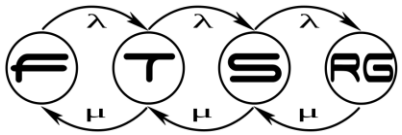
Állapot alapú modellezés

Budapesti Műszaki és Gazdaságtudományi Egyetem
Hibatűrő Rendszerek Kutatócsoport



Szolgálati közlemények

- Emelt gyakorlat két hét – 1 nap múlva
- Tárgy honlapját hét végén figyelni
  - HF információ
- Jövő héten opcionális kisfeladat
- Jövő előadáson HF demó

Tartalom

- Ismétlés, kitekintés
- Állapottér
- Állapotgépek
- Állapotgépek kiterjesztései
- Szoftvertámogatás

Bevezetés

Ismétlés, kitekintés

- Állapottér
- Állapotgépek
- Állapotgépek kiterjesztései
- Szoftvertámogatás

Ismétlés: felépítési vs. viselkedési modellek

- Felépítési (*structural*) modellek
  - Statikus
  - Rész és egész, összetevők
  - Kapcsolatok, összeköttetések
- Viselkedési (*behavioral*) modellek
  - Dinamikus
  - Időbeli lefolyás
  - Állapot, folyamat
  - Reakciók a külvilágra
- Nem fed le mindent, nem válik élesen szét…

Az autóban van kamera és kormányvezérlő

A kamera jeleket küld a sáv elhagyásáról (mennyit? mikor?)

A sávtartó rendszerben a kamera jeleit fogadva a kormányvezérlő beavatkozik (mikor/hogyan?)

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

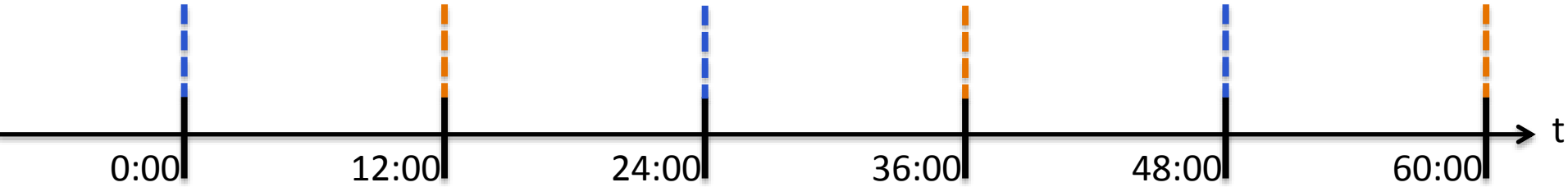
- Mi történik, ha megérintem a bal felső sarokban?
  - Q, q, 1 vagy =
  - Csak a „múlt” alapján eldönthető

Alapozás: diszkrét eseménysor

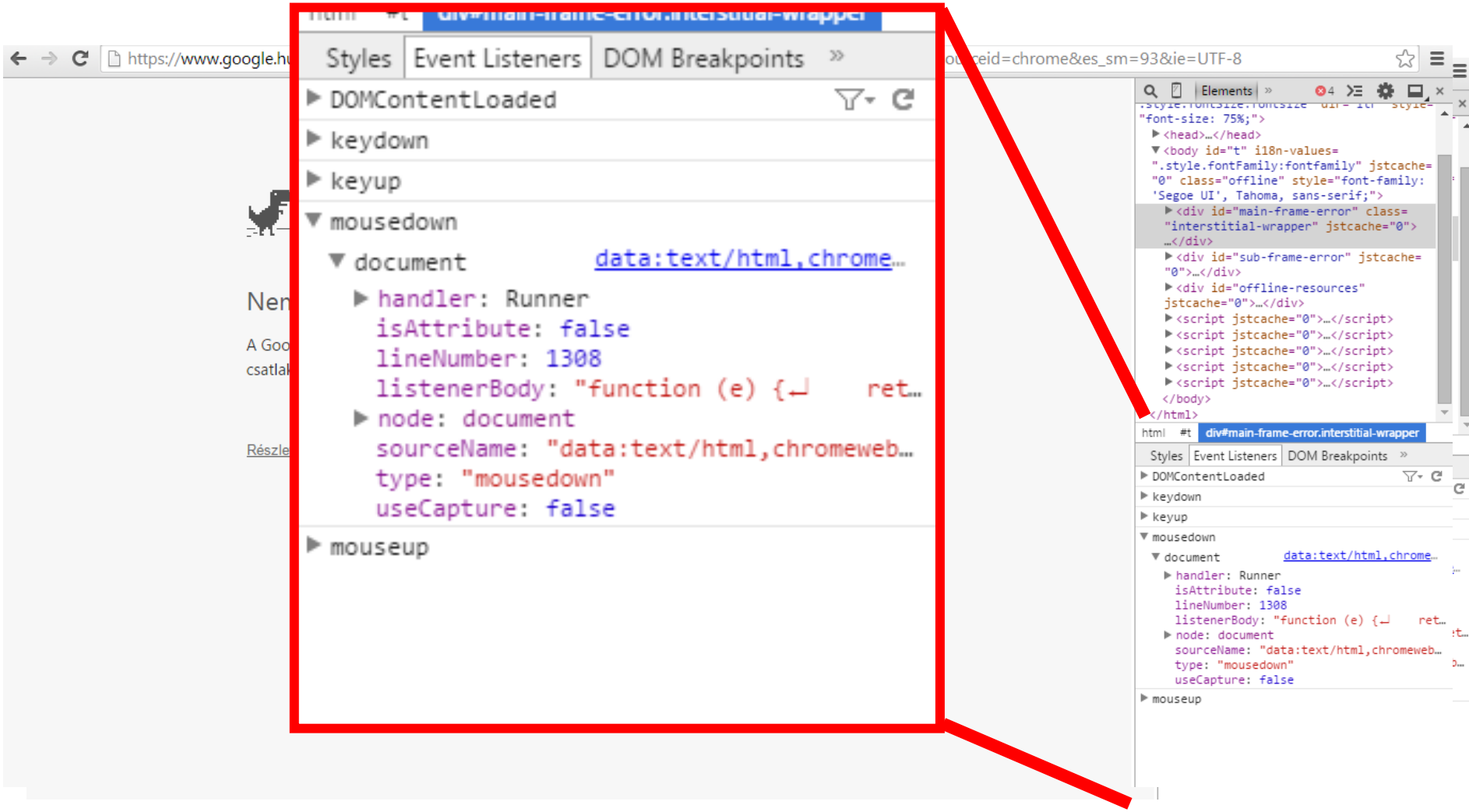
- **Esemény**
  - pillanatszerű változás (a rendszerben vagy ki/bemeneten)
- **Eseményfolyam**
  - Pl. input/output adatforrás
- **Eseménytér** – {megengedett események}
  - Beolvasható input értékek, kibocsátható output értékek
- Pillanatszerű események sora, egyszerre ≤ 1

Eseményfolyam: toronyóra ütései

Eseménytér: {*éjfél*, *dél*}



Eseményorientált programozás



Ismétlés, kitekintés

Állapottér

Állapotgépek

Állapotgépek kiterjesztései

Szoftvertámogatás

Kulcsfogalom: állapottér

- **Állapottér**
  - Egymástól megkülönböztetett **rendszerállapot**ok halmaza
  - Példák
    - Pl. {*hétfő*, *kedd*, *szerda*, *csütörtök*, *péntek*, *szombat*, *vasárnap*}
    - Pl. mikró állapotai: {*teljes fokozat*, *kiolvasztó mód*, *kikapcsolva*}
  - **DEF**: Az állapottér egy halmaz, amelynek minden időpontban *pontosan egy* eleme jellemzi a rendszert.
- **Pillanatnyi állapot** ???
  - Pl. most *szerda* van, a mikró ajtaja *nyitva*
  - **DEF**: Egy adott időpontban a rendszer **pillanatnyi állapota** az állapottér azon egyetlen eleme, amelyik abban az időpontban jellemző a rendszerre.

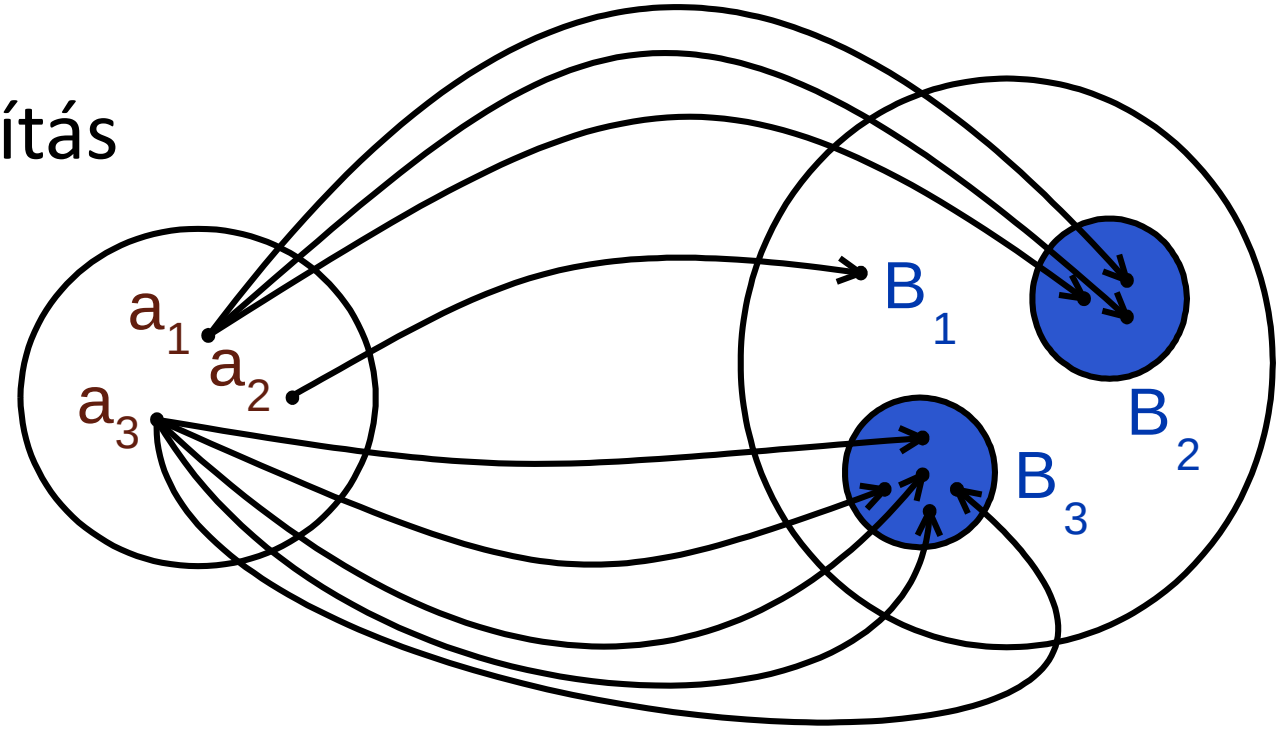
Az állapottér tulajdonságai

- **Teljesség**
  - Mindig legalább az egyik állapot fennáll
  - Ellenpélda
    - $\{hétfő, kedd, csütörtök, szombat\}$ nem teljes
- Kölcsönös **kizárólagosság**
  - Egyszerre csak egy állapot állhat fent
  - Ellenpéldák (nem alkalmasak állapottérnek!)
    - $\{hétköznap, hétvége, délután\}$ nem kizárólagos (holott teljes)
    - mikróra $\{ajtaja\ tárva, kikapcsolva\}$

