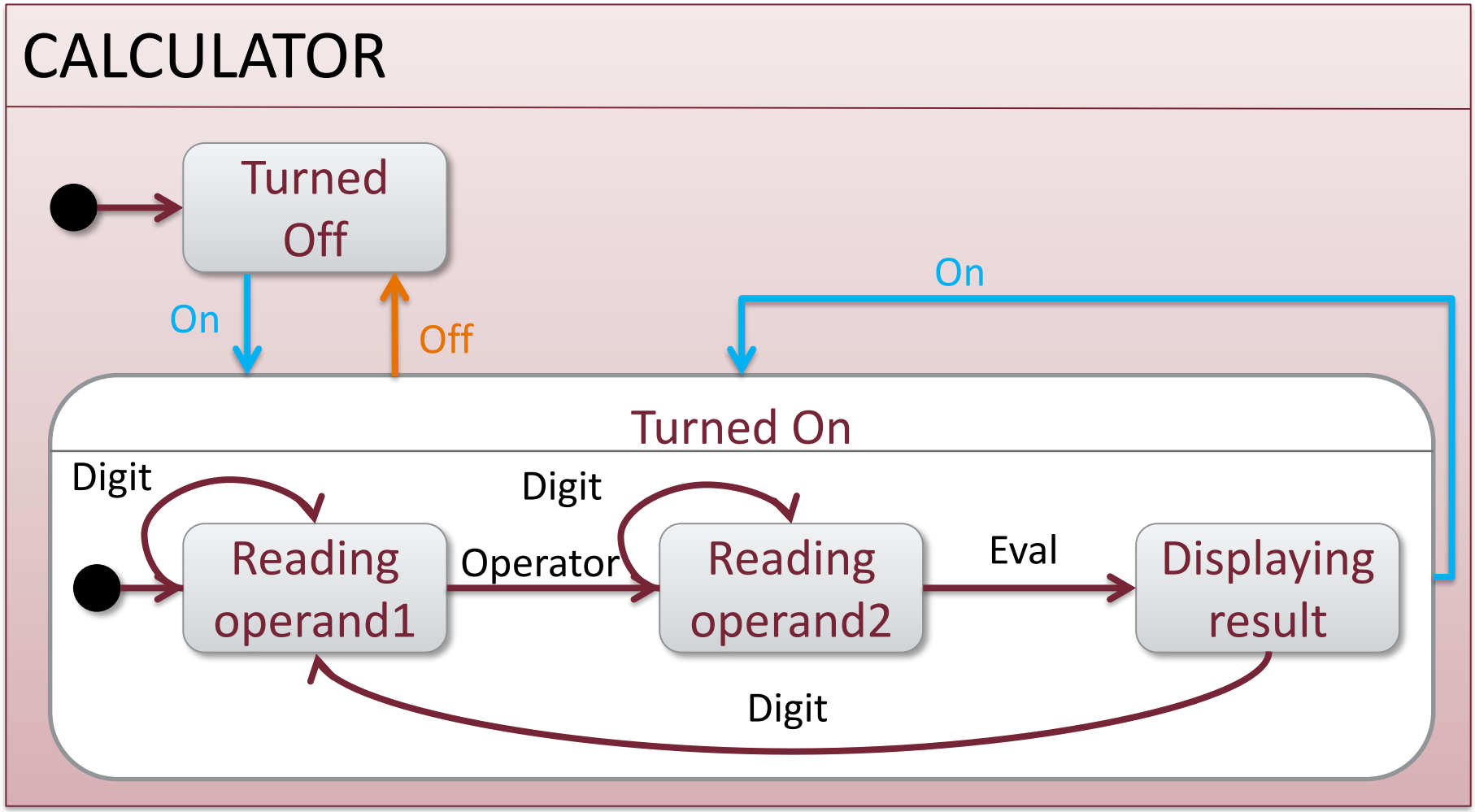


Állapothierarchia



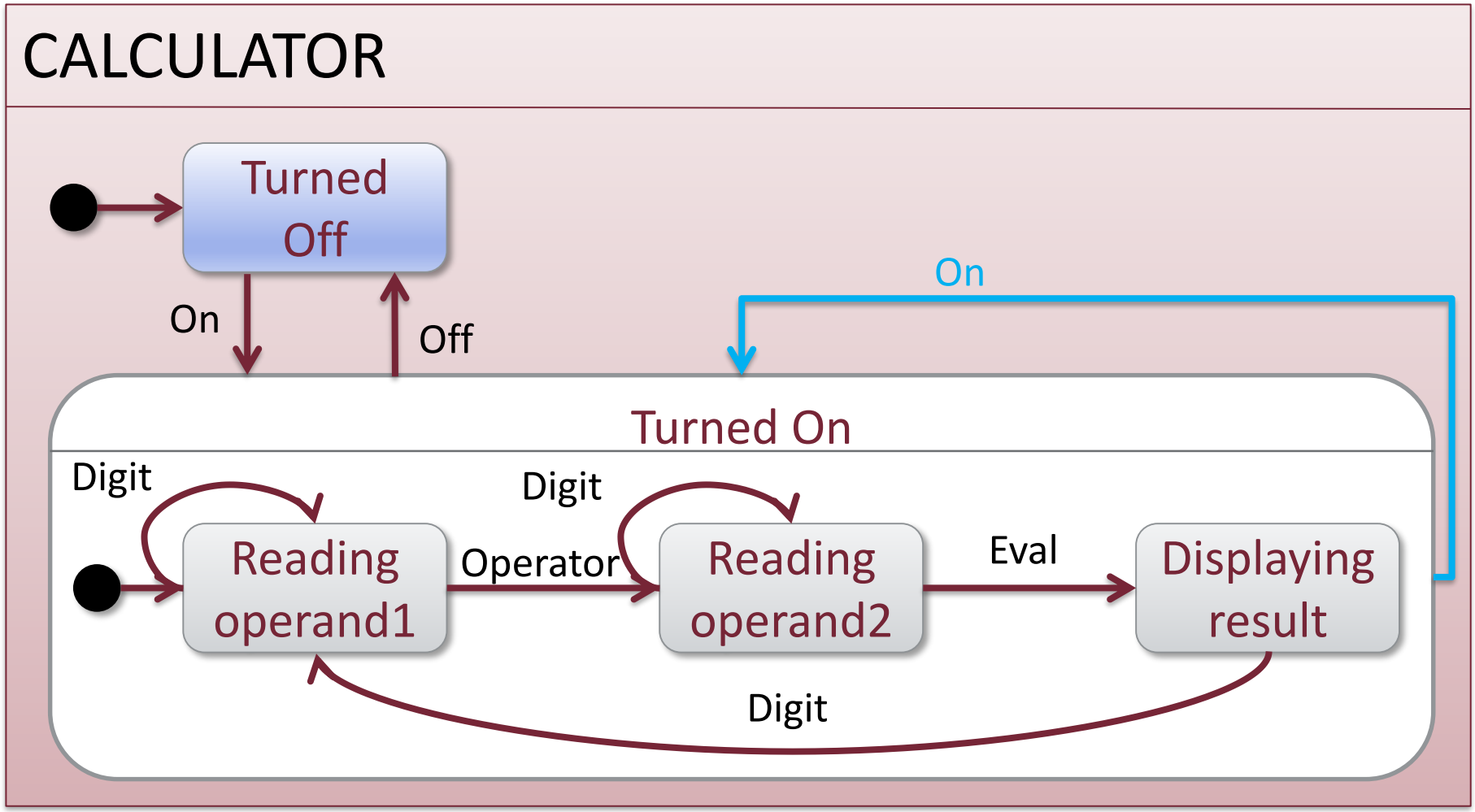
- Bemenet: {*on, off, digit, operand, eval*}
- Feltételezés: kétoperandusú műveleteink vannak

Állapothierarchia



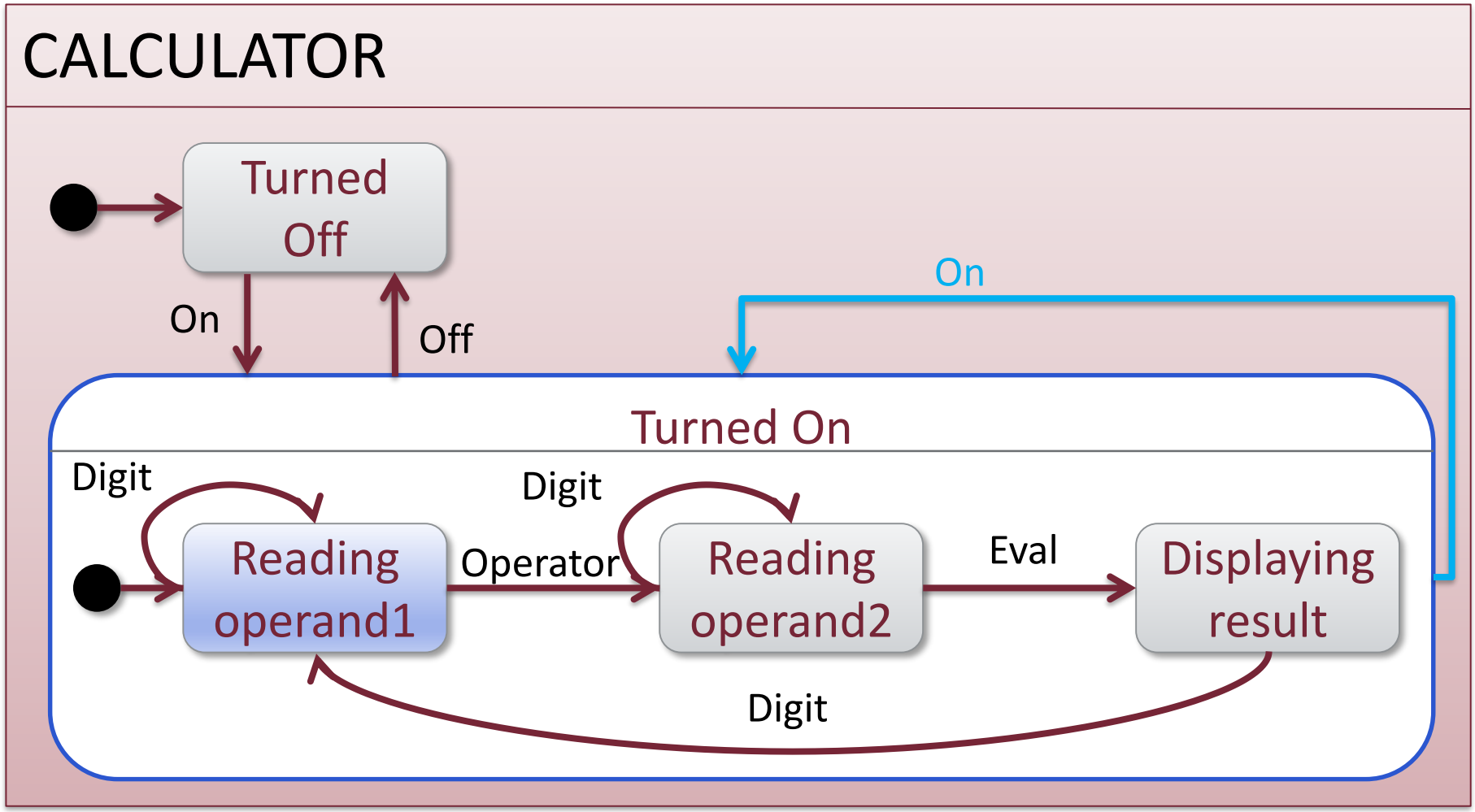
- Közös viselkedés kiemelése közös absztrakcióba

Állapothierarchia



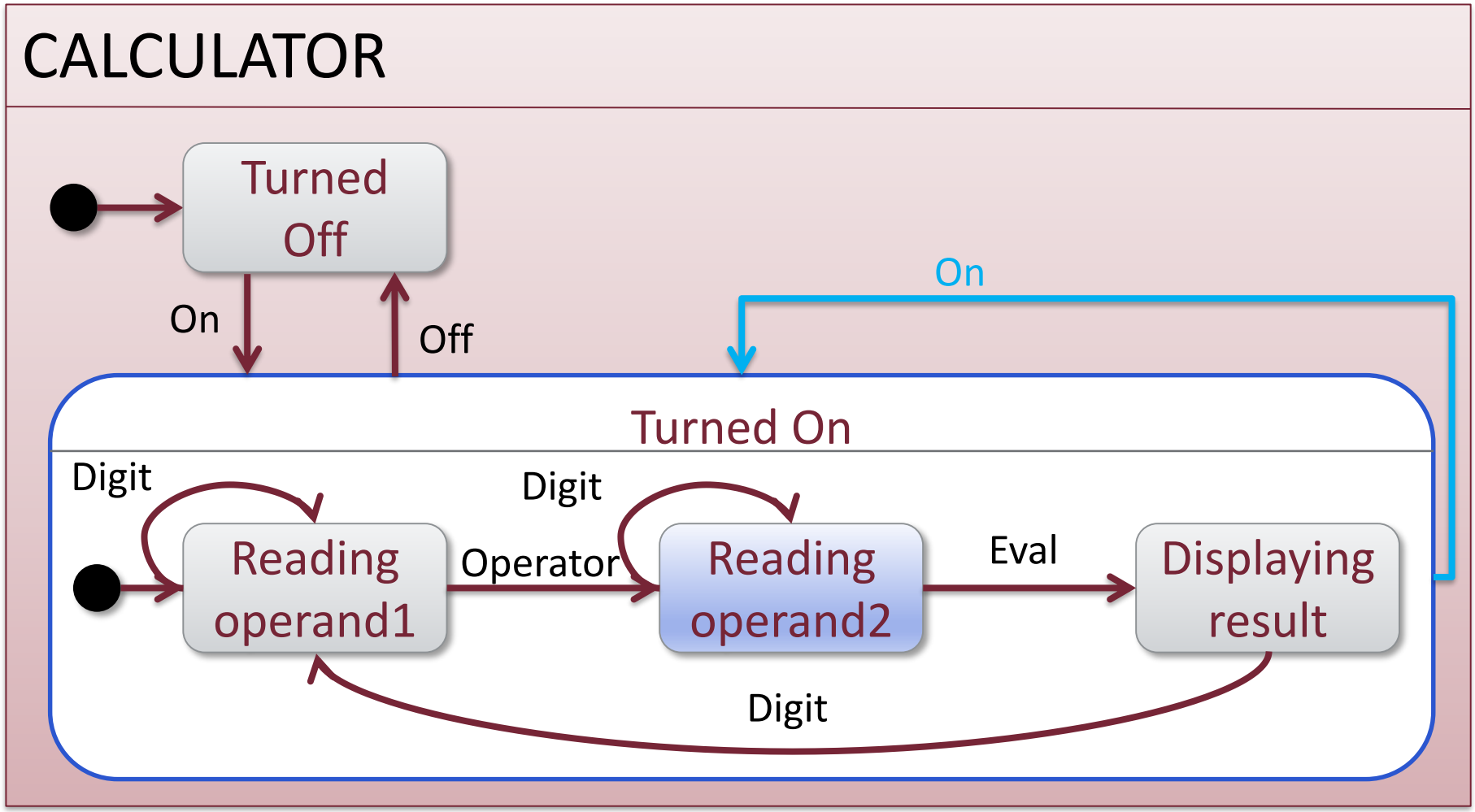
- Állapotkonfiguráció: {*Turned Off*}

Állapothierarchia



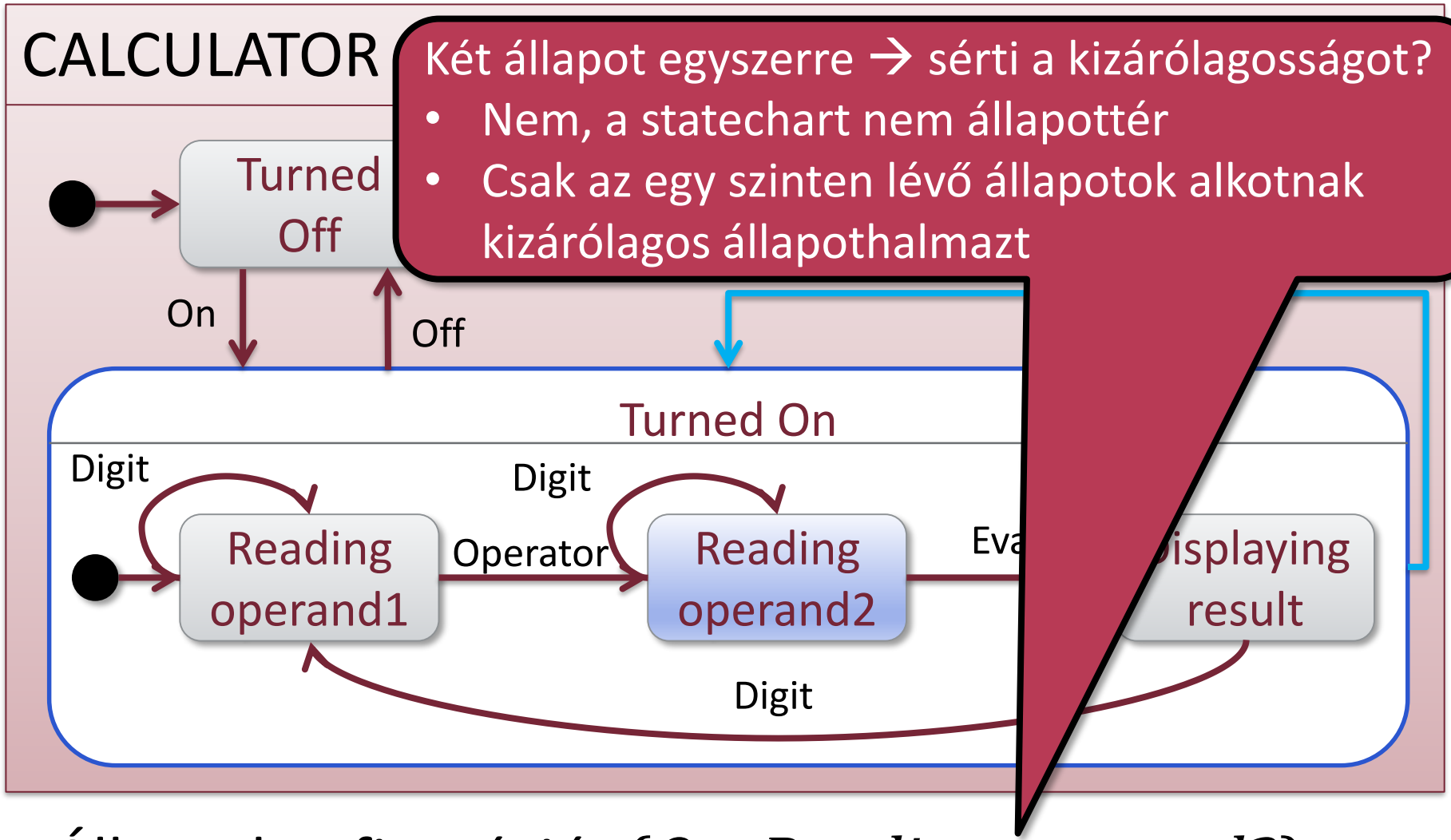
- Állapotkonfiguráció: $\{On, Reading\ operand1\}$

Állapothierarchia



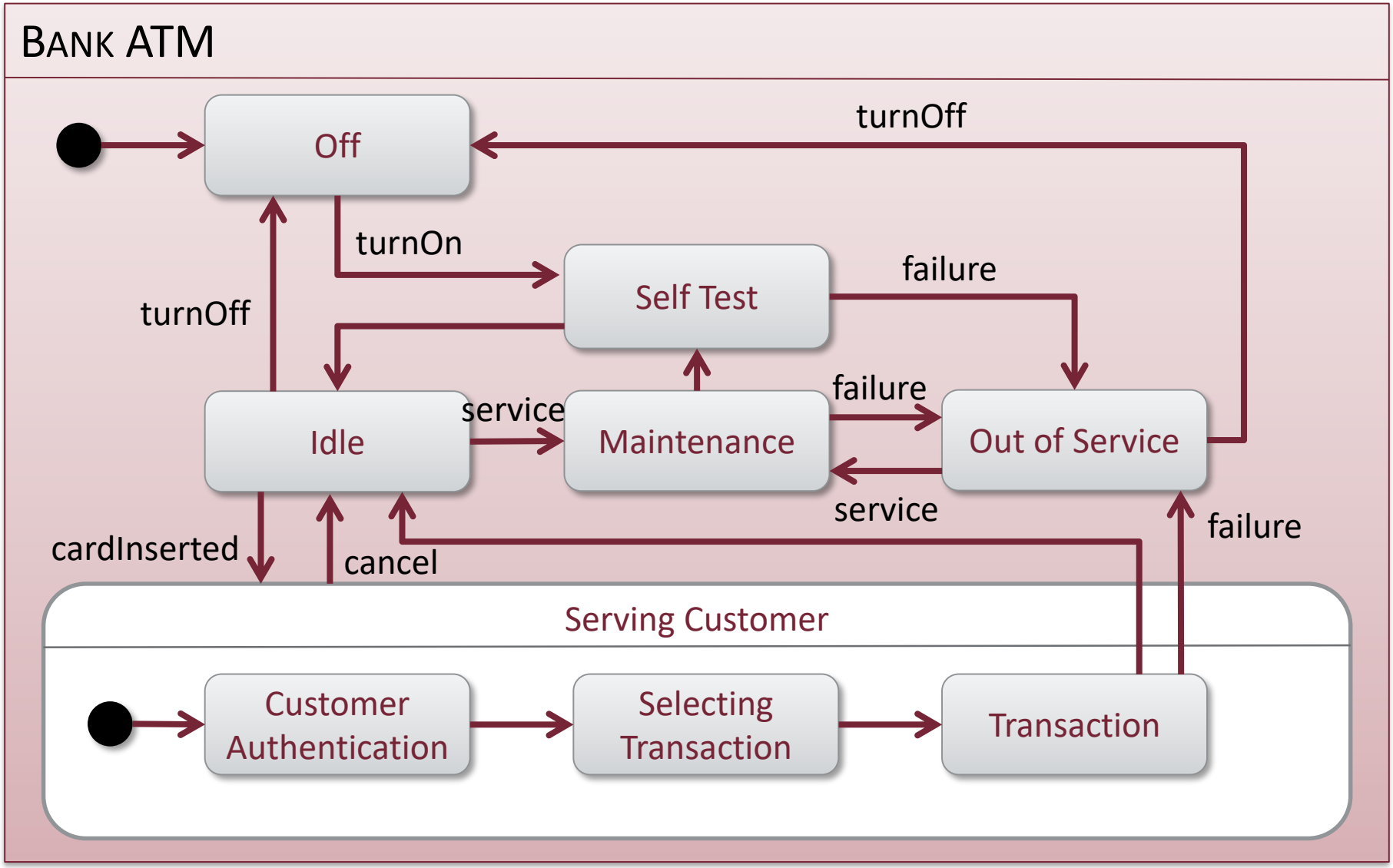
- Állapotkonfiguráció: $\{On, Reading\ operand2\}$

Állapothierarchia



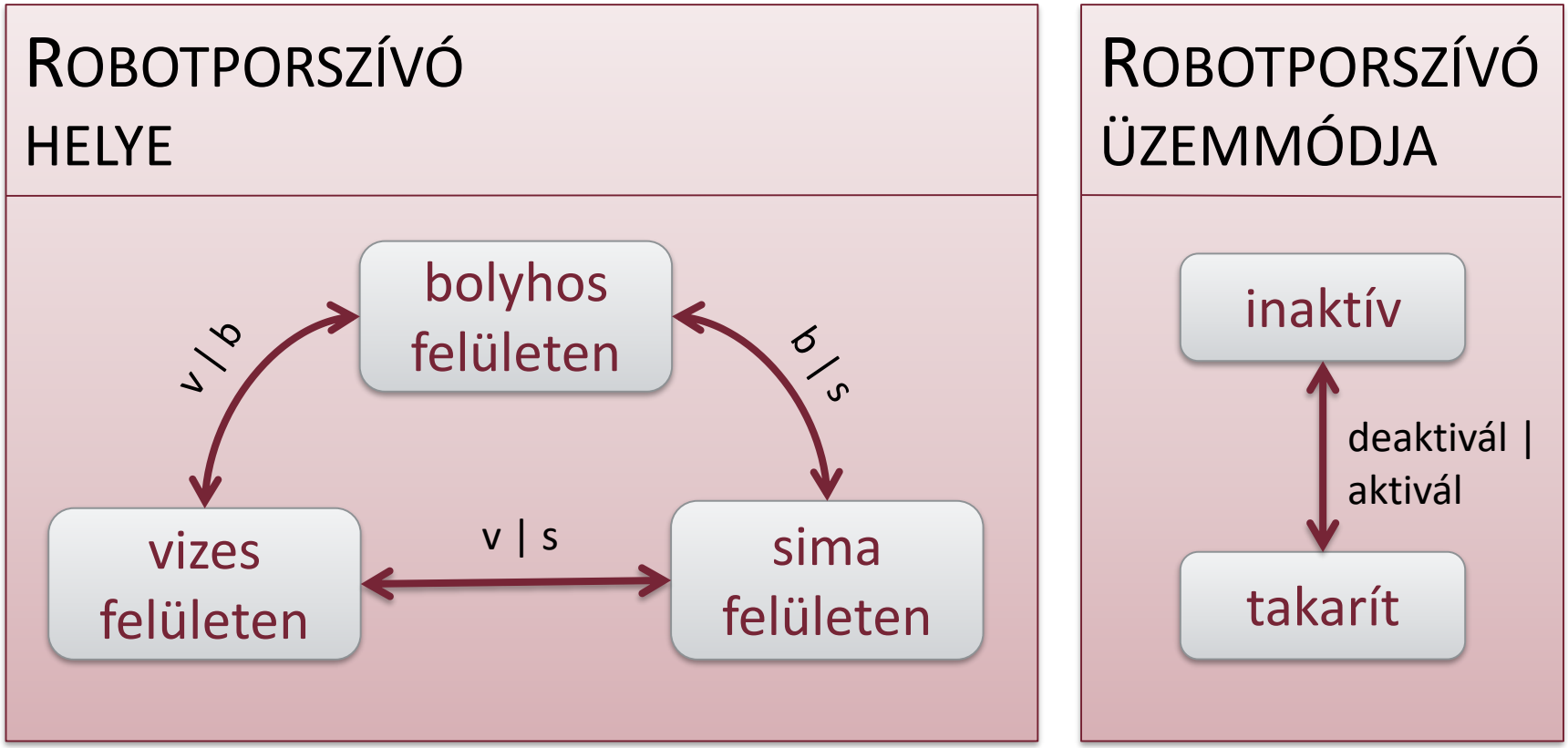
- Állapotkonfiguráció: $\{On, Reading\ operand2\}$

Példa: ATM


BANK ATM
Off
turnOff
turnOn
Self Test
failure
turnOff
Idle
service
Maintenance
failure
Out of Service
service
failure
cardInserted
cancel
Serving Customer
Customer Authentication
Selecting Transaction
Transaction