Állapotfinomítás, állapotabsztrakció

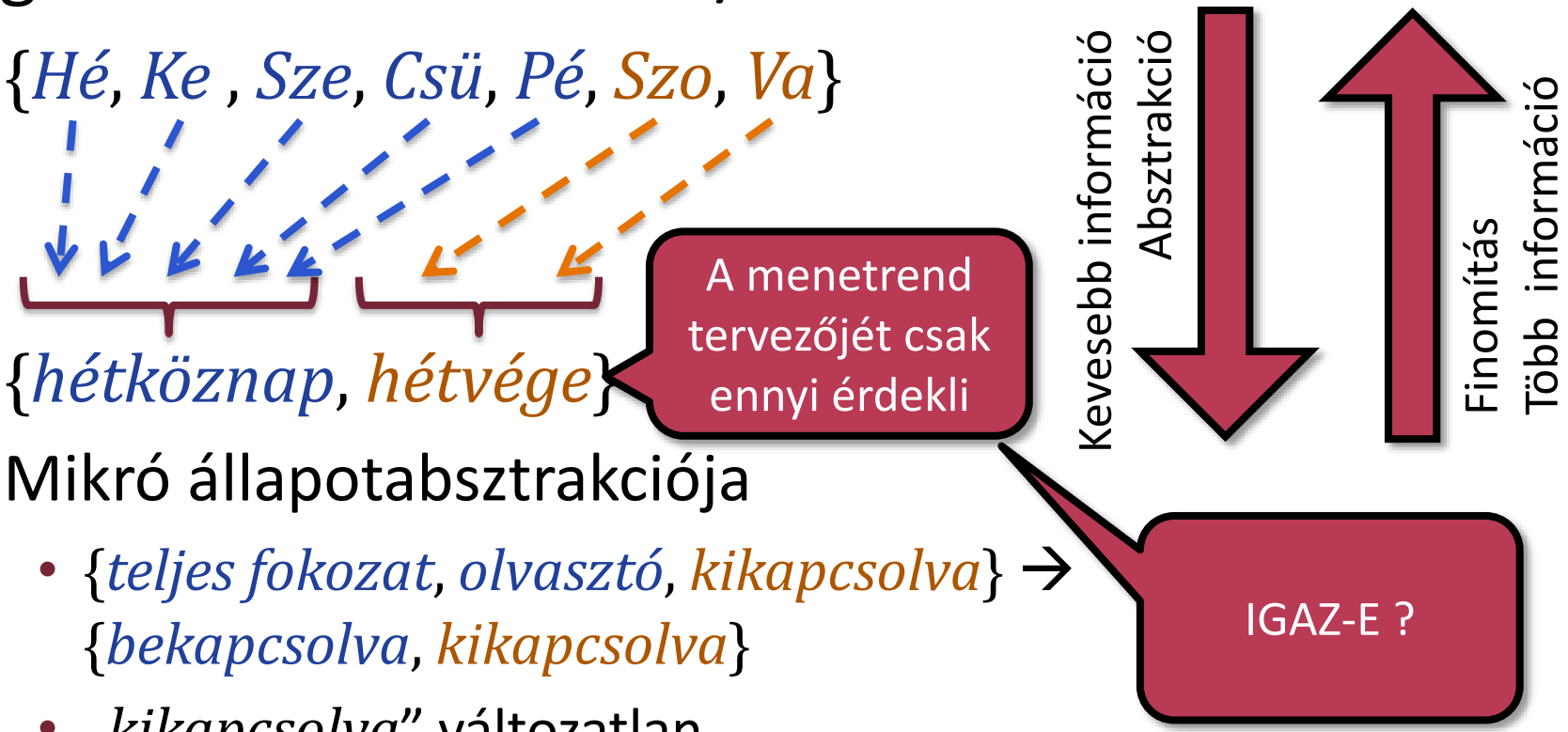
- Állapotfinomítás/absztrakció: az állapottéren végzett halmazfinomítás/absztrakció

Ismétlés:
halmazfinomítás



Állapotfinomítás, állapotabsztrakció

- Állapotfinomítás/absztrakció: az állapottéren végzett halmazfinomítás/absztrakció
  - {*Hé*, *Ke* , *Sze*, *Csü*, *Pé*, *Szo*, *Va*}
  - {*hétköznap*, *hétvége*}
  - Mikró állapotabsztrakciója
    - {*teljes fokozat*, *olvasztó*, *kikapcsolva*} → {*bekapcsolva*, *kikapcsolva*}
    - „*kikapcsolva*” változatlan



Állapotfinomítás, állapotabsztrakció

- 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?

Állapotfinomítás, állapotabsztrakció

- 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)

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*}

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

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

- „A szorzatnál ***finomabb*** kompozit állapottér”
  - … mivel két kompozit állapot előfordulását kizártuk
  - Itt a finomított állapottérnek van *kevesebb* eleme!
    - V.ö.: állapotfinomítás esetén *nőtt* az állapotok száma
    - A finomítás után 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
- Miért dekomponálunk?
  - Állapotváltozók külön kezelhetőek
  - Állapotváltozók külön tárolhatóak

Kitekintés: szakmai példák

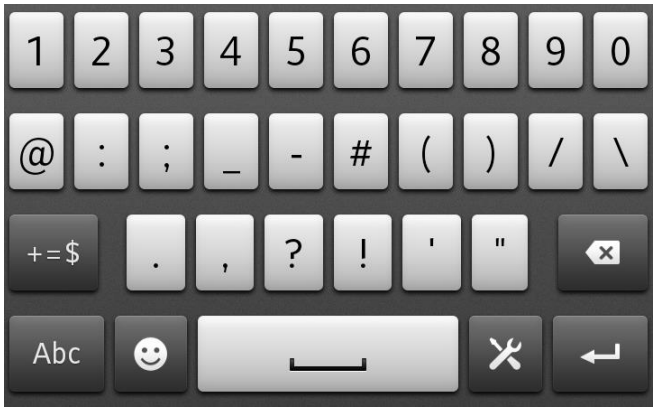
- Hol jön elő az állapot alapú modellezés az IT-ban?
- Közösségi háló – Jancsi és Juliska ismeretsége
  - (ettől függ néhány funkció, pl. milyen képek látszanak)
  - Állapotváltozók:
    - Jancsi bejelölte-e Juliskát
    - Vice versa

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
  - 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