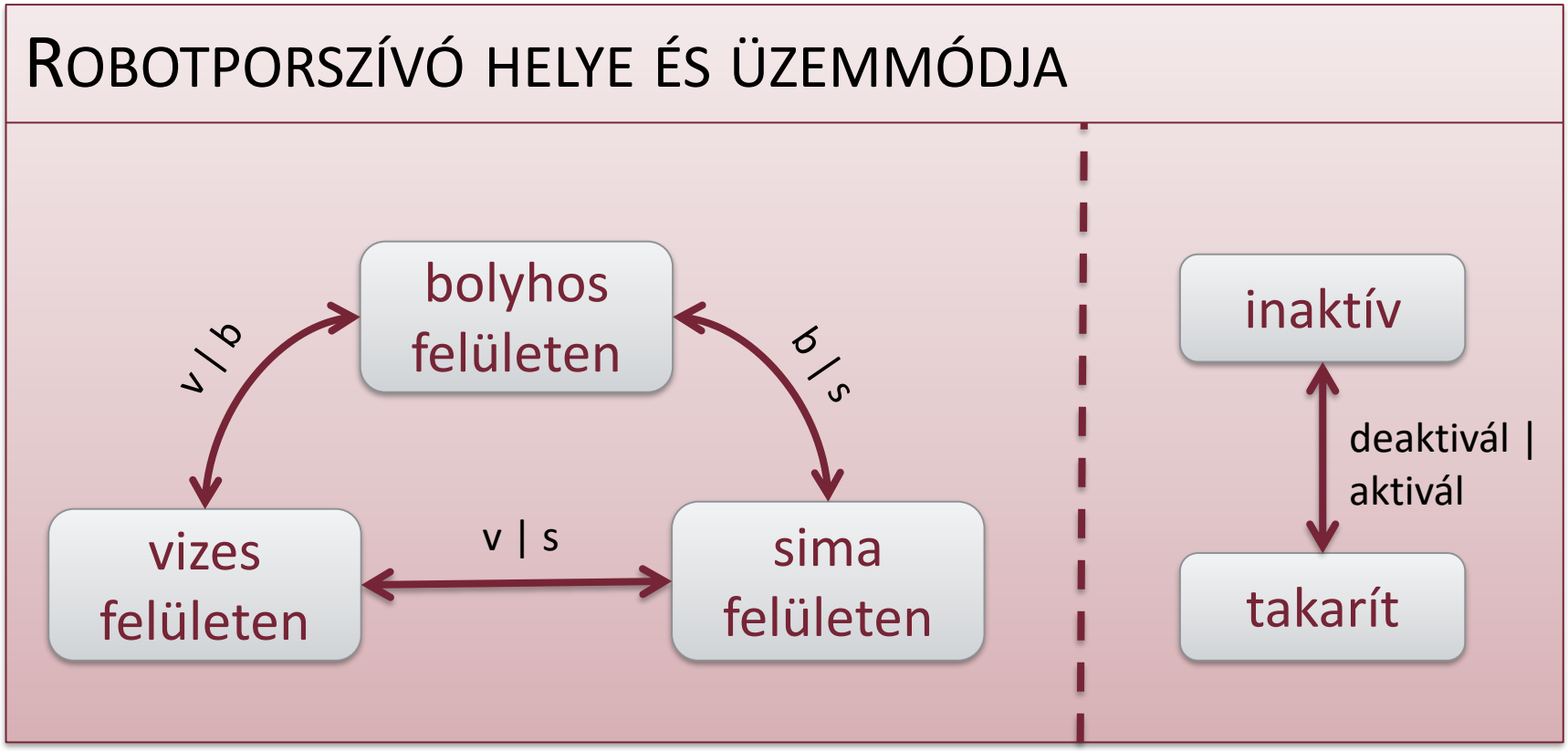
Példa: robotporszívó

Ortogonalitás

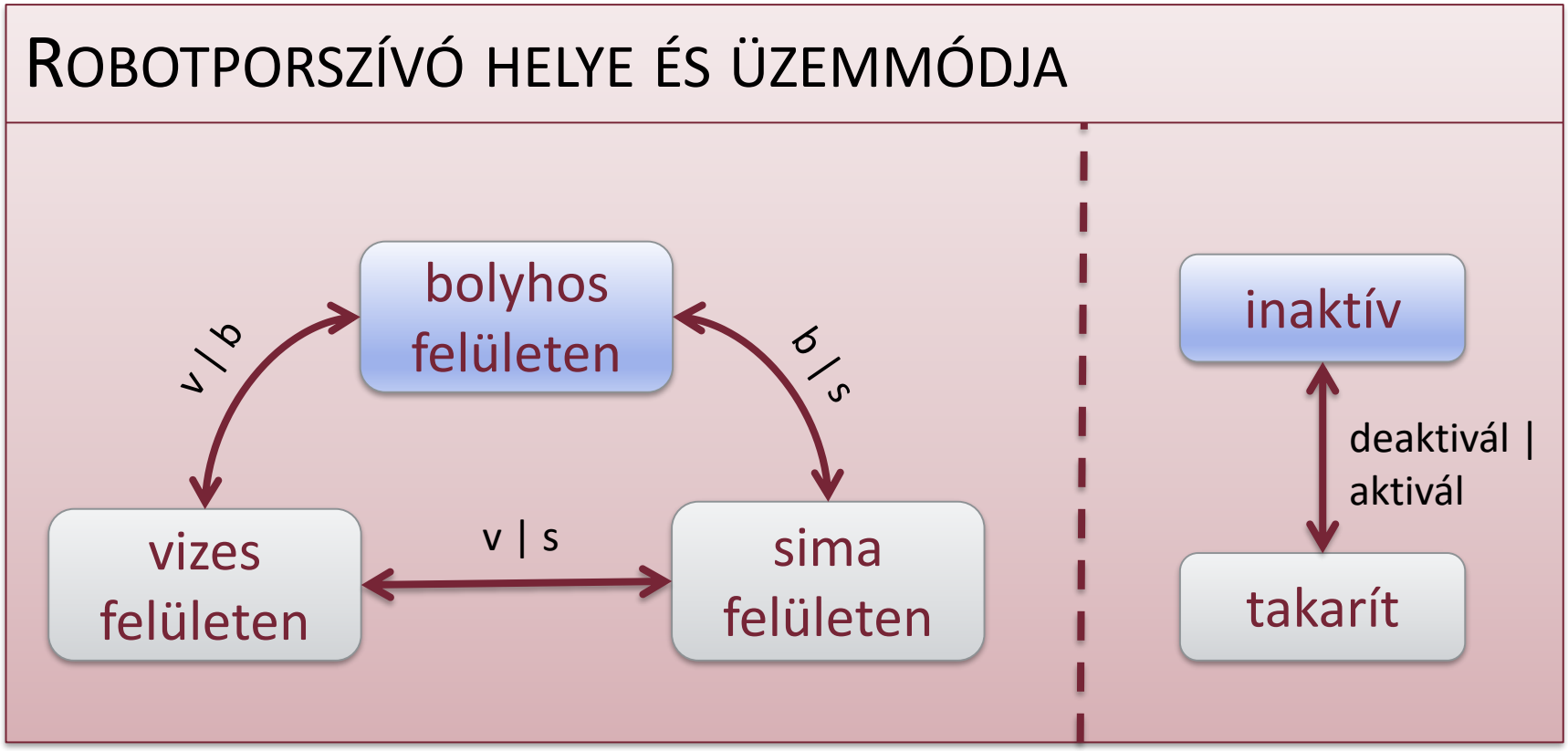


Megj: kétirányú nyíl nem szokásos, csak a tömör ábrázolás miatt használjuk…

Ortogonalitás

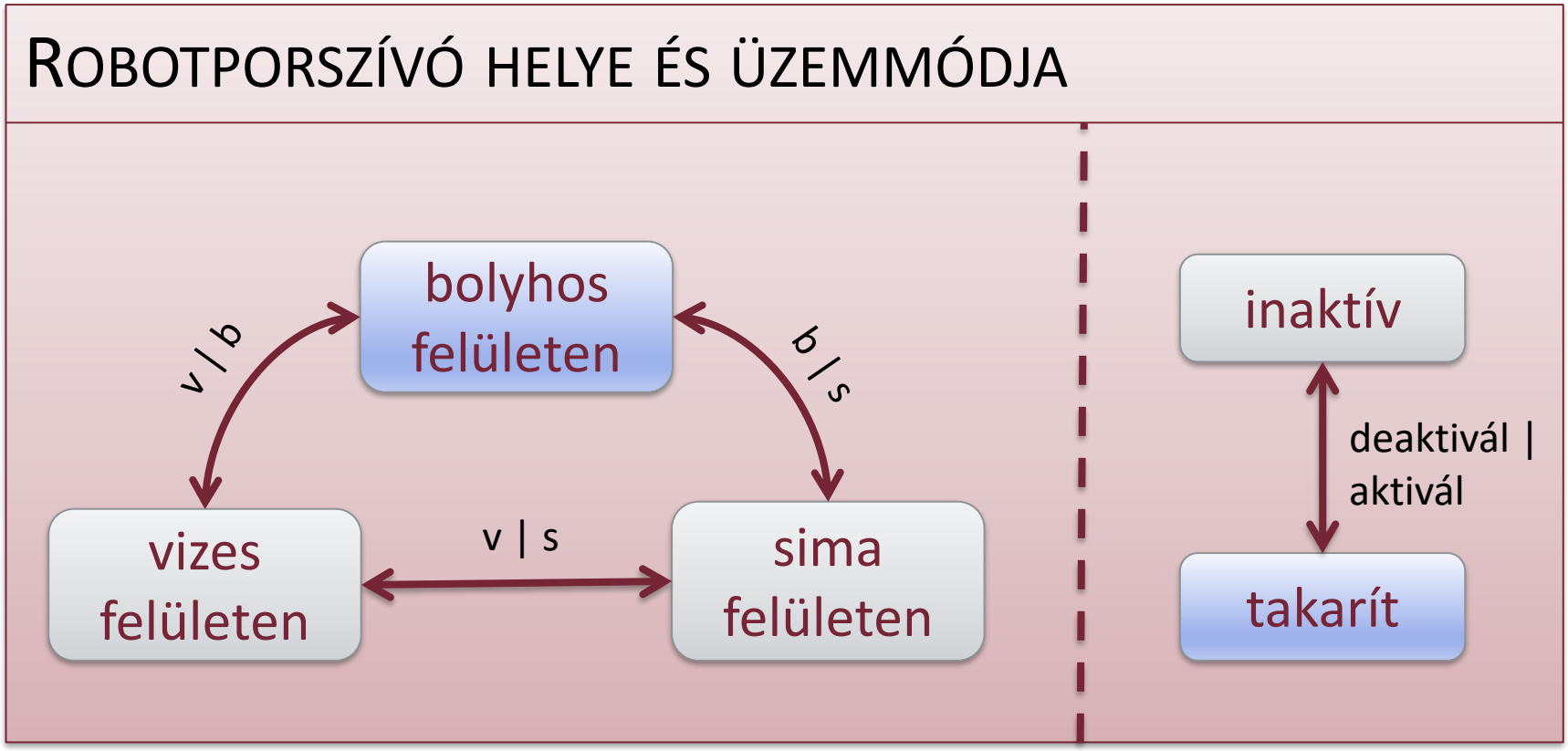


Ortogonalitás



- Állapotkonfiguráció: {*bolyhos felületen*, *inaktív*}

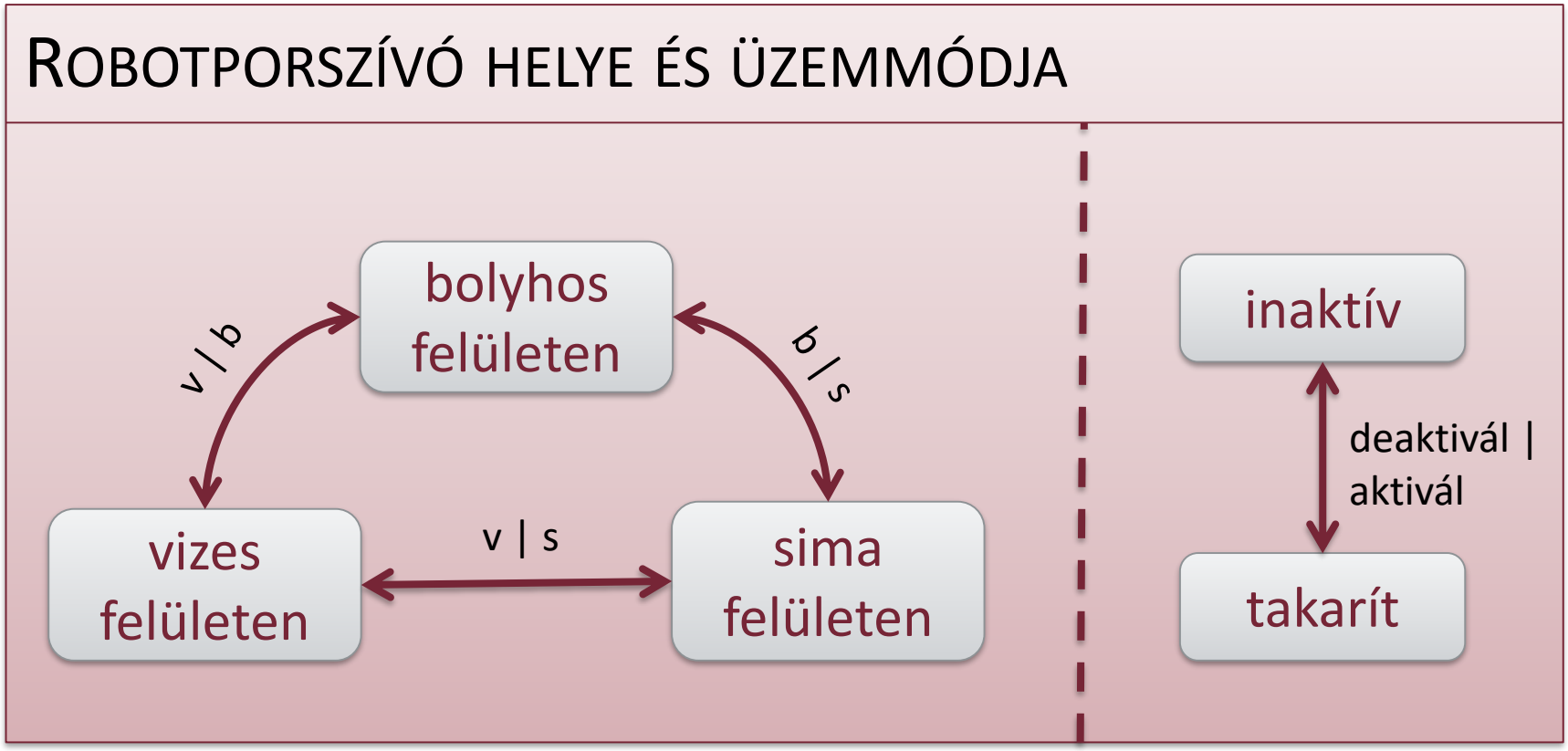
Ortogonalitás



- Állapotkonfiguráció: {*bolyhos felületen*, *takarít*}

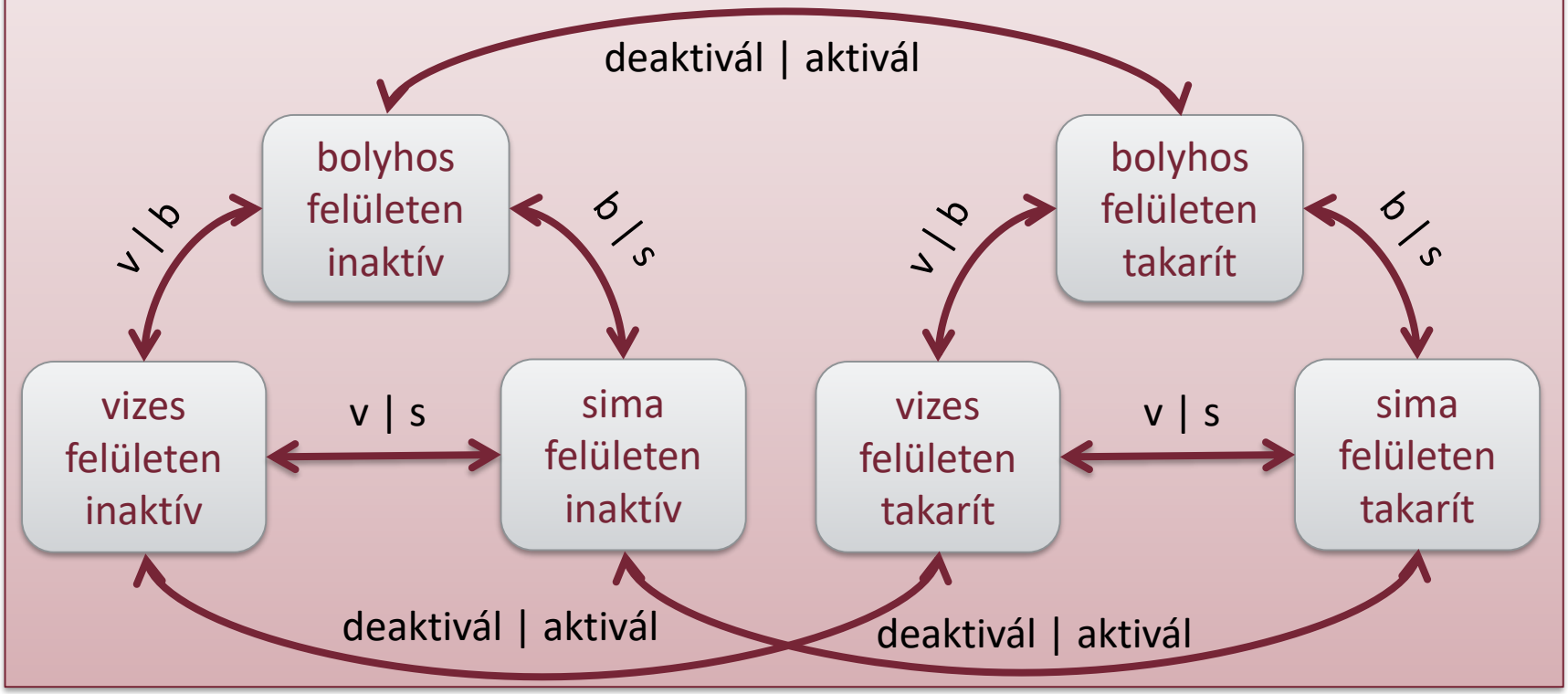
KOOPERÁCIÓ: SZORZATOK

Régiók aszinkron szorzata



Régiók aszinkron szorzata

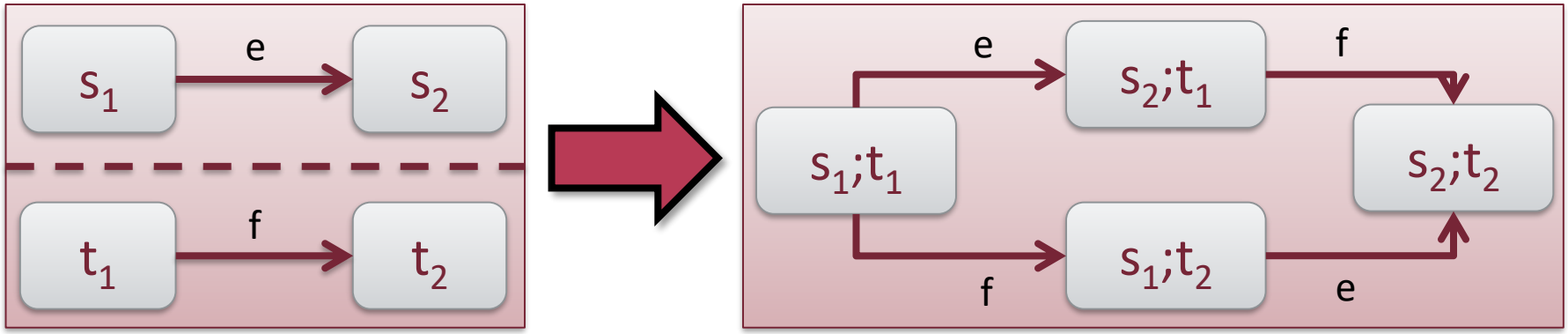
Robotporszívó helye és üzemmódja



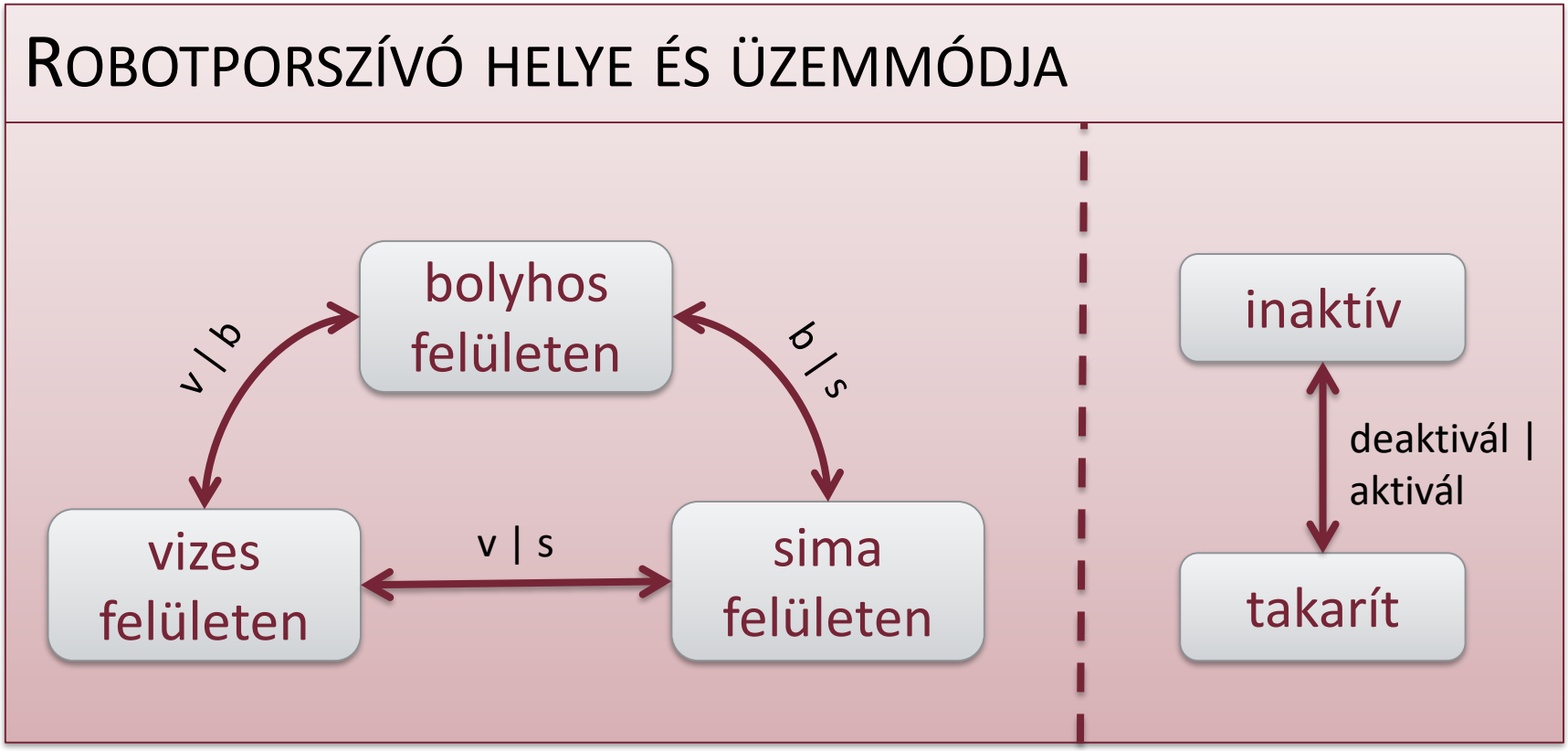
Definíció: Aszinkron szorzat

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.