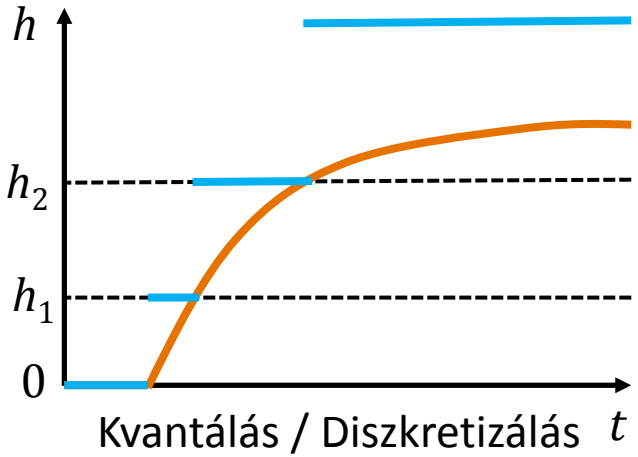
Kitekintés: szakmai példák

- 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;
```

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 $\in \mathbb{N}$)

Ismétlés, kitekintés

Állapottér

Állapotgépek

Állapotgépek kiterjesztései

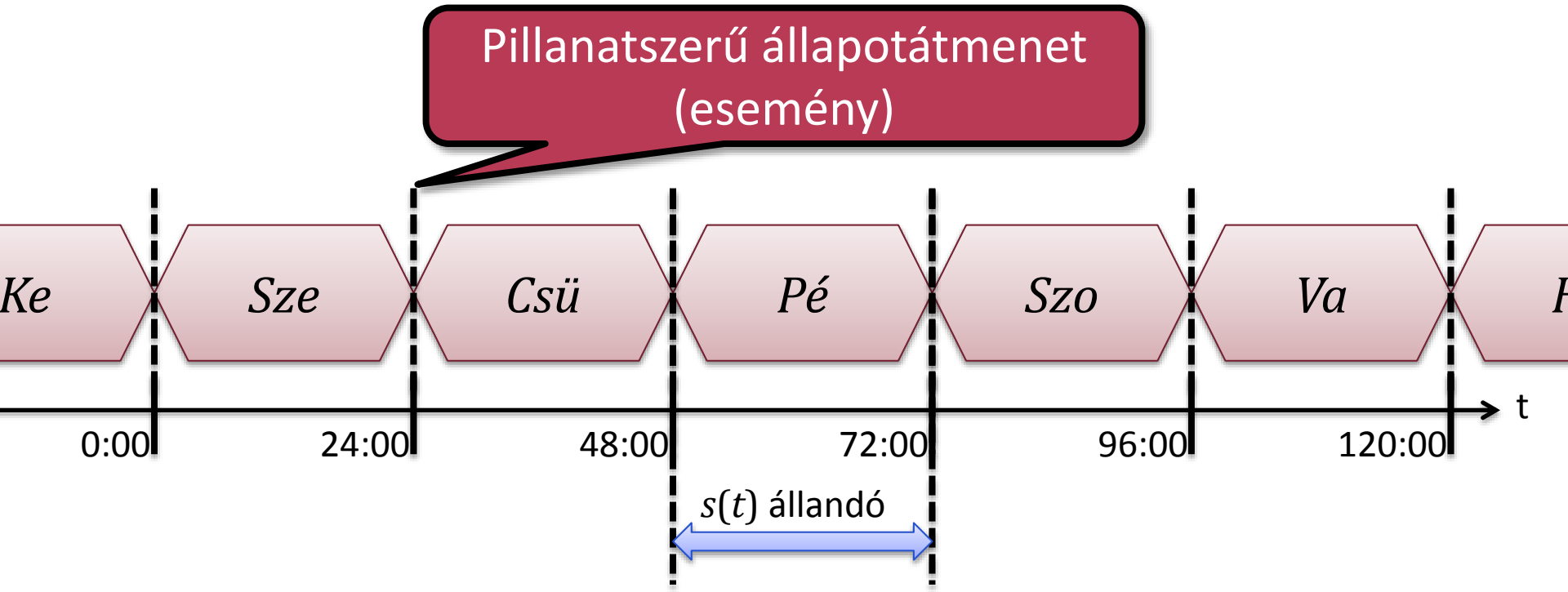
Szoftvertámogatás

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

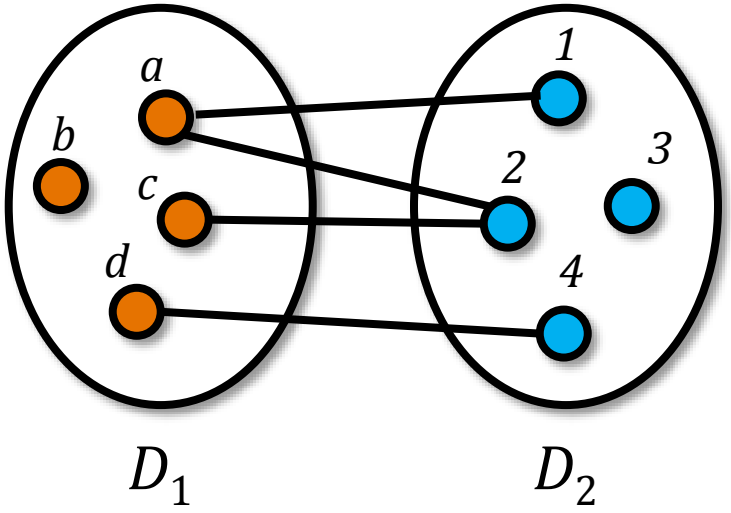
Állapotátmenetek

- 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
  - $R \subseteq D_1 \times D_2$



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

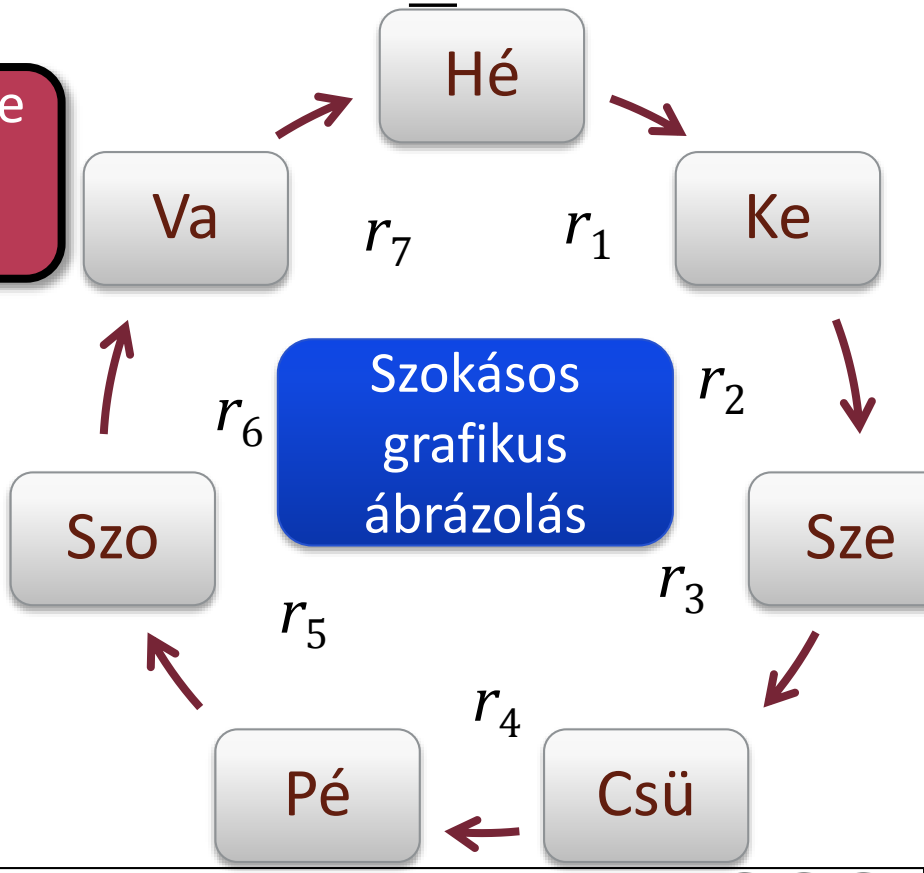
Állapotátmenetek

- 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 → minden átmenet engedélyezett
    - Pl. $S_{\text{kismacska}}=\{alszik, játszik, iszik\}$
  - … 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