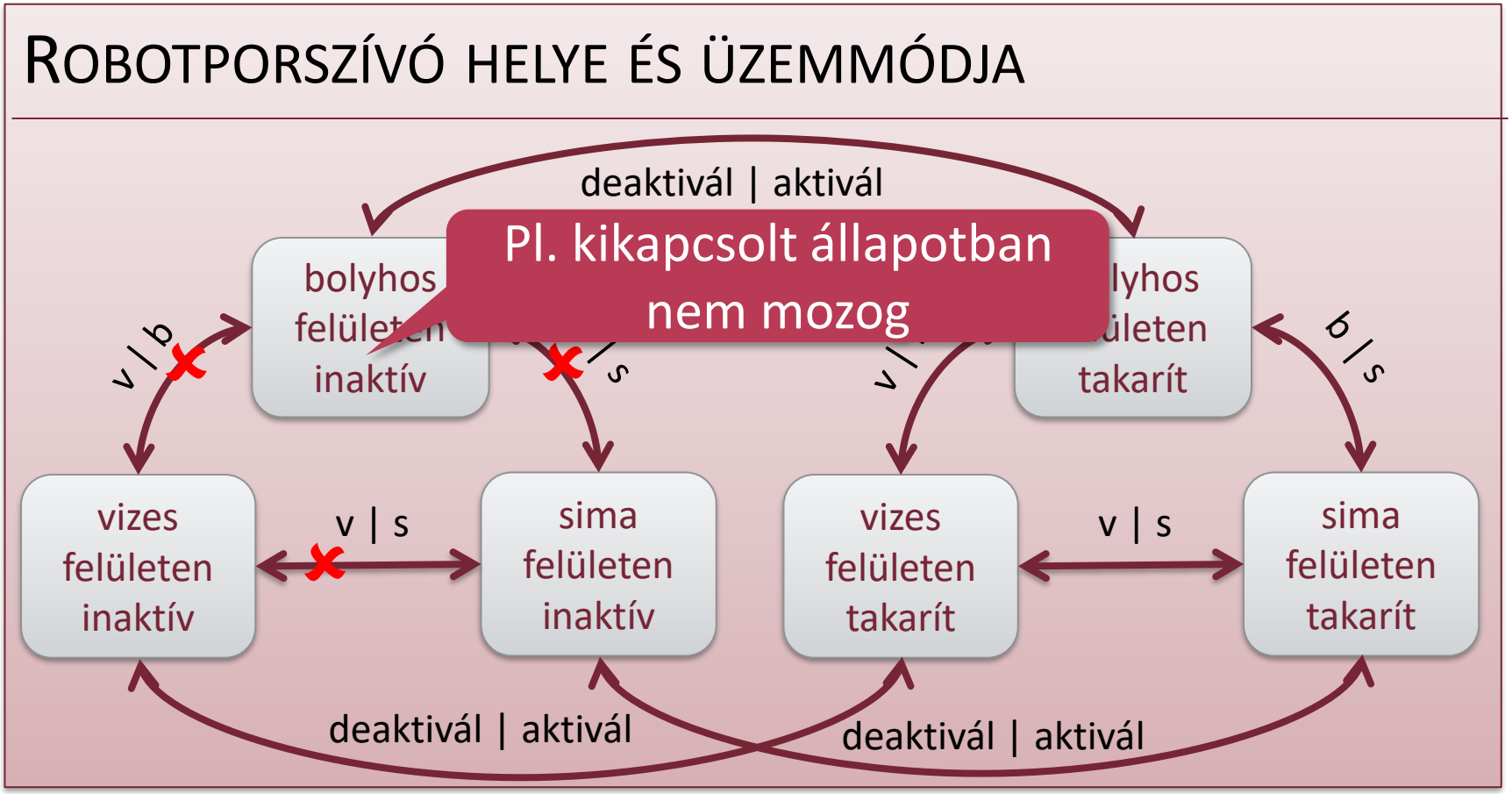


Kooperáció aszinkron szorzat esetén



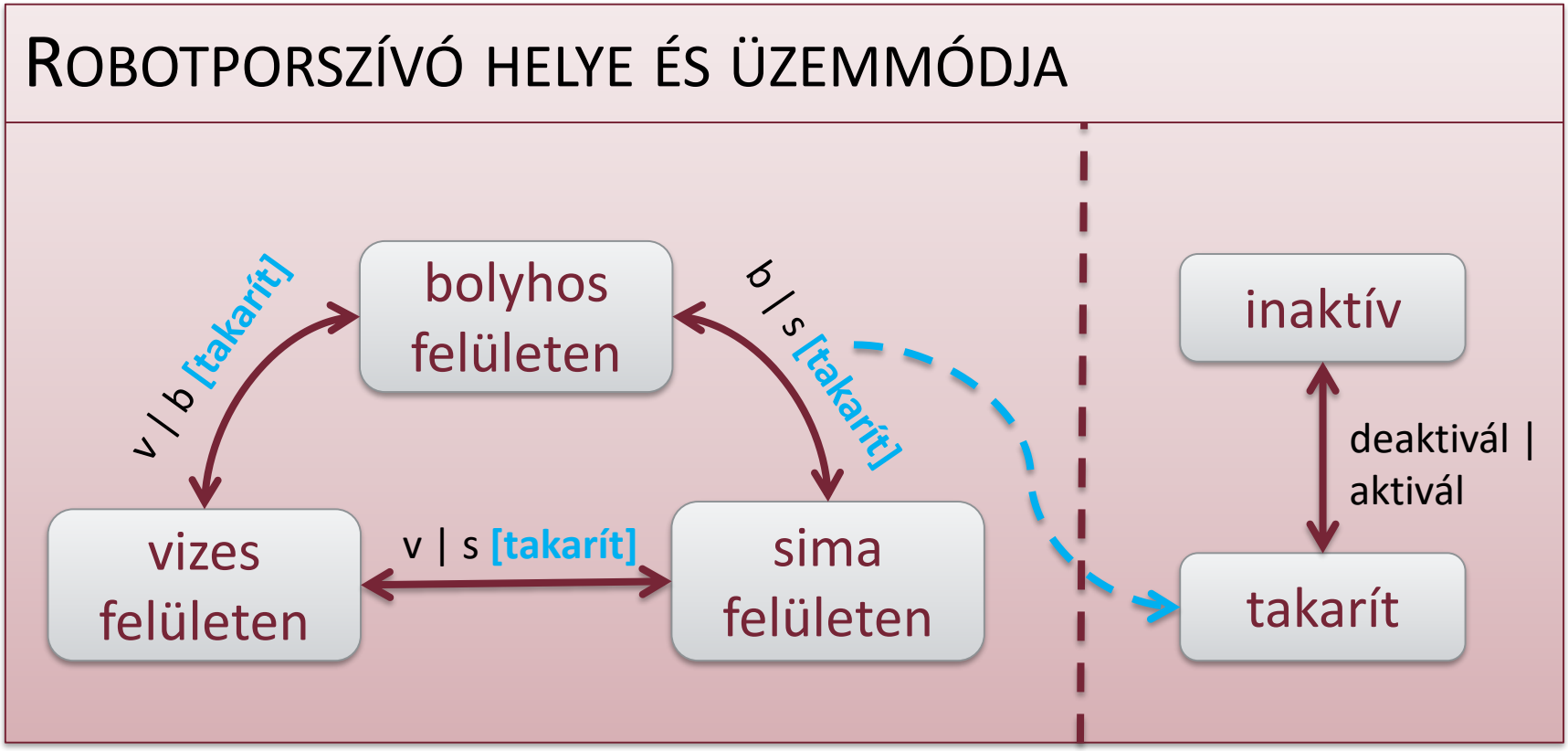
- Hogyan modellezzük a kooperációt?
  - Egészen pontosan hogyan **nem független** a két régió?

Régiók aszinkron szorzata



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

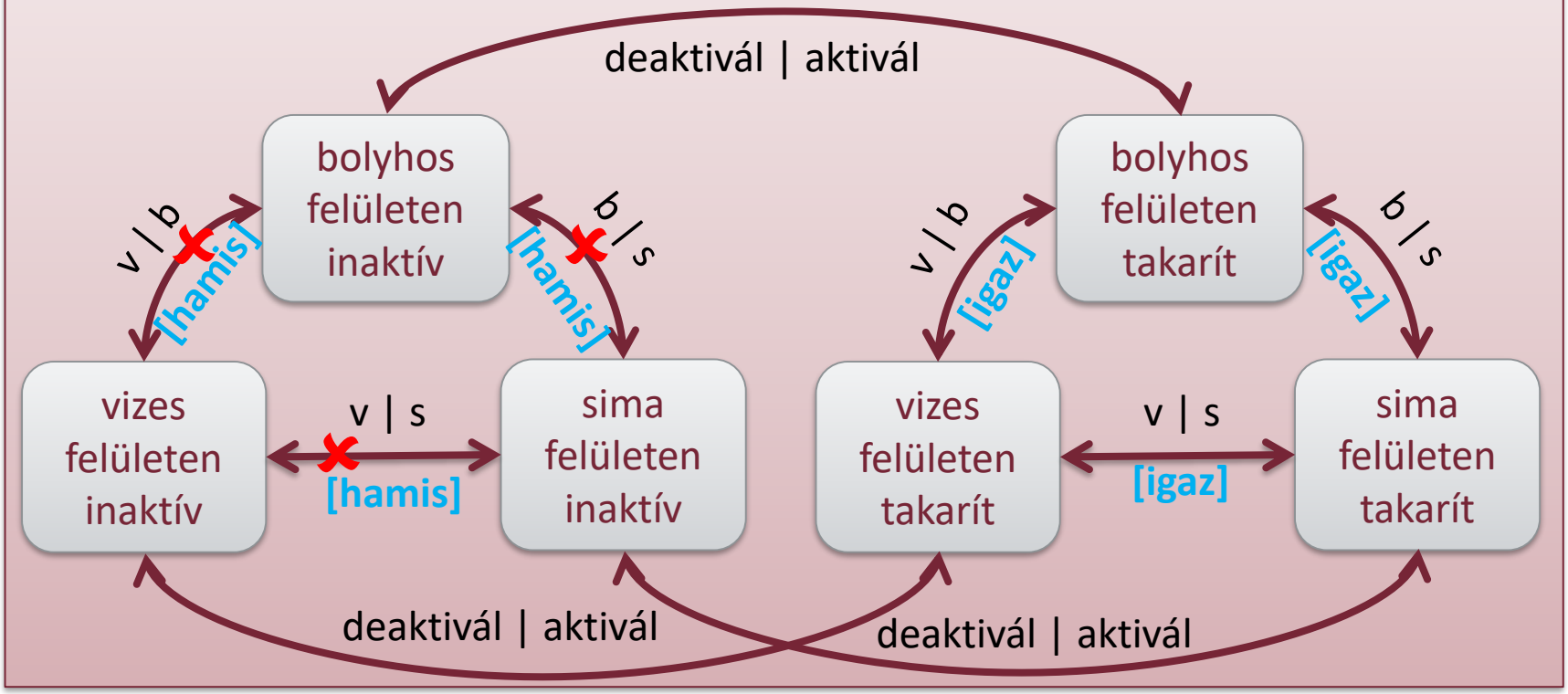
Kooperáció aszinkron szorzat esetén



- Kooperáció **őrfeltétel**lel
  - Egyik régióban átmenet feltétele másik régió állapota

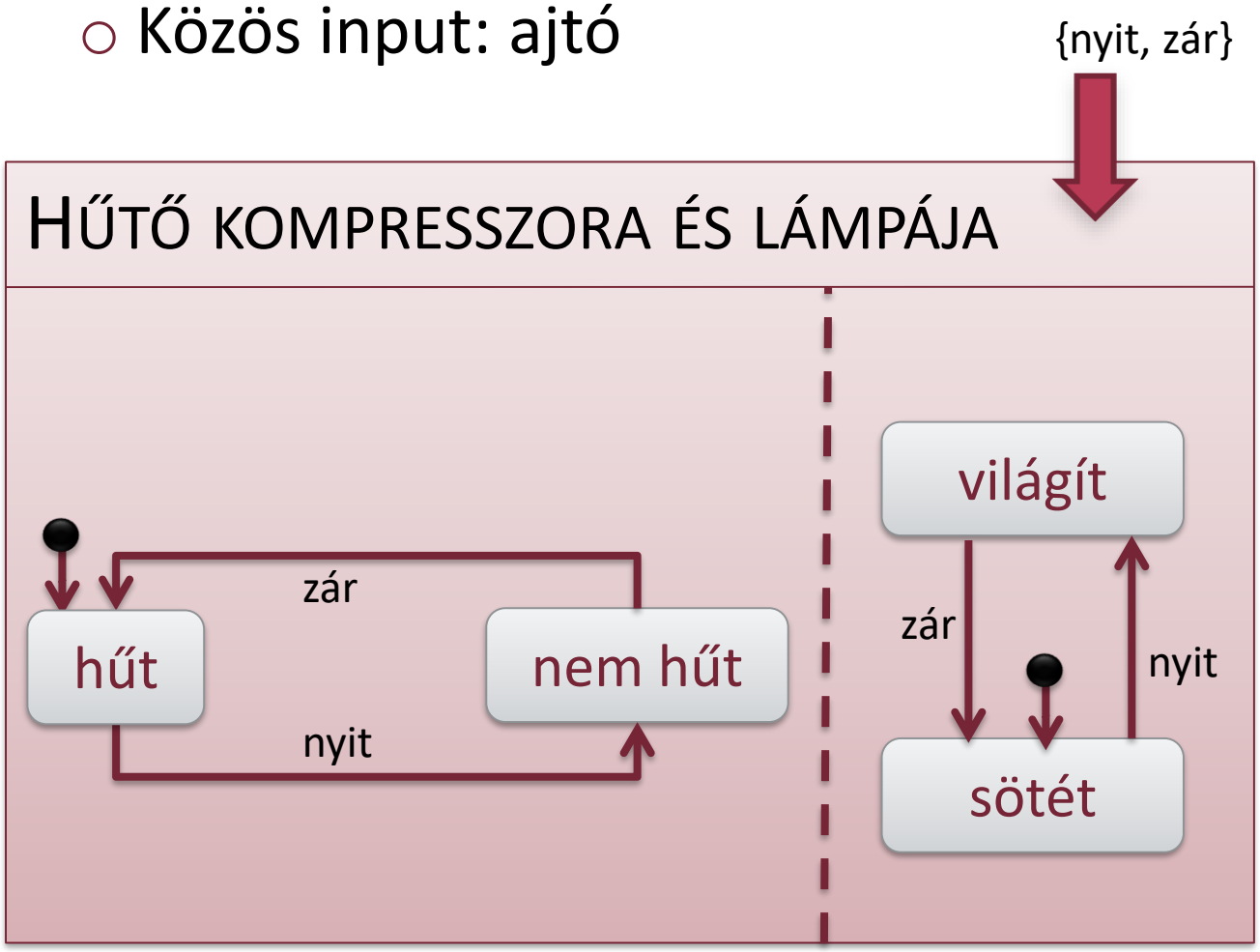
Kooperáció aszinkron szorzat esetén

Robotporszívó helye és üzemmódja



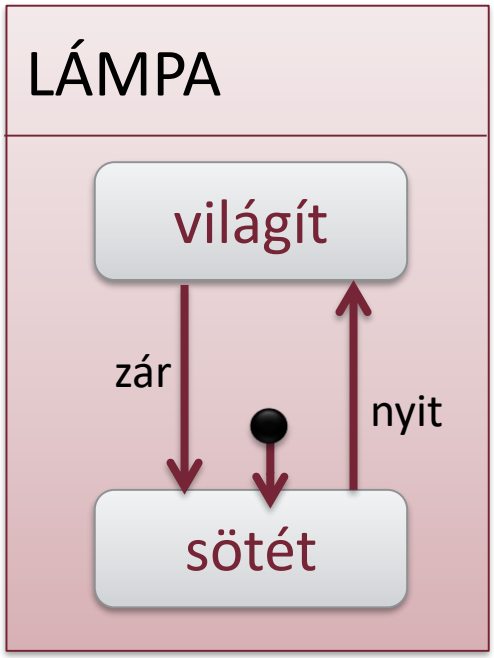
Szinkron szorzat példa

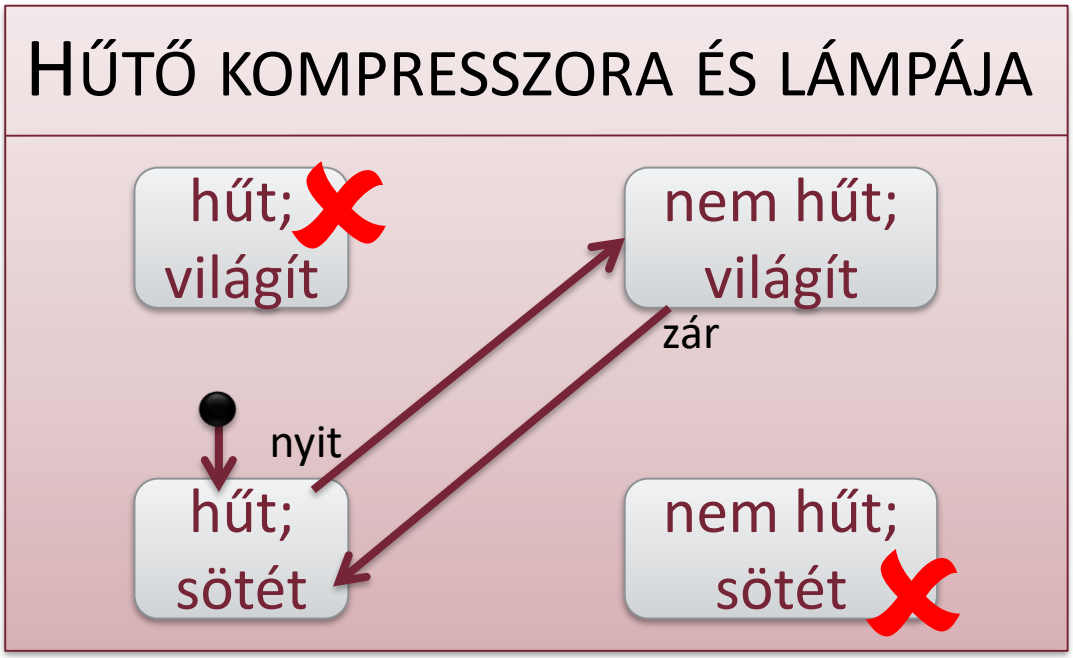
- Hűtőszekrény kompresszor vs. lámpa
  - Közös input: ajtó