A NAGY KÍSÉRLET AZ USÁBAN

Strukturális dekompozíció

A vizsgálati objektum

Bal oldali kávéfőző

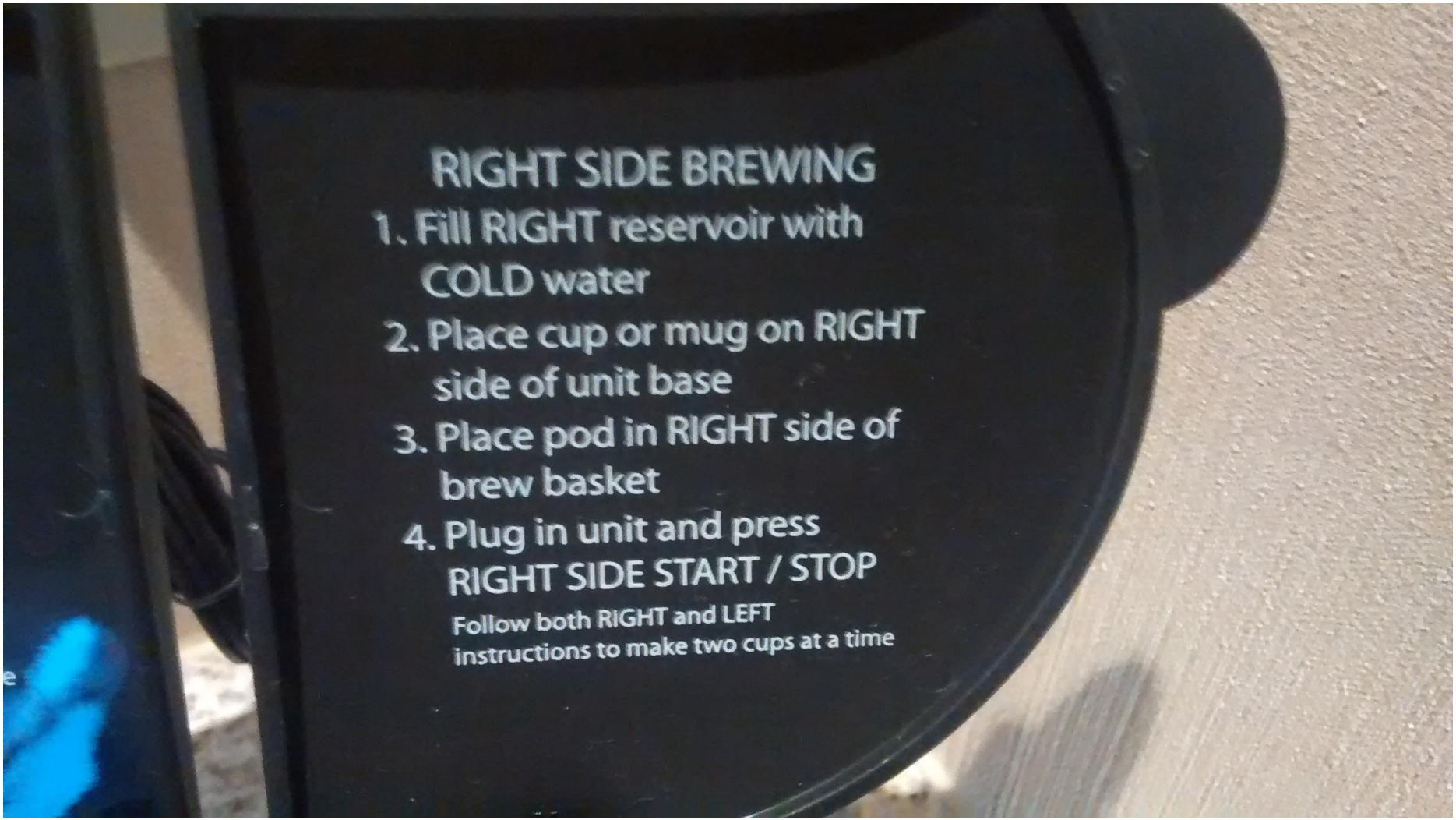
Jobb oldali kávéfőző

Kapcsoló


LEFT SIDE
START / STOP
UNIT WILL BEEP WHEN FINISHED
RIGHT SIDE
START / STOP
UNIT WILL BEEP WHEN FINISHED
Cuisinart

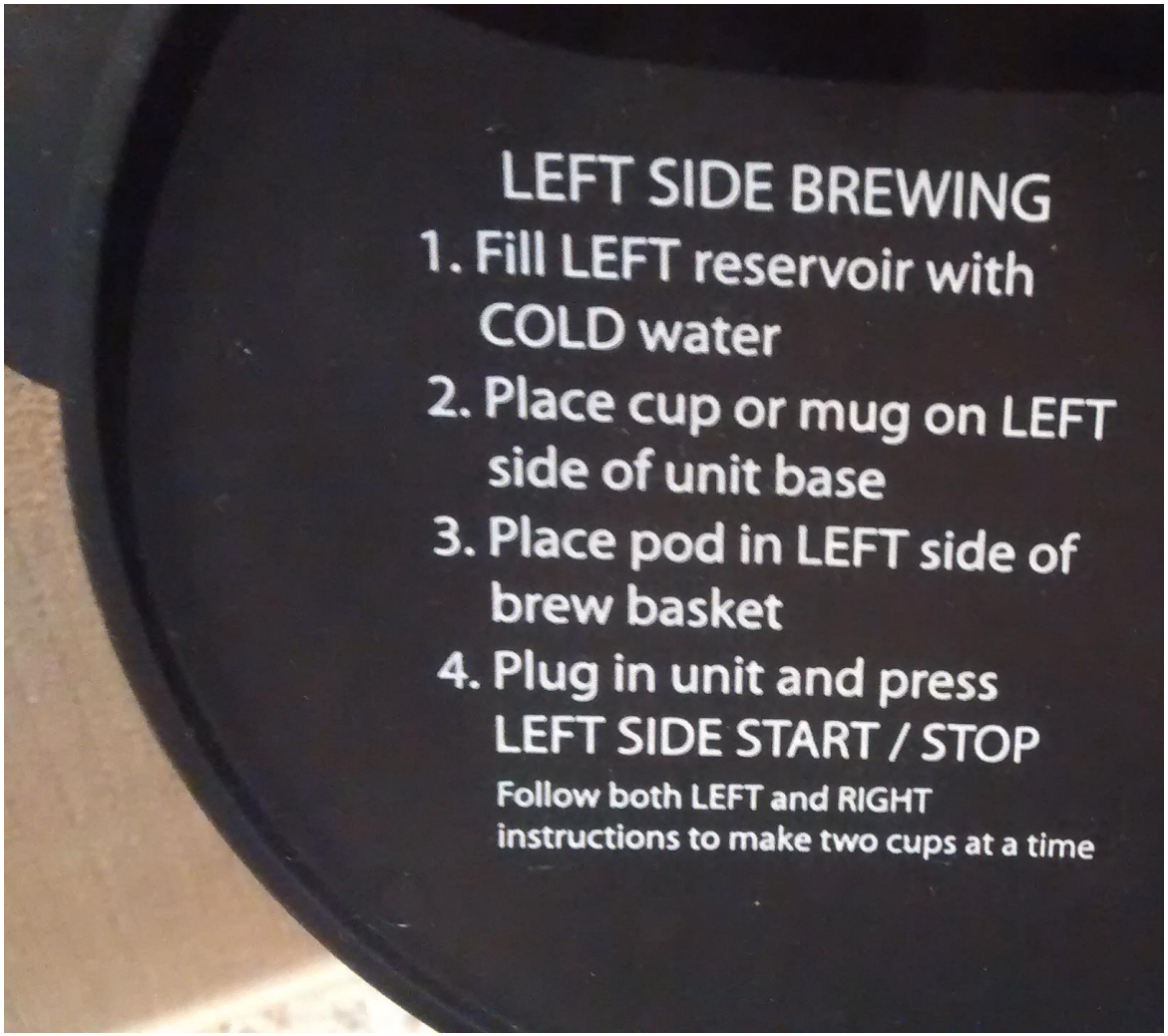
Kávétartó

Jobb fedél


RIGHT SIDE BREWING
1. Fill RIGHT reservoir with COLD water
2. Place cup or mug on RIGHT side of unit base
3. Place pod in RIGHT side of brew basket
4. Plug in unit and press RIGHT SIDE START / STOP
Follow both RIGHT and LEFT instructions to make two cups at a time

FUNKCIONÁLIS MODELL

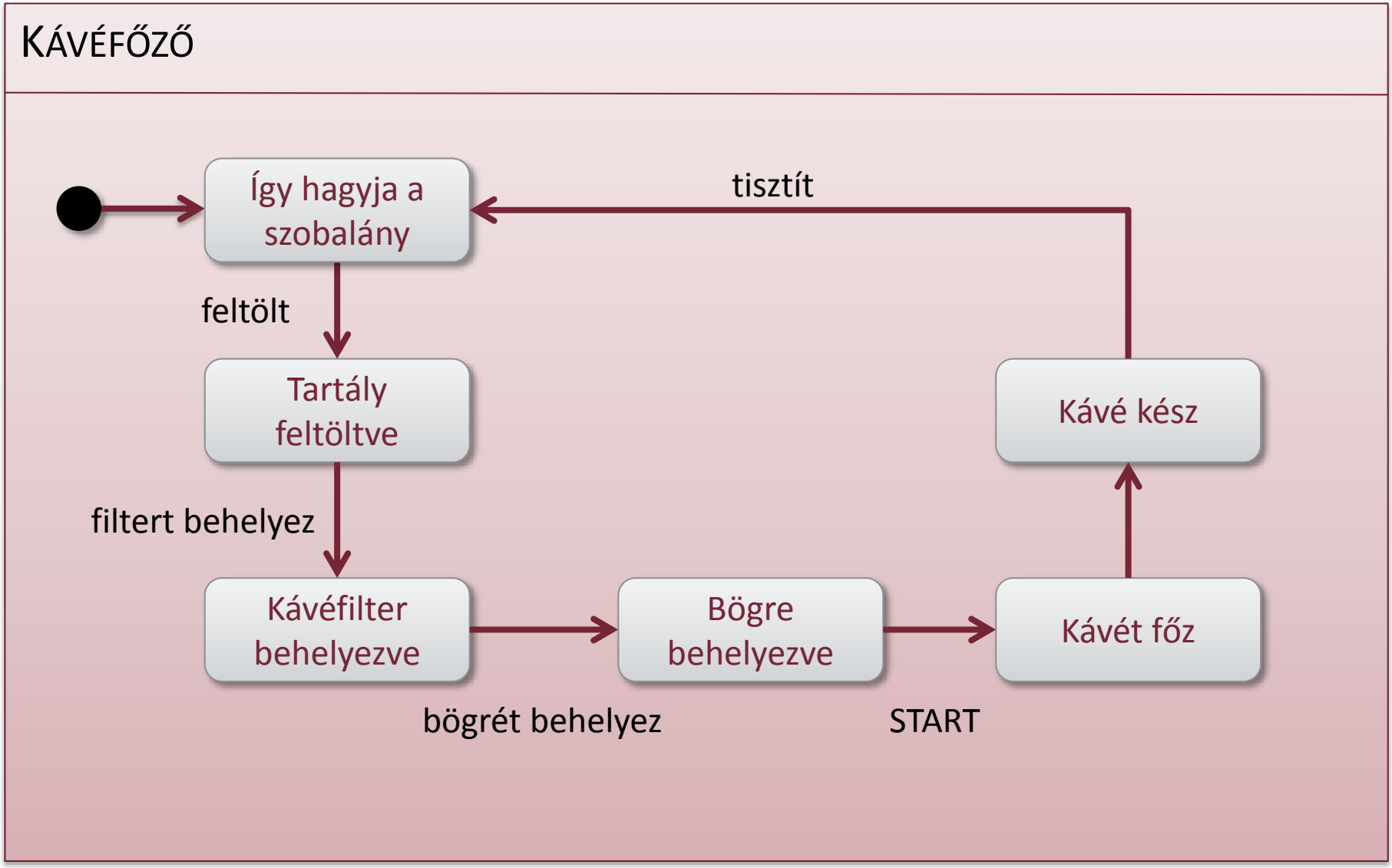
Bal fedél



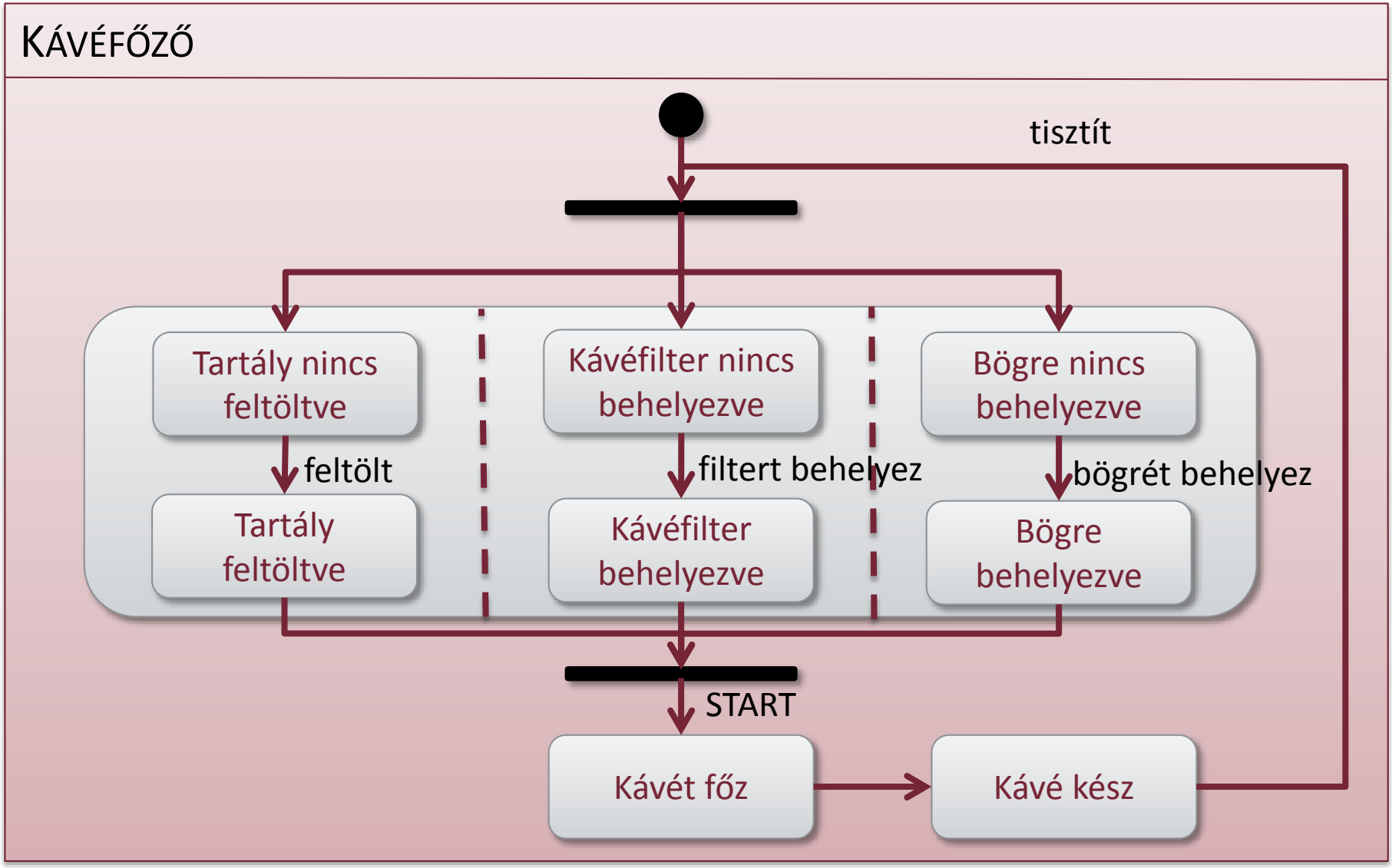
1. Töltse meg a BAL tartályt hideg vízzel
2. Tegyen egy bögrét vagy poharat a BAL pohártartóra
3. Tegyen be egy kávépárnát a BAL oldali tartóba
4. Dugja be a kábelt és nyomja meg a BAL OLDAL START/STOP gombot

Ha egyszerre két kávét akar főzni, egyszerre kövesse a BAL és JOBB utasításokat

Állapotgráf példa: Kávéfőző



Példa: Kávéfőző

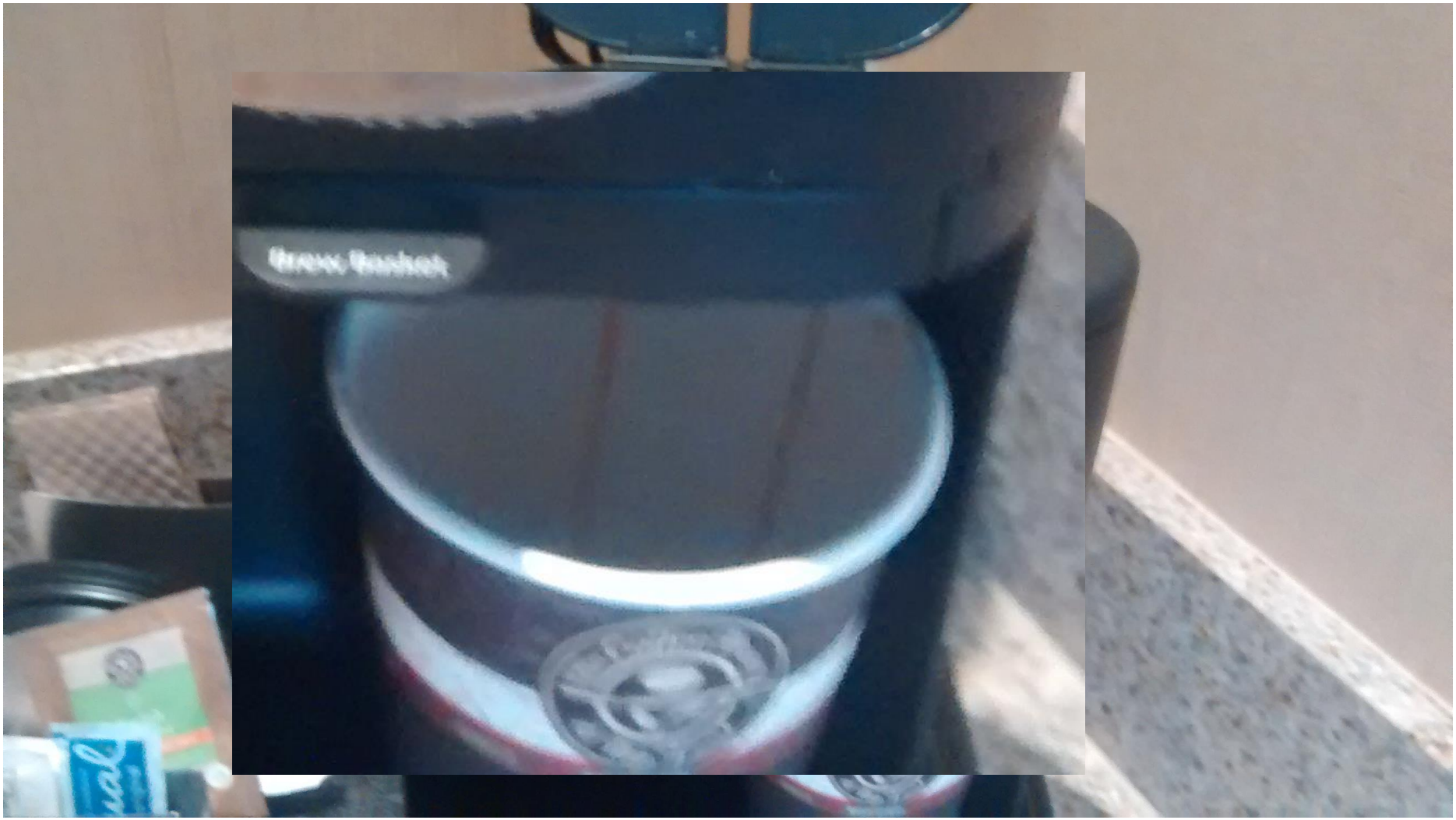


IMPLEMENTÁCIÓ TESZTELÉSE

Sikertelen teszt

Sikertelen teszt

Sikeres teszt



Hibadiagnosztika

Tanulságok

- A strukturális és funkcionális dekompozíció
  - Általában független
  - Azonos modulok – egy leírás
    - OO modellezés: egy viselkedéstípus → több példány
- Ellenőrzés modell és implementáció összehasonlításával
  - Verifikáció: *jól építem-e a rendszert?*
  - Modell alapú tesztelés
- A professzor olyan őrült, aki
  - Kávéját automatelmélettel főzi
  - Cipőjét diffegyenletekkel köti meg.

Ismétlés, kitekintés

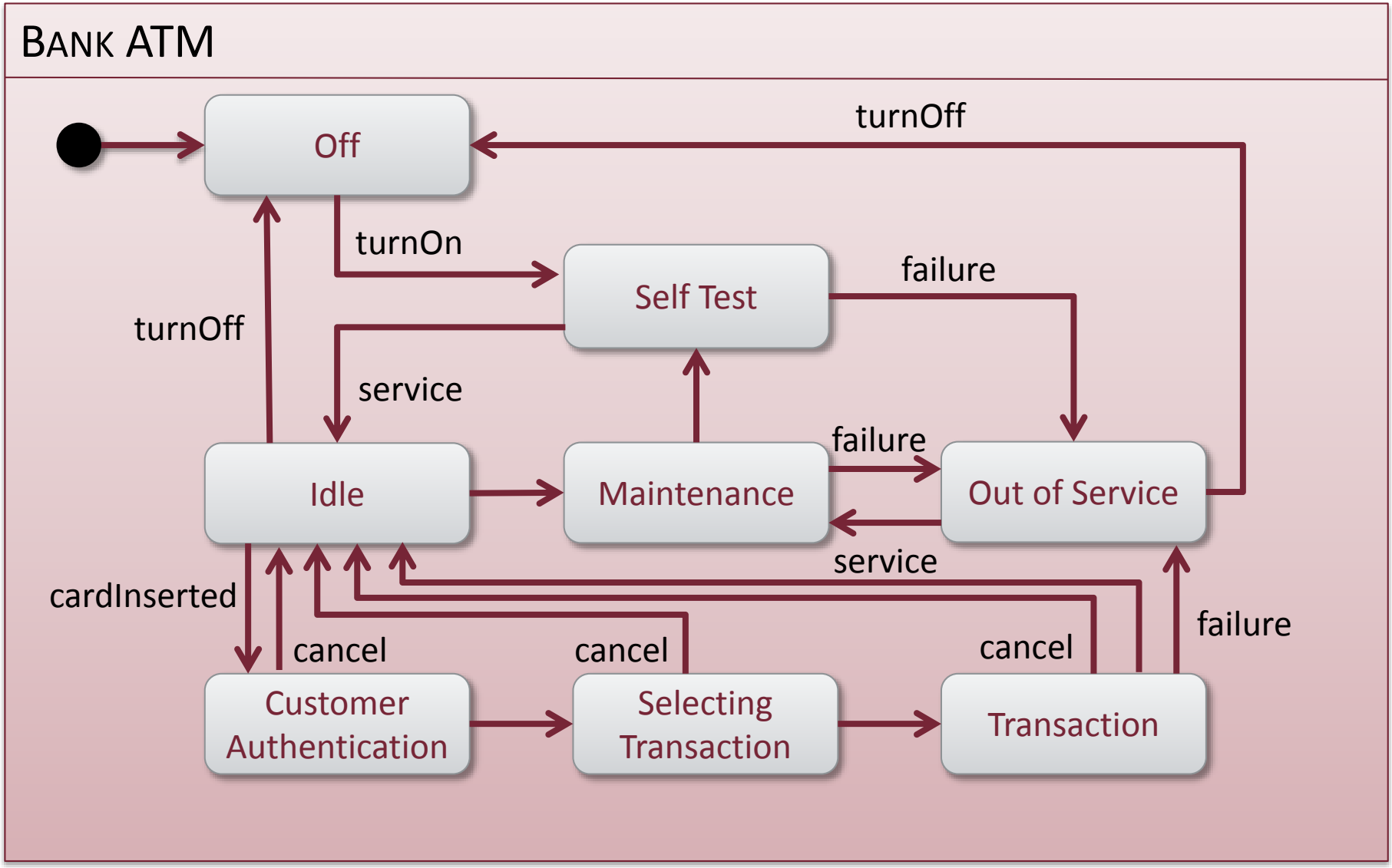
Állapottér

Állapotgépek