Szinkron szorzat példa

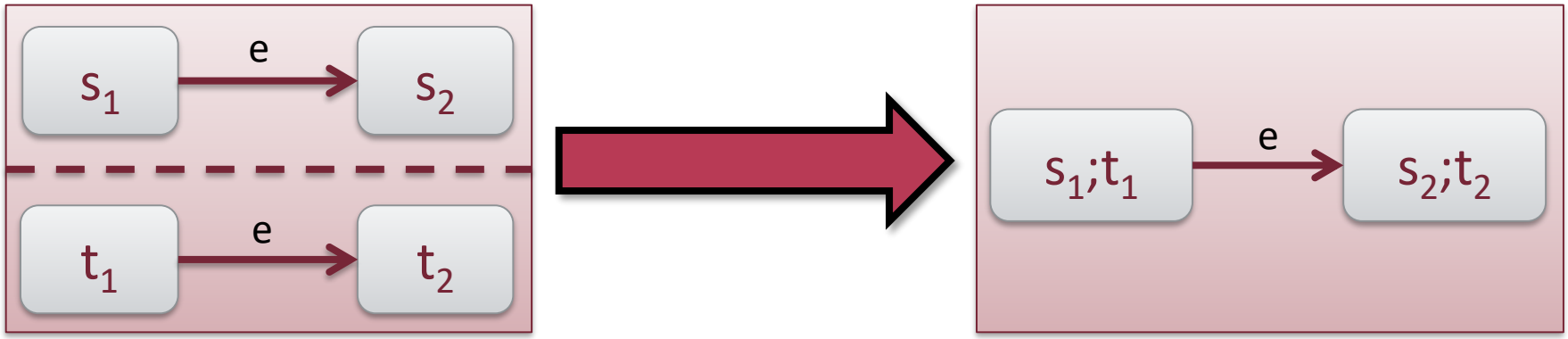
- Közös inputolvasás → szinkron lépés
  - (A kezdőállapotból elérhetetlenné válhatnak állapotok)





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



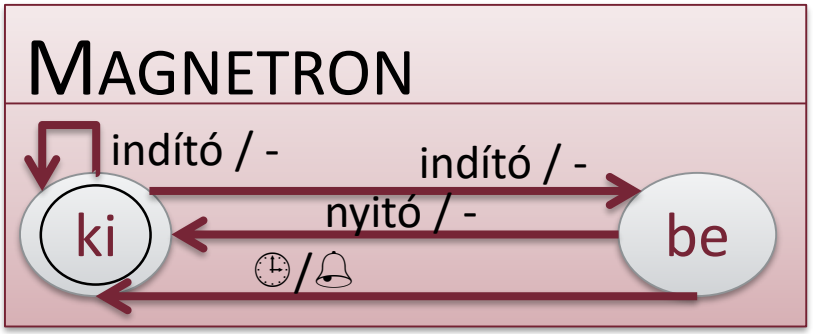
Vegyes szorzat

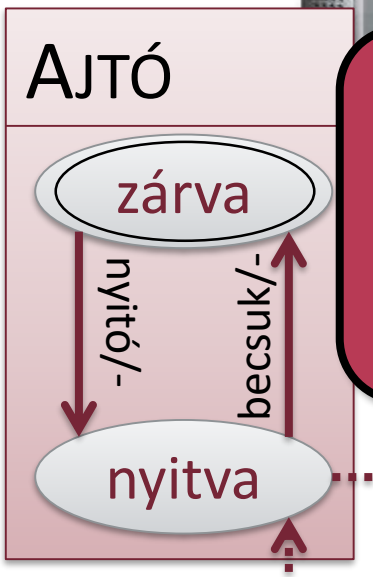
- Állapotgépek szorzata
  - Modell felépítése állapotrégiókból (komponensekből)
  - Állapothalmaz: a régiók állapottereinek **direkt szorzata**
  - Kezdőállapot: a régiók kezdőállapotaiból álló n-es
- Átmenetek
  - **Vegyes szorzat**: helyenként szinkronizáció történik
    - Alapvetően aszinkron kompozíció…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

Szinkronizáció egyszerű esete: van közös input (is)

Vegyes szorzat példa

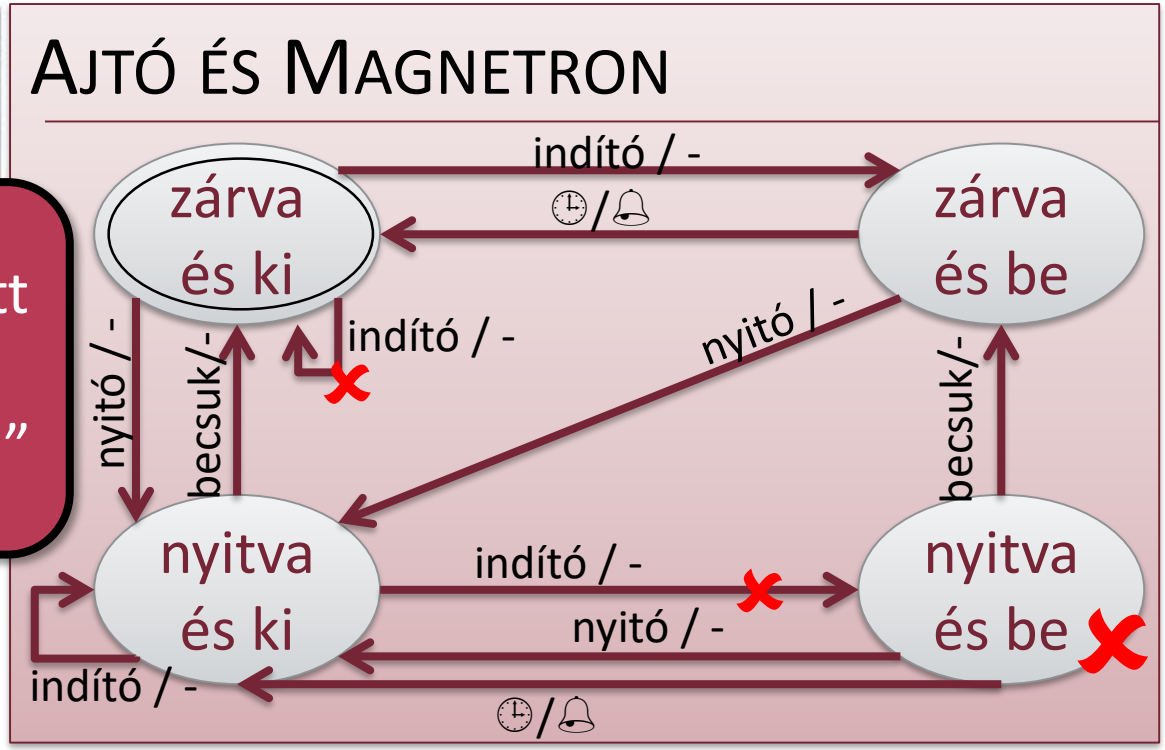
- További finomítást igényel
  - Átmenetek kieshetnek
  - (A kezdőállapotból így elérhetetlenné válhatnak állapotok)





Figyelmen kívül hagyott inputok: „odaképzelt” hurokél

indító / -



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)

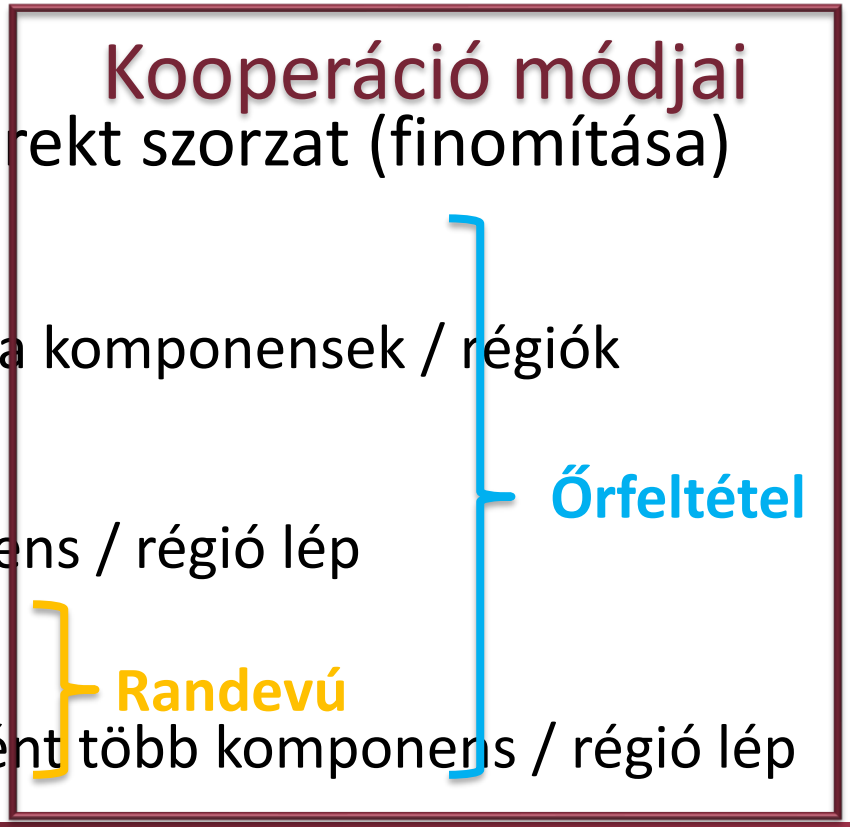
Fejlett kooperáció: randevú (belső szinkronizáló esemény)

Szorzatok áttekintése

- Állapotterek szorzata
  - **Direkt szorzat**: $S_1 \times S_2 \times \ldots \times S_n$
    - 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**: $S_1 \times S_2 \times \ldots \times S_n$
    - 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

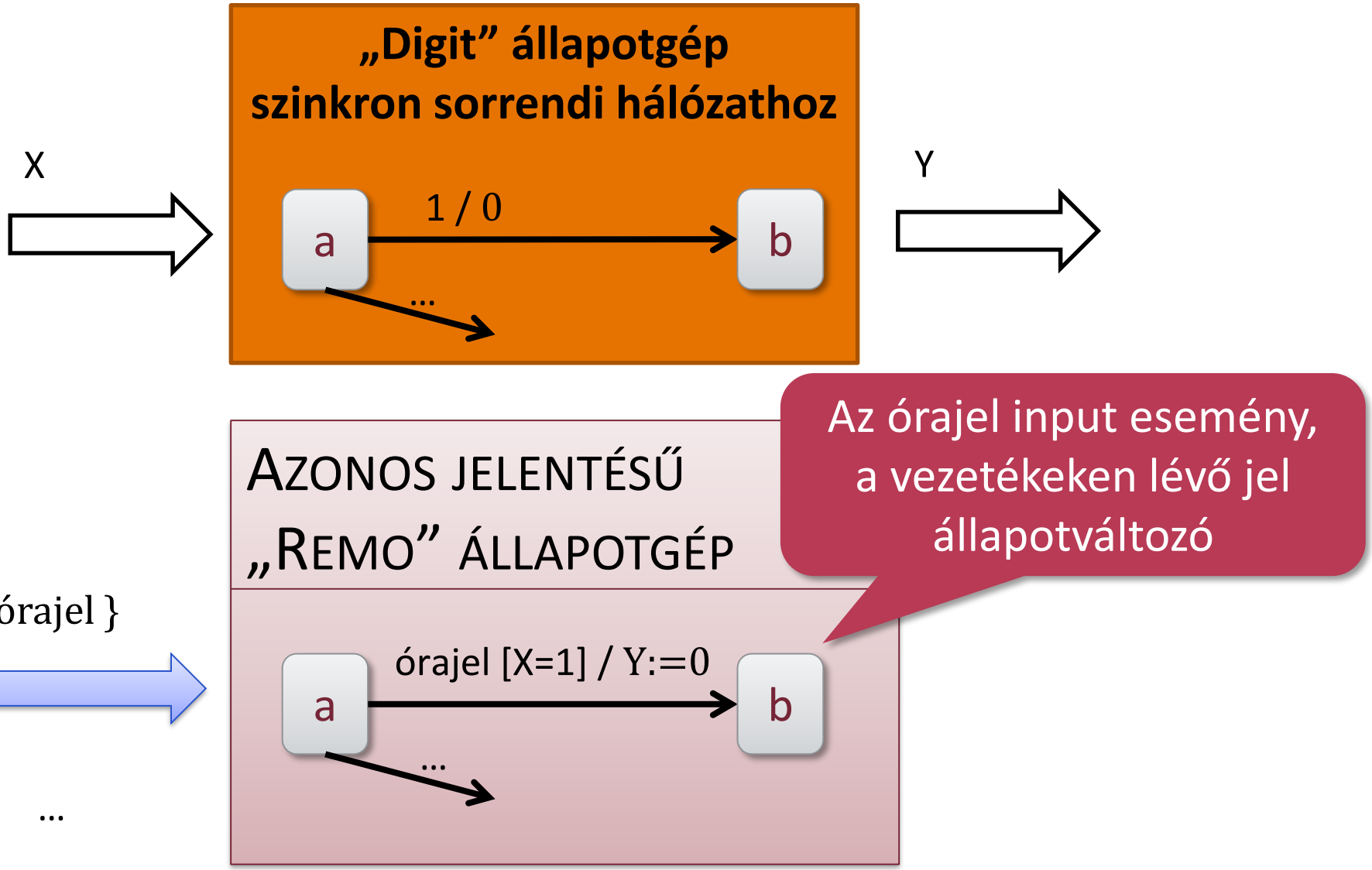


KITEKINTÉS, SZOFTVERTÁMOGATÁS

Az állapotátmenet értelmezése

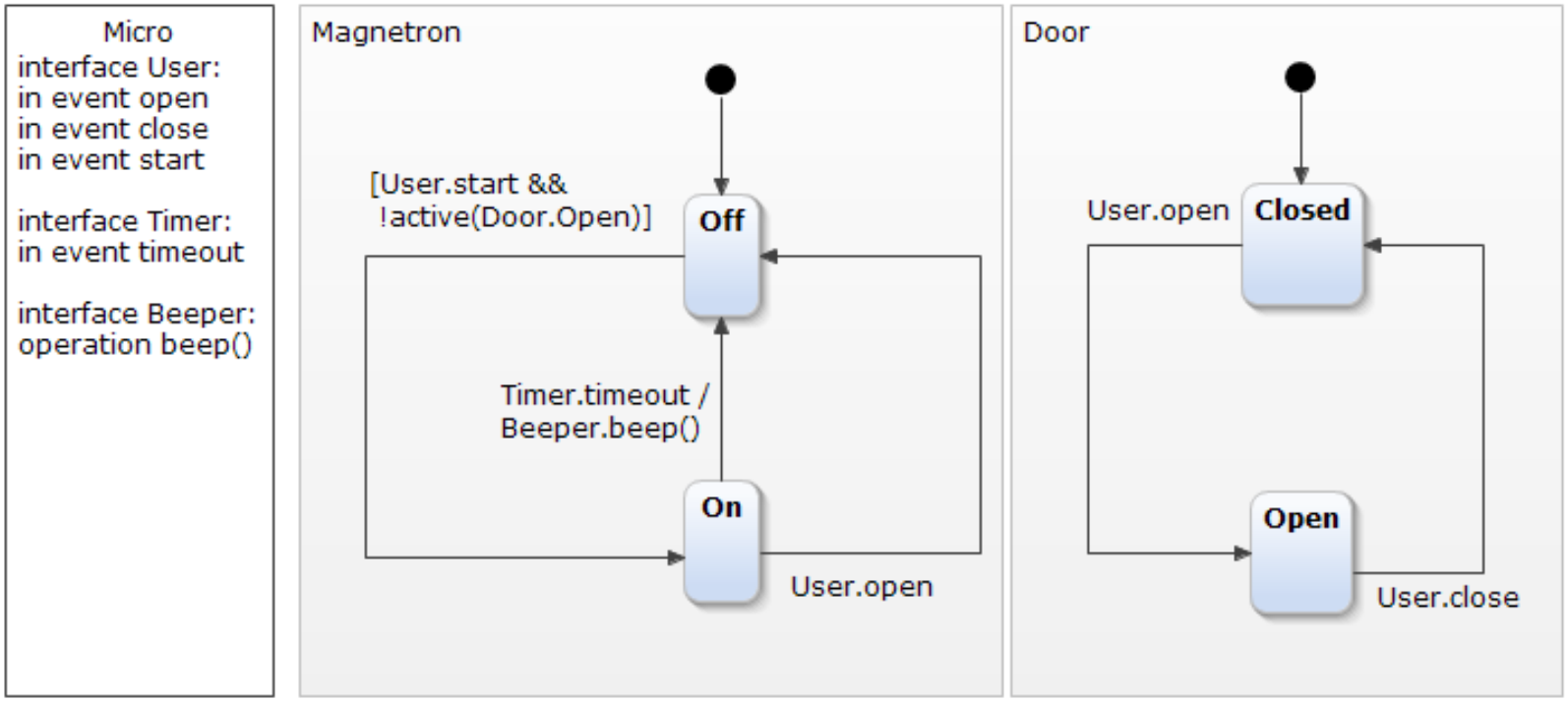
- Digitális áramkörtervezésnél
  - Szinkron sorrendi hálózat (ld. *Digitális Technika*)
  - Egyféle input esemény (órajel)
  - Állapotátmeneten feltüntetett input feltételek
    - Mindig órajelre lép →fel se tüntetjük
    - Input feltétel: a bemenő jelvezetékek *állapotára* vonatkozik
- Ezzel szemben **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



Yakindu Statechart Tools

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



Yakindu végrehajtási üzemmódok

- „Digites” állapotgép:
  - Cycle-based működési mód (alapértelmezett)
  - `CycleBased(200ms)` annotáció
  - 200ms-onként „magától” végrehajtódik és feldolgozza az összes bejövő eseményt
  - Triggerelhetünk több esemény kombinációjára is
- „Remo” állapotgép:
  - Event-based működési mód (alapértelmezett)
  - `EventBased` annotáció
  - Minden (időzítő) eseményre azonnal végrehajtódik
  - Egy eseményre triggerelhetünk