Állapotgépek kiterjesztései

Szoftvertámogatás

Állapotgráf példa: ATM

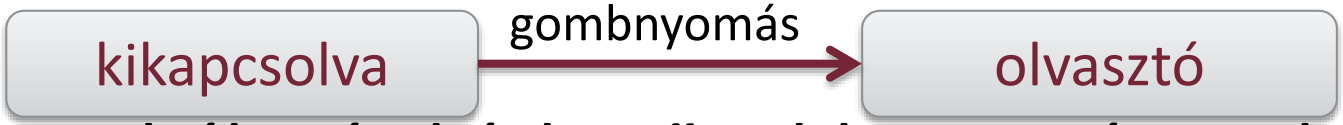


Állapotátmenet címkézése eseménnyel

- Átmeneti szabály címkéje
  - Pillanatszerű esemény
  - Az átmenet csak az eseménnyel együtt következhet be

- Különféle értelmezés: lehet az esemény…
  - … az állapotátmenet következménye (posztkondíció/utófeltétel)

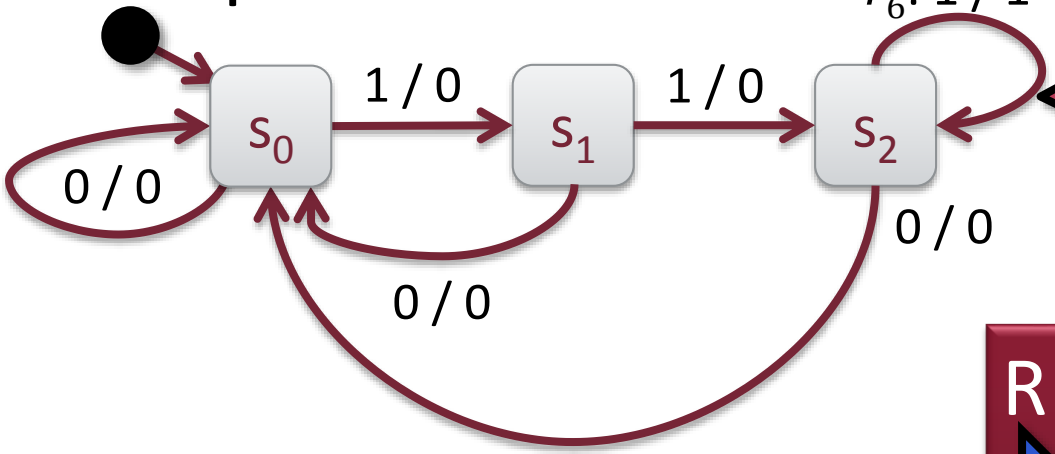
  - … az állapotátmenet oka (prekondíció/előfeltétel)



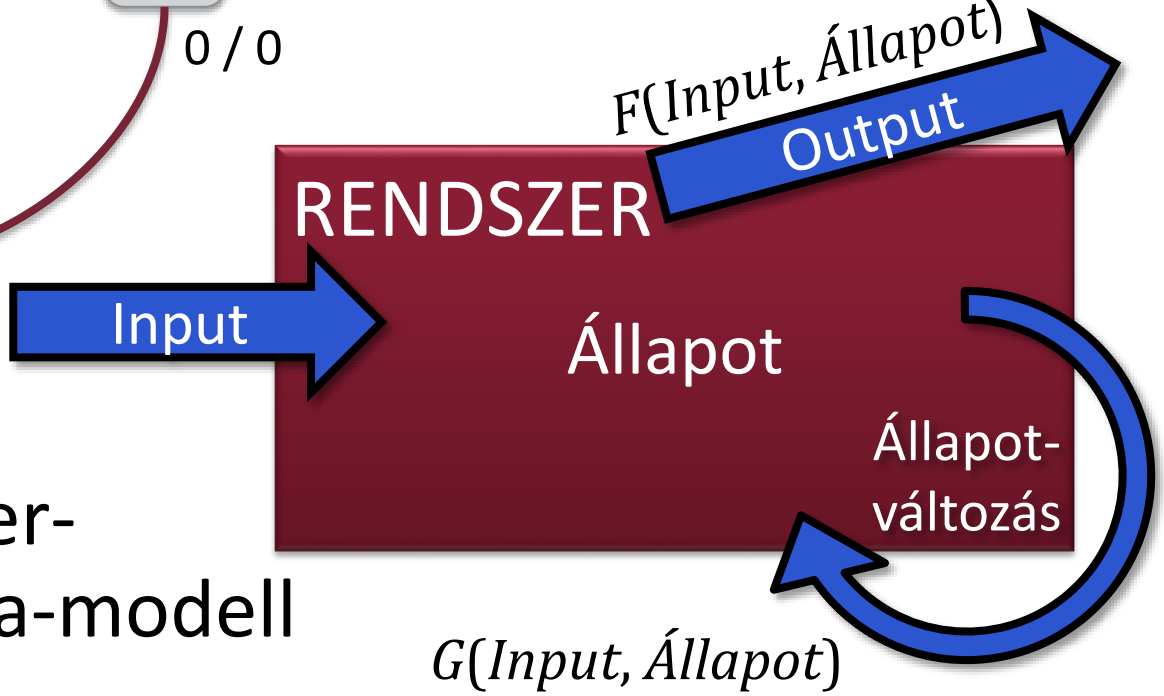
- Egy szabály címkézhető több eseménnyel is
  - input beolvasás / output kiírás

Emlékeztető: Mealy véges automata

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



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

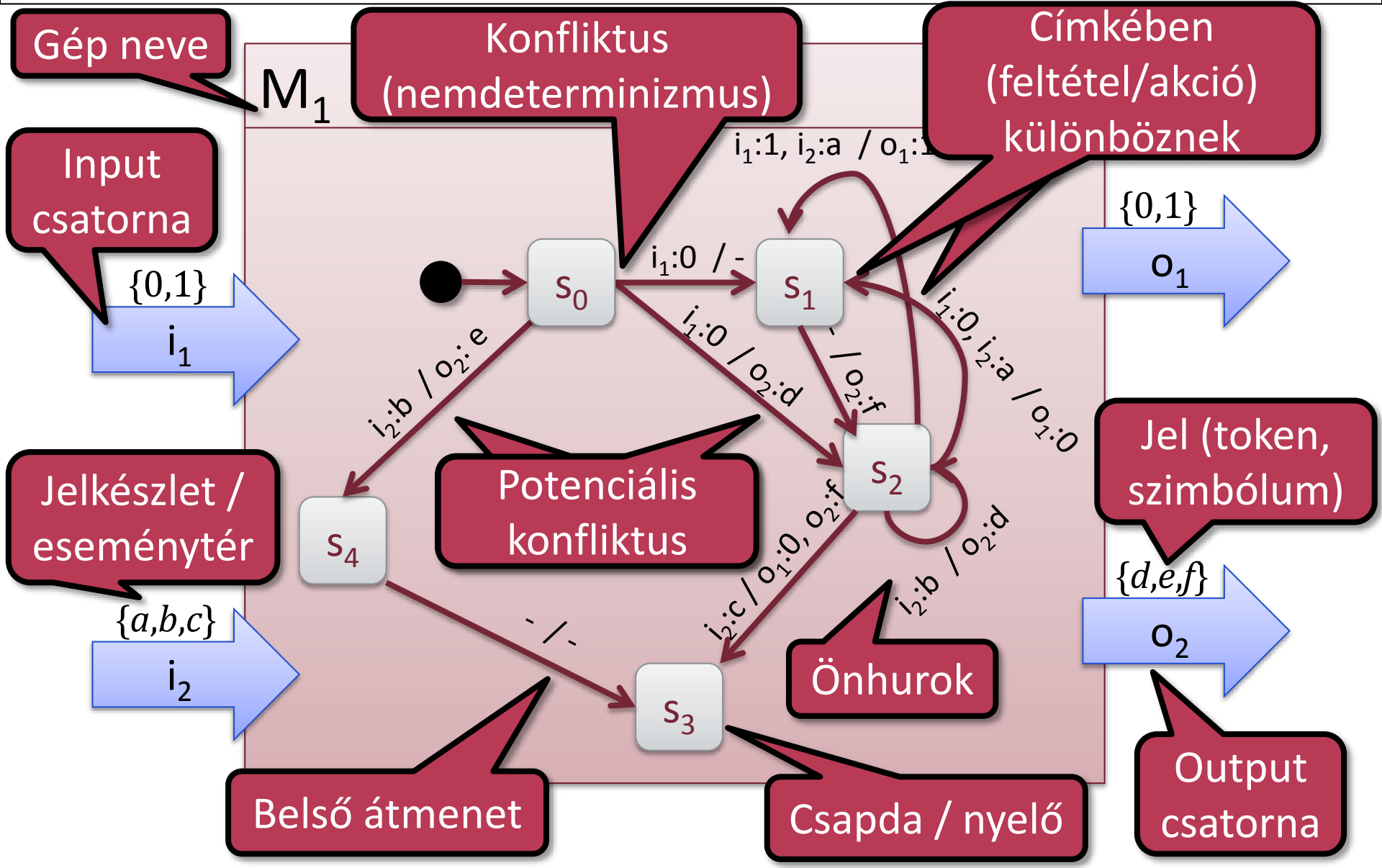


- Klasszikus rendszer-elméleti automata-modell

A Mealy-gép kiterjesztései