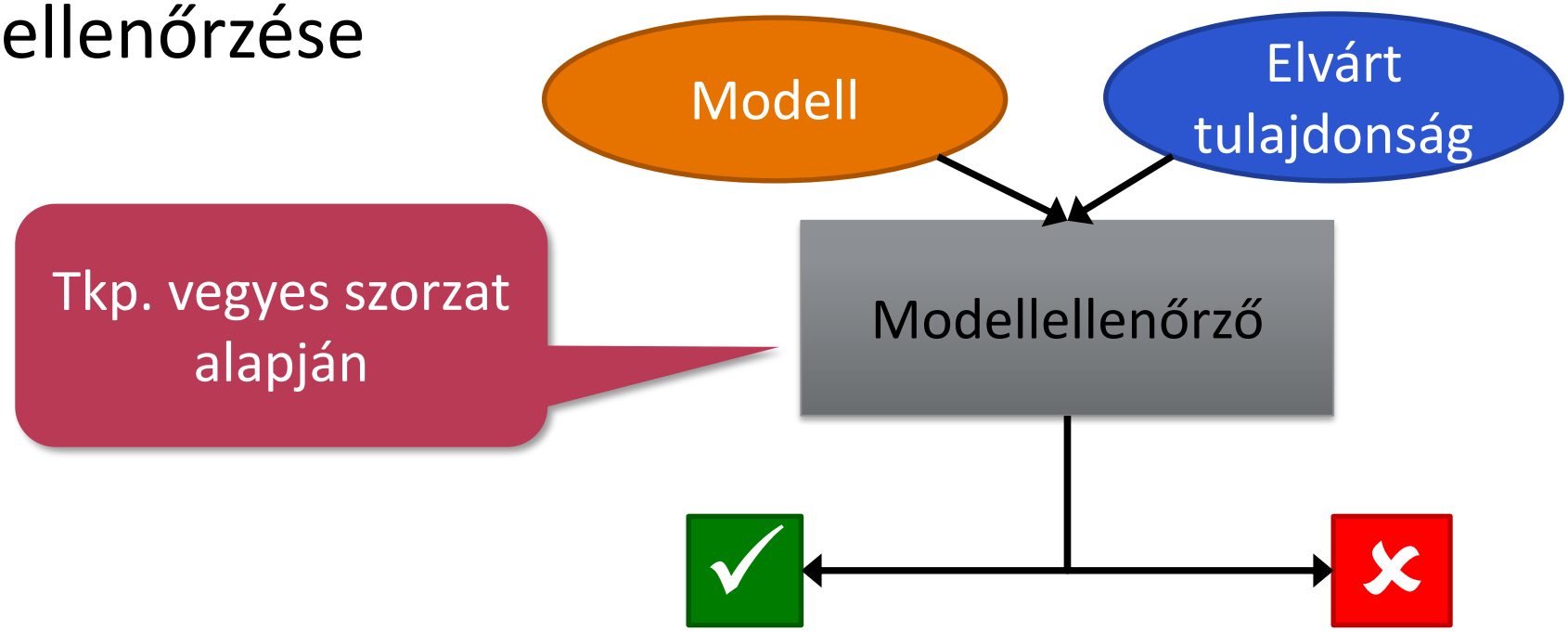
Yakindu Statechart Tools

- 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;
        }
    }
}
```

Modellellenőrzés alapok

- Modellezés (egyik célja): tulajdonságok ellenőrzése



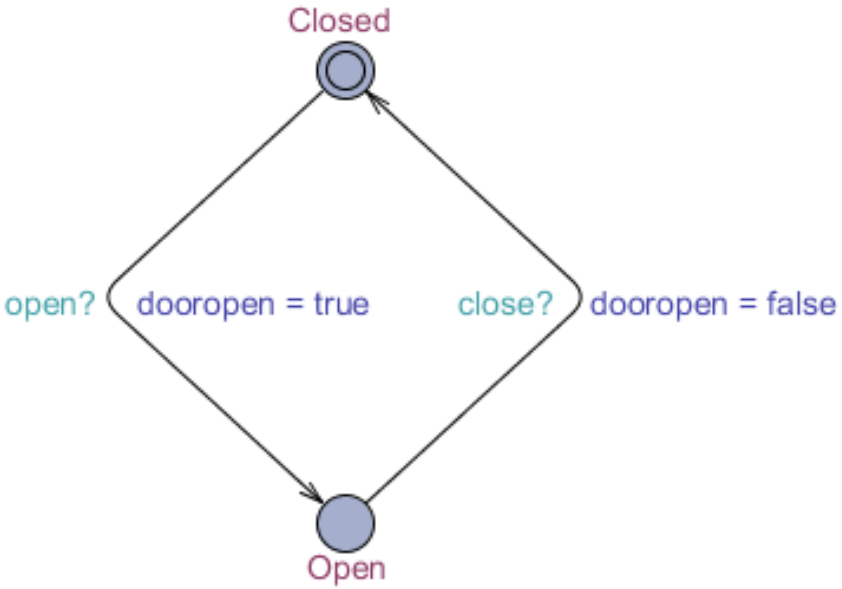
- Követelmények lehetnek:
  - Hibás állapot nem elérhető
  - „0” után mindig „1” jel jön a kimeneten

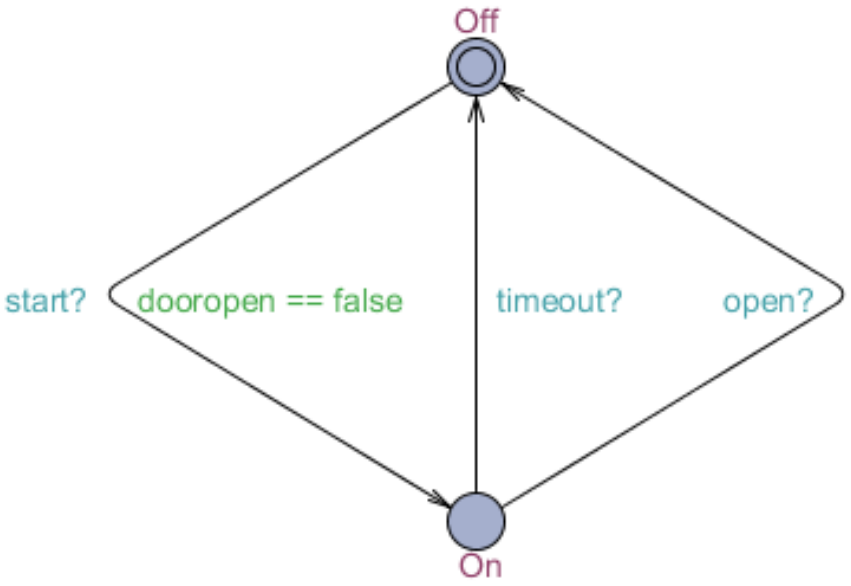
Temporális logika

- Időbeliség megfogalmazására
- Logikai operátorok: és, vagy, nem, következik
- Temporális operátorok:
  - X p: „neXt p”, a következő állapotban igaz lesz p
    - Ha a lámpa zöld, a következő állapotban sárga lesz: $zöld \rightarrow X\ sárga$
  - F p: „Future p”, egy elérhető állapotban igaz lesz p
    - Ha a lámpa piros, előbb vagy utóbb zöld lesz: $piros \rightarrow F\ zöld$
  - G p: „Globally p”, minden elérhető állapotban igaz lesz p
    - Minden állapotban igaz, hogy a lámpa előbb vagy utóbb zöld lesz: $G(F\ zöld)$
  - p U q: „p Until q”, egy elérhető állapotban igaz lesz q, és addig minden állapotban igaz p
    - Ha a lámpa elromlik, addig kikapcsolva marad, amíg meg nem javítják: $G(rossz \rightarrow kikapcsolt\ U\ jó)$

UPPAAL (ld. modellek ellenőrzése)

- 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 állapot**a az állapottér azon egyetlen eleme, amelyik abban az időpontban jellemző a rendszerre.

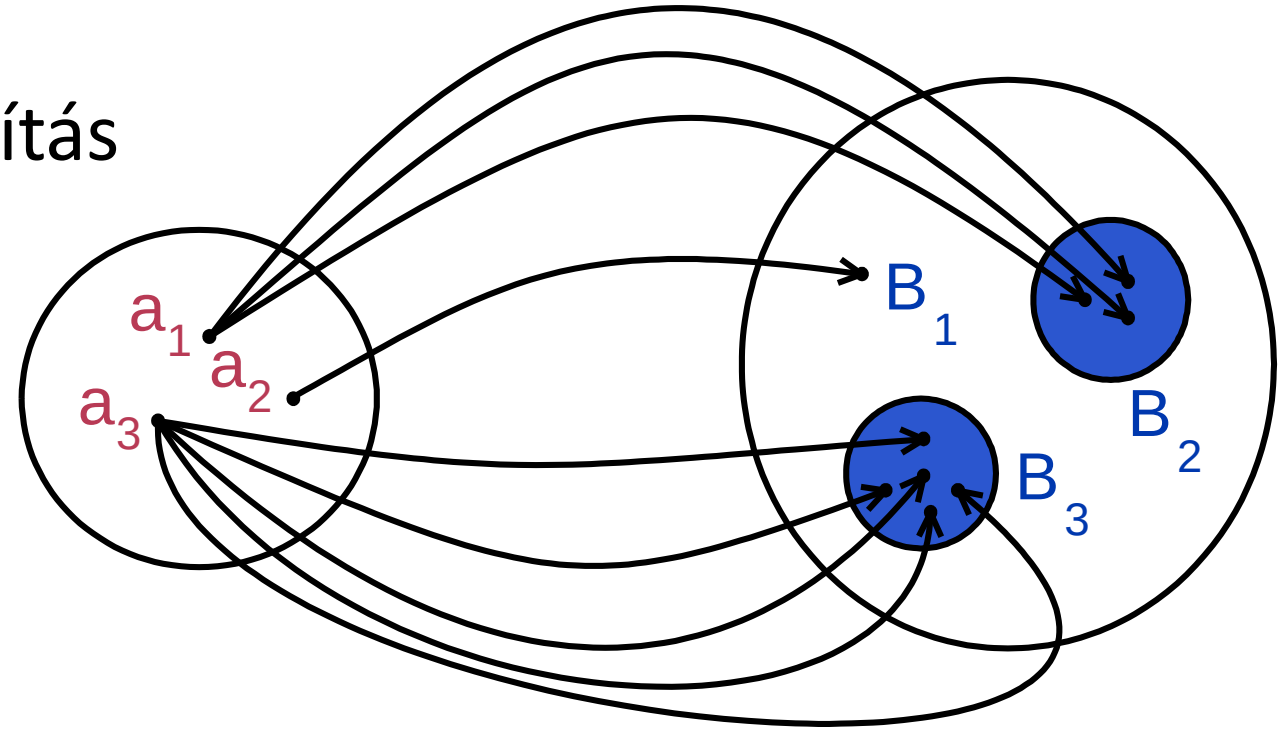
- Állapottér példák
  - {*hétfő*, *kedd*, *szerda*, *csütörtök*, *péntek*, *szombat*, *vasárnap*}
  - mikró állapotai: {*teljes fokozat*, *kiolvasztó mód*, *kikapcsolva*}
- Pillanatnyi állapot példák
  - ma *szerda* van
  - a mikró ajtaja *nyitva*

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

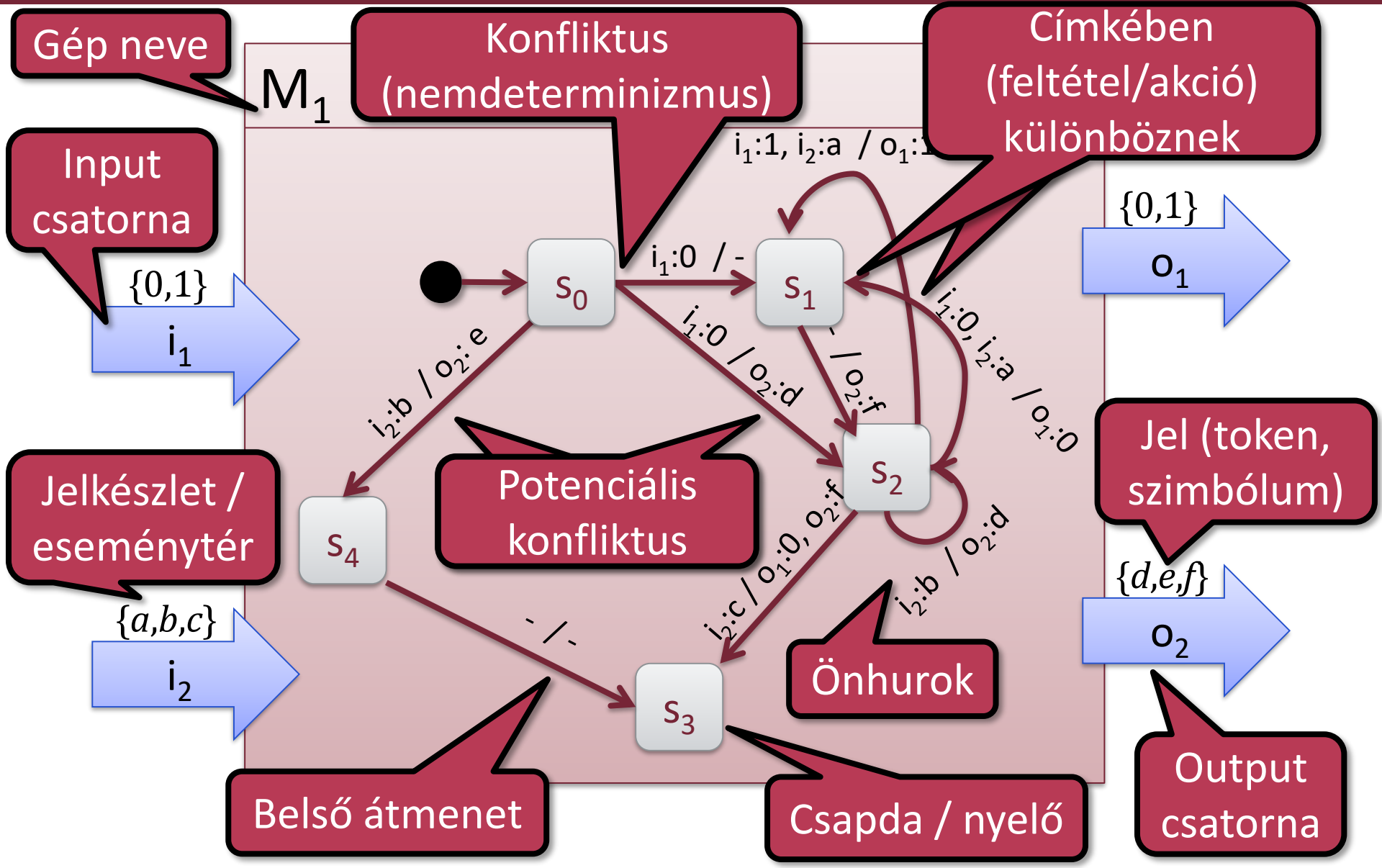
Az állapotterek **direkt szorzata** komponens állapottereken végzett **kompozíciós művelet.**

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

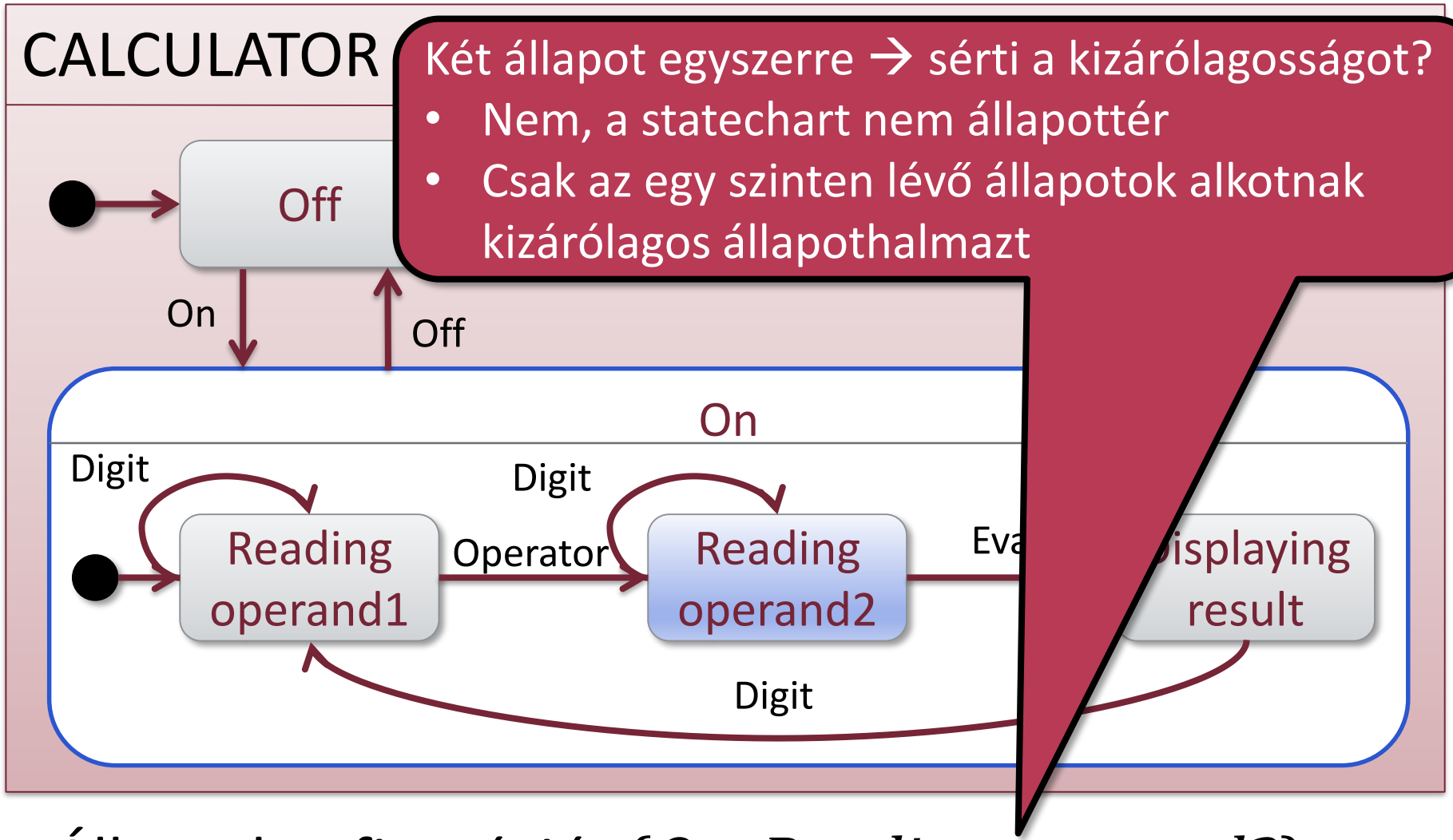
A szorzat állapottérben a komponens állapotterek minden állapot-kombinációjának egy összetett állapot felel meg.

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

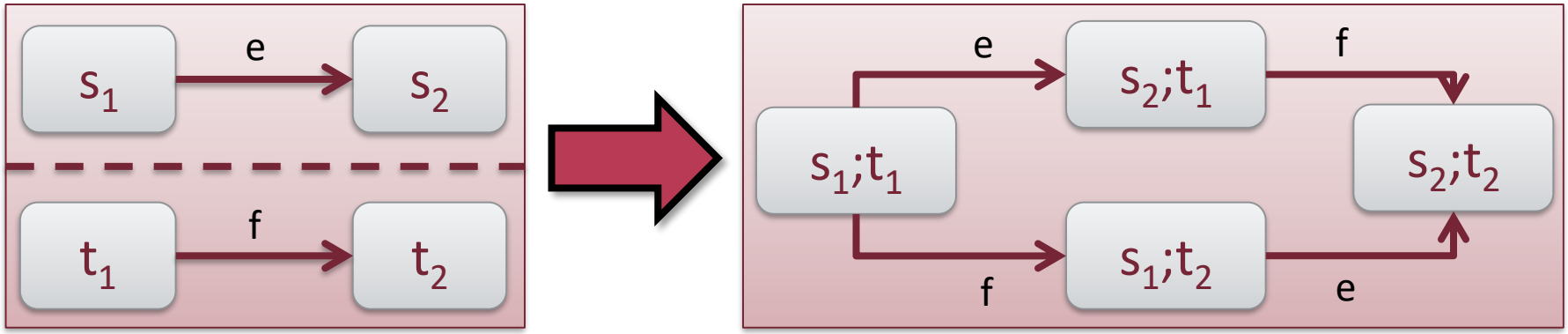


- Állapotkonfiguráció: $\{On, Reading\ operand2\}$

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
Diszkrét („nem pletykál”)	*Discreet* ☺
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