- Nemdeterminisztikus modell (vs. input)
- Input olvasás nélküli 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

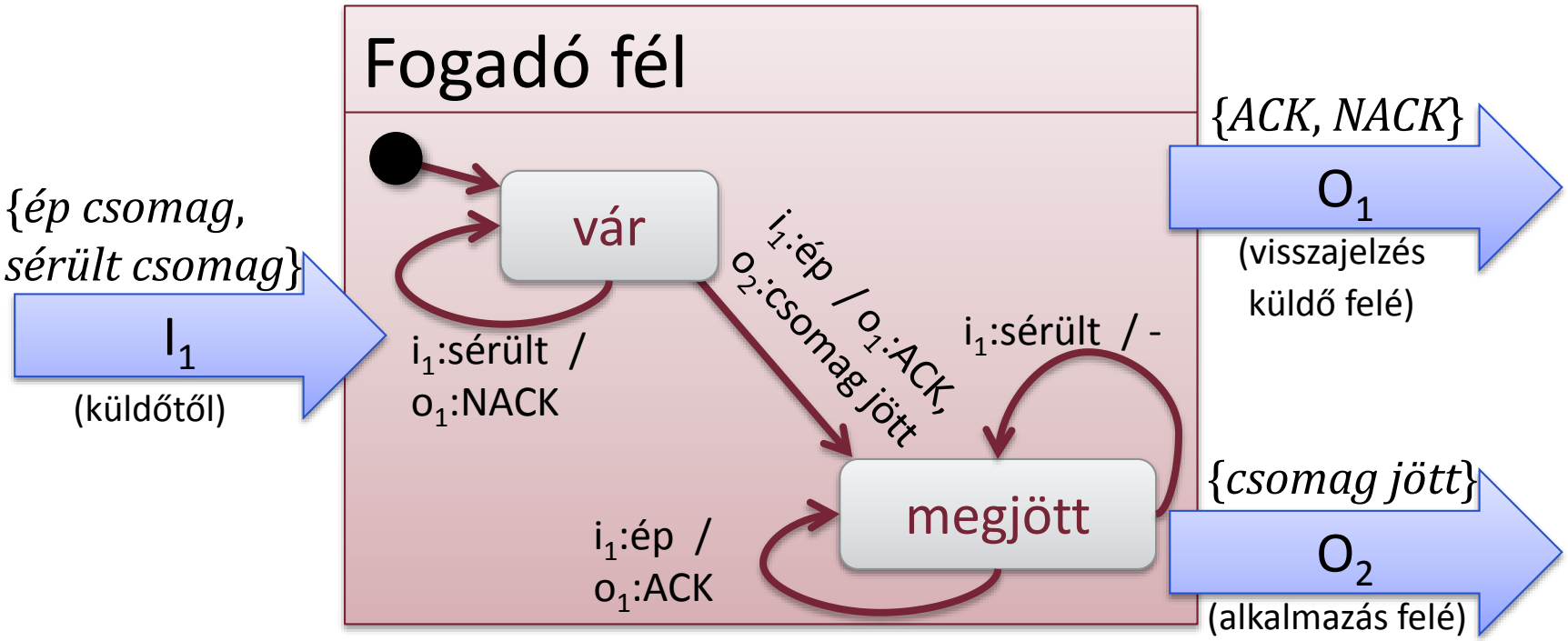


Specifikálatlan input

- 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
    - Pl. kritikus beágyazott rendszereknél
    - Mindenképp kell “visszajelzés” az eseményre
    - Az is érvénytelen, ha több szabály van (determinizmus kell)

Kitekintés: szakmai példák

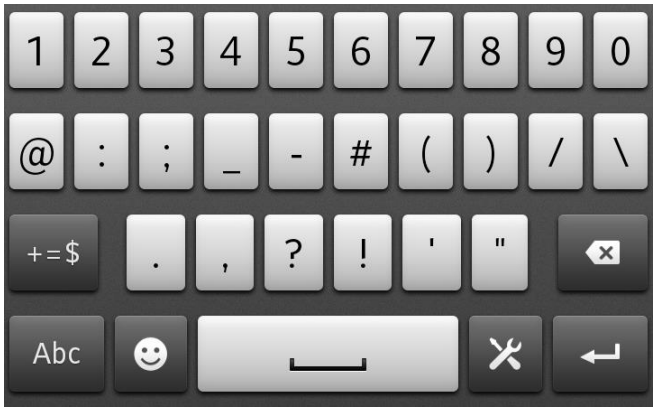
- Csomag alapú adatátviteli protokoll
  - Csomagok elveszhetnek, megsérülhetnek
  - Nyugtázás, újraküldés



- (Bővebben ld. Kommunikációs hálózatok 1. tárgy)

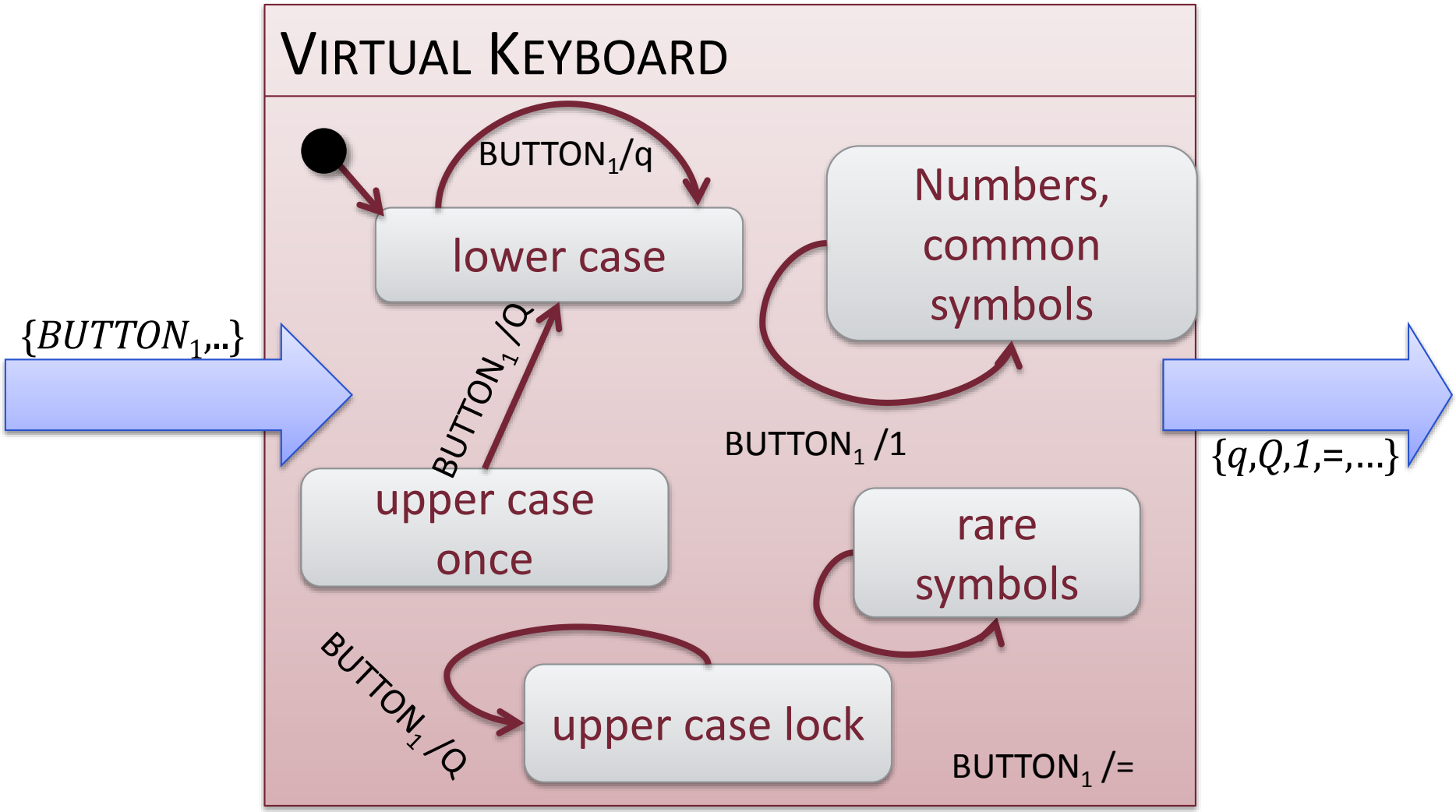
Emlékeztető

- Virtuális billentyűzet érintőképernyőre



Kitekintés: szakmai példák

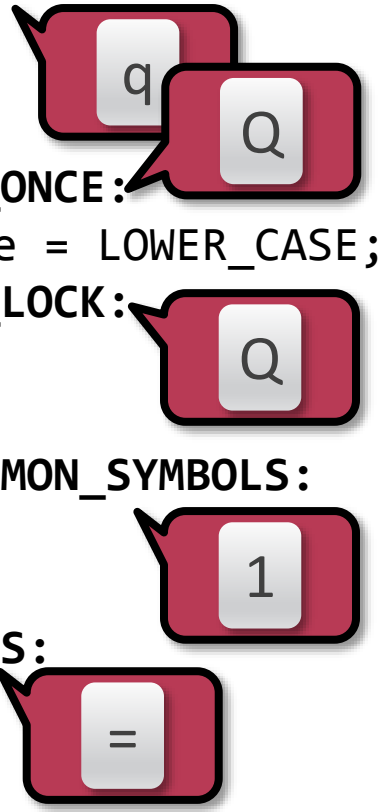
- Virtuális billentyűzet (részleges) állapotgépe



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

Tartalom

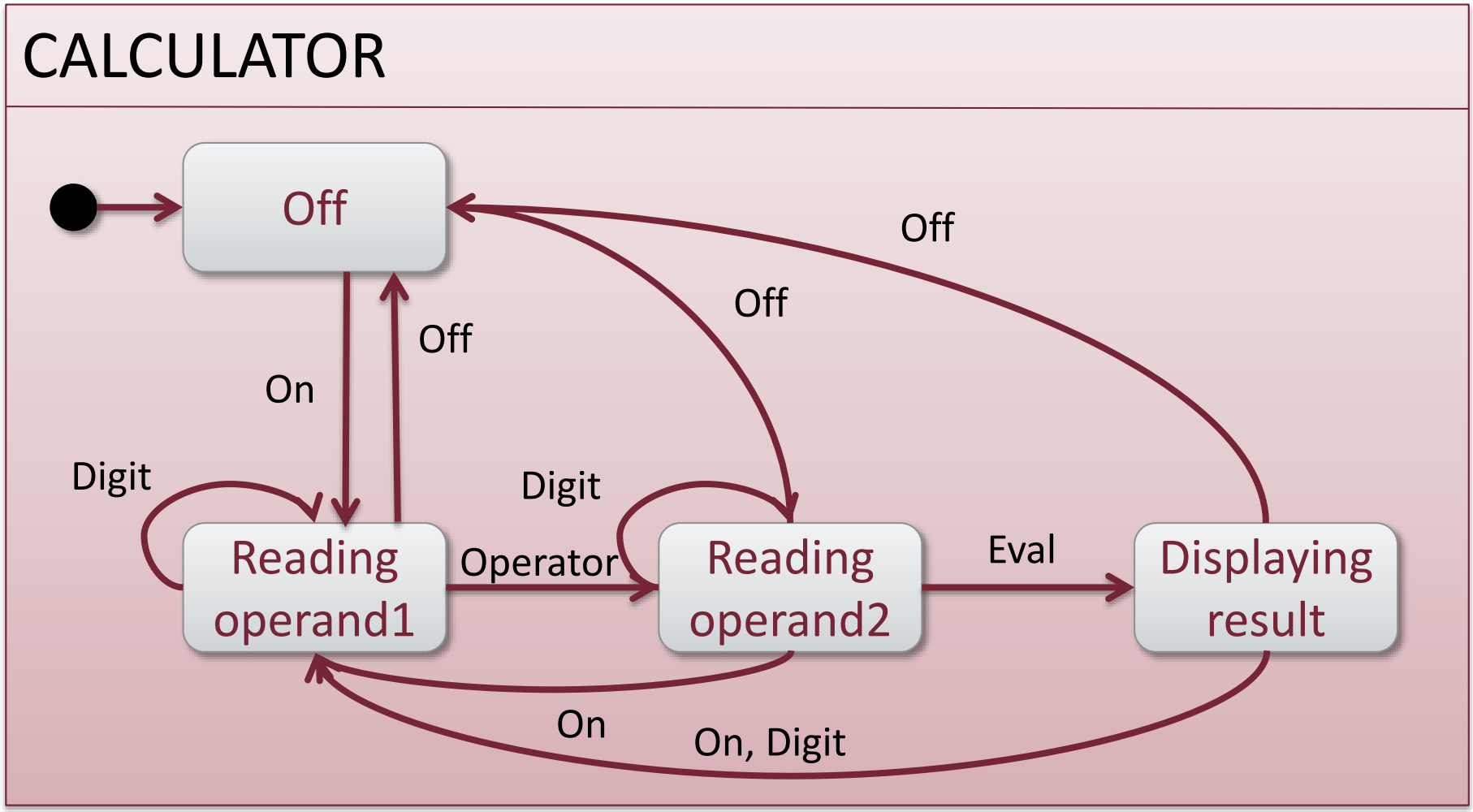
- Ismétlés, kitekintés
- Állapottér
- Állapotgépek
- **Állapotgépek kiterjesztései**
- Szoftvertámogatás

Statechart nyelvek

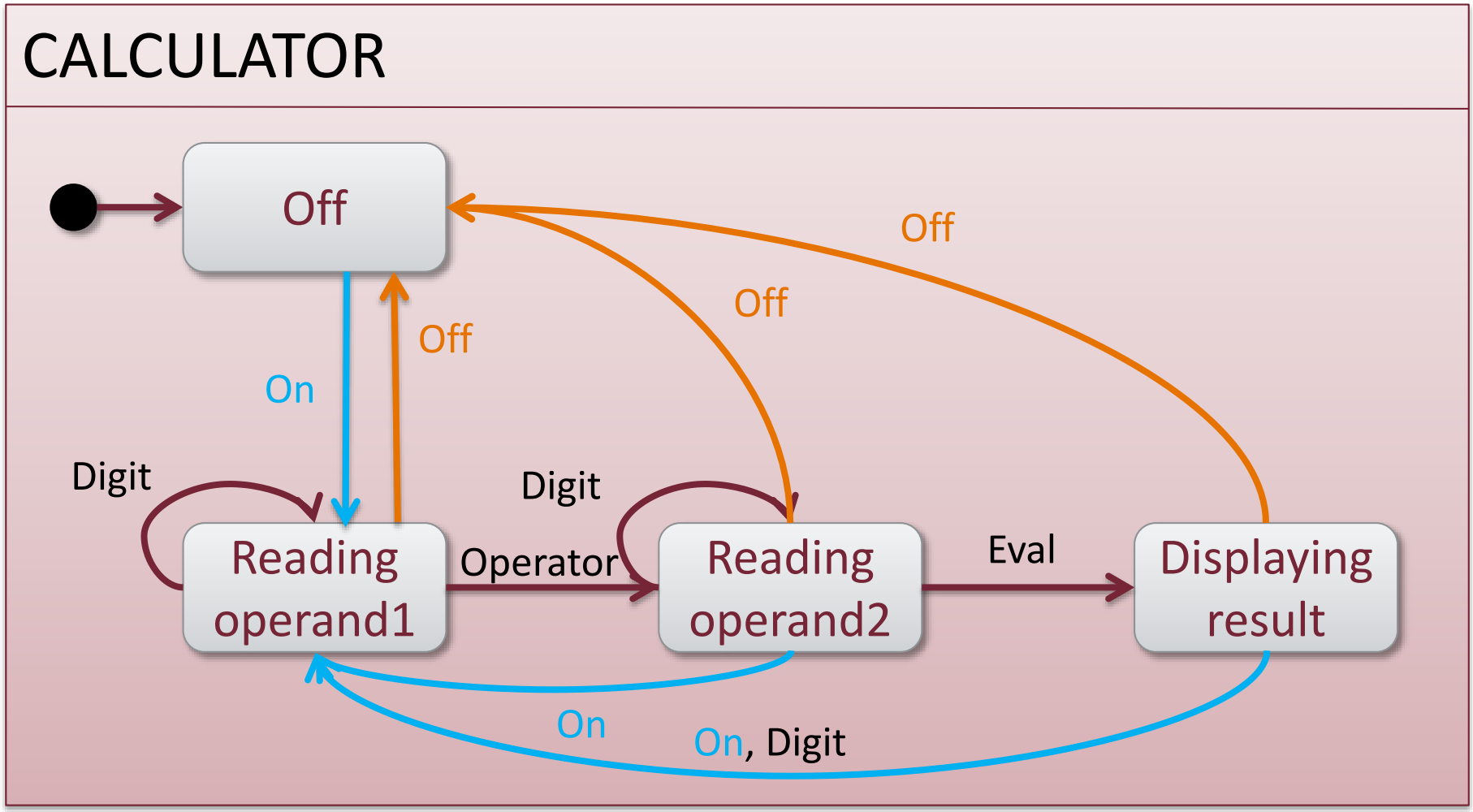
- Statechart: állapotgép +
  - Állapothierarchia
  - Ortogonalitás
  - Változók
  - Pszeudoállapotok
  - …
- Pl.
  - Yakindu (HF)
  - UML (SzoftTech)

Számológép

Állapothierarchia

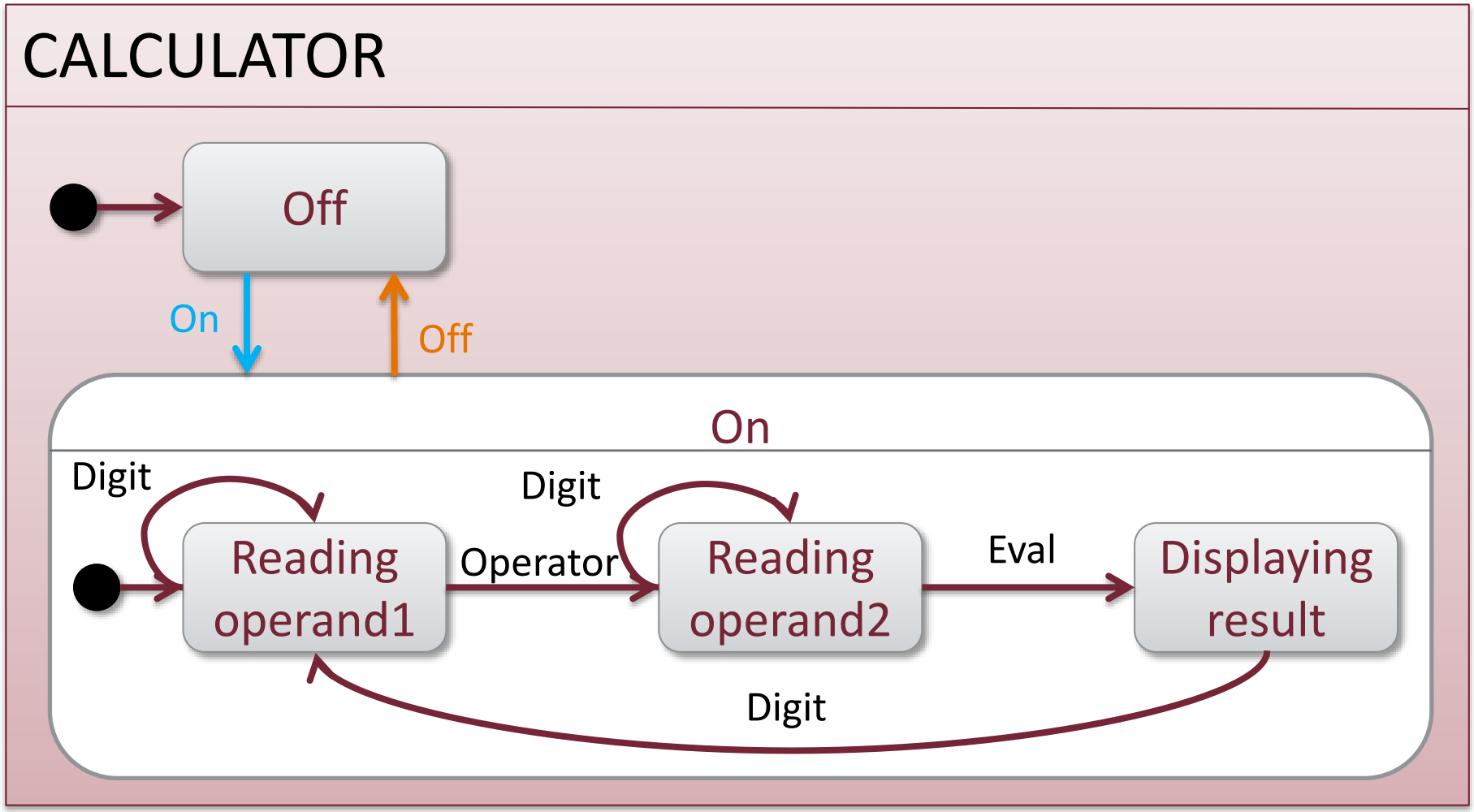


Állapothierarchia



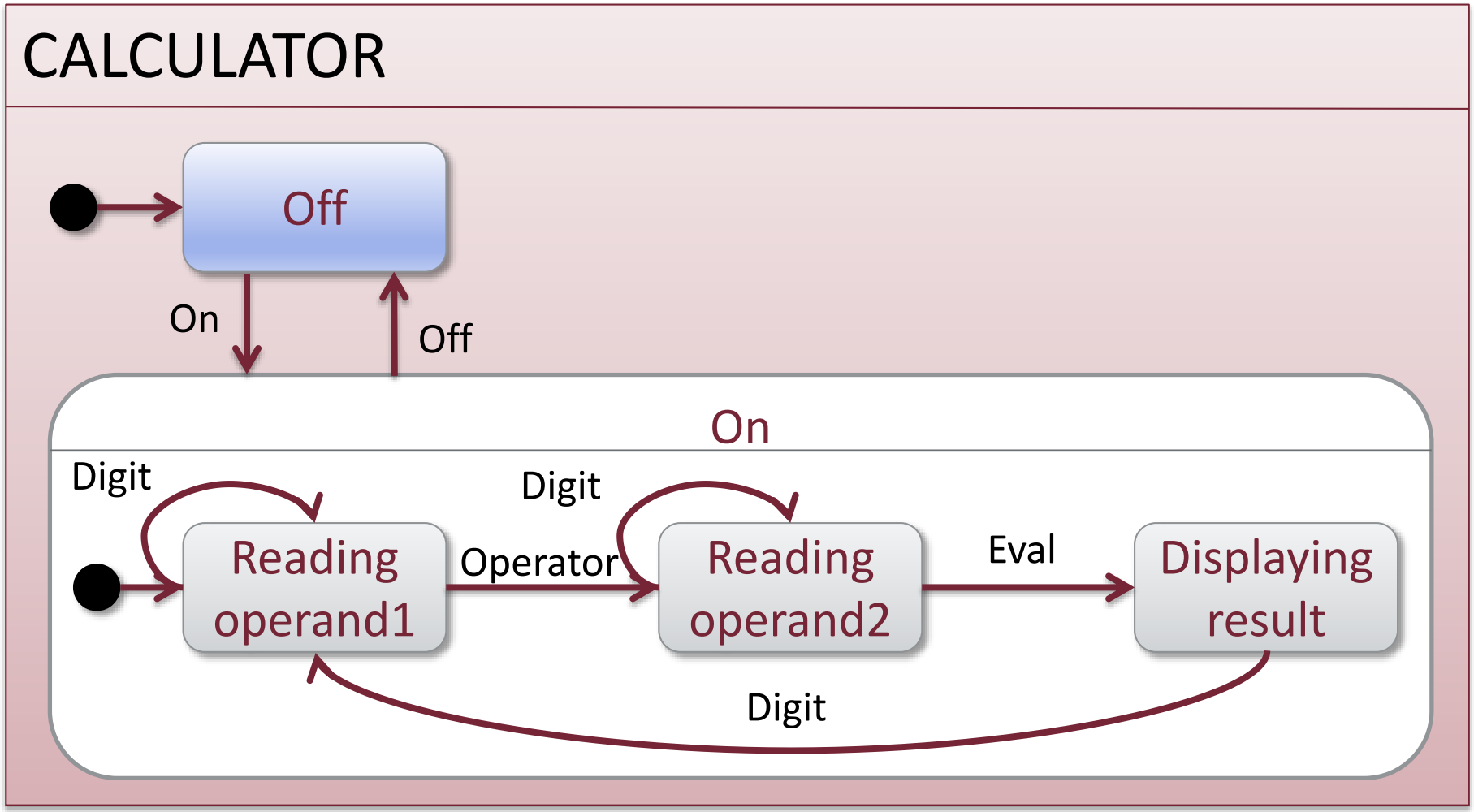
- Bemenet: {*on, off, digit, operand, eval*}
- Feltételezés: két operandusú műveleteink vannak

Állapothierarchia



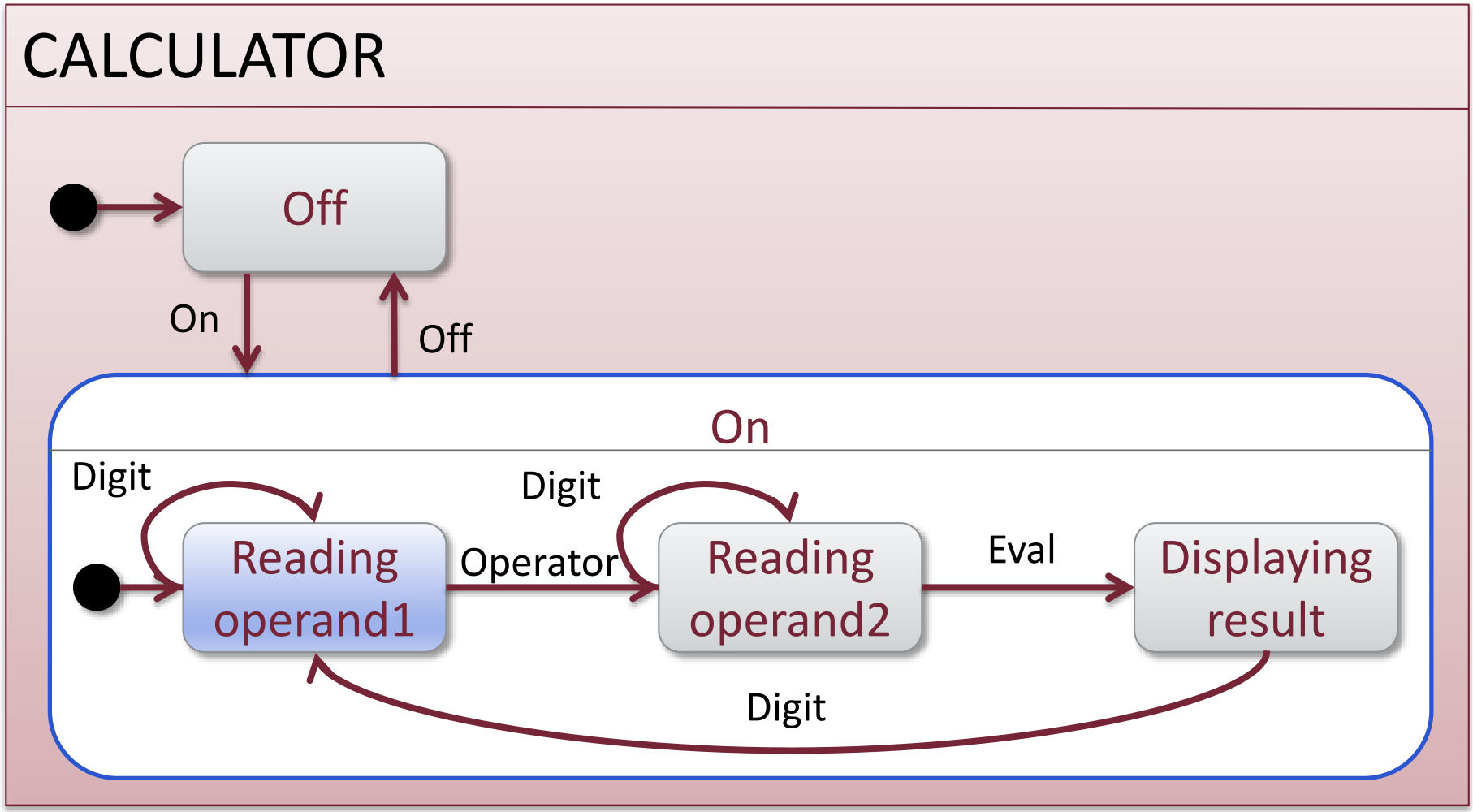
- Közös viselkedés kiemelése közös absztrakcióba

Állapothierarchia



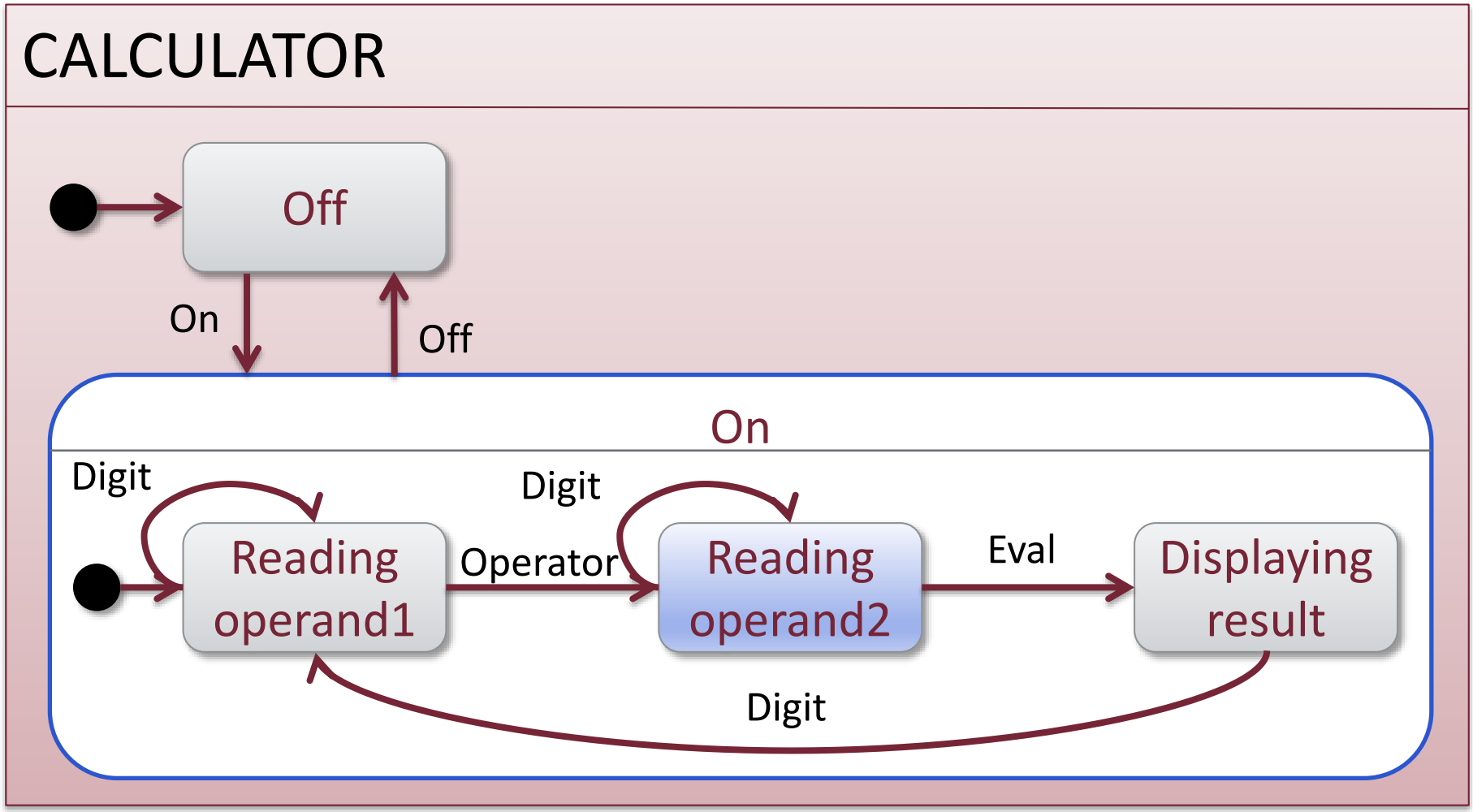
- Állapotkonfiguráció: {*Off*}

Állapothierarchia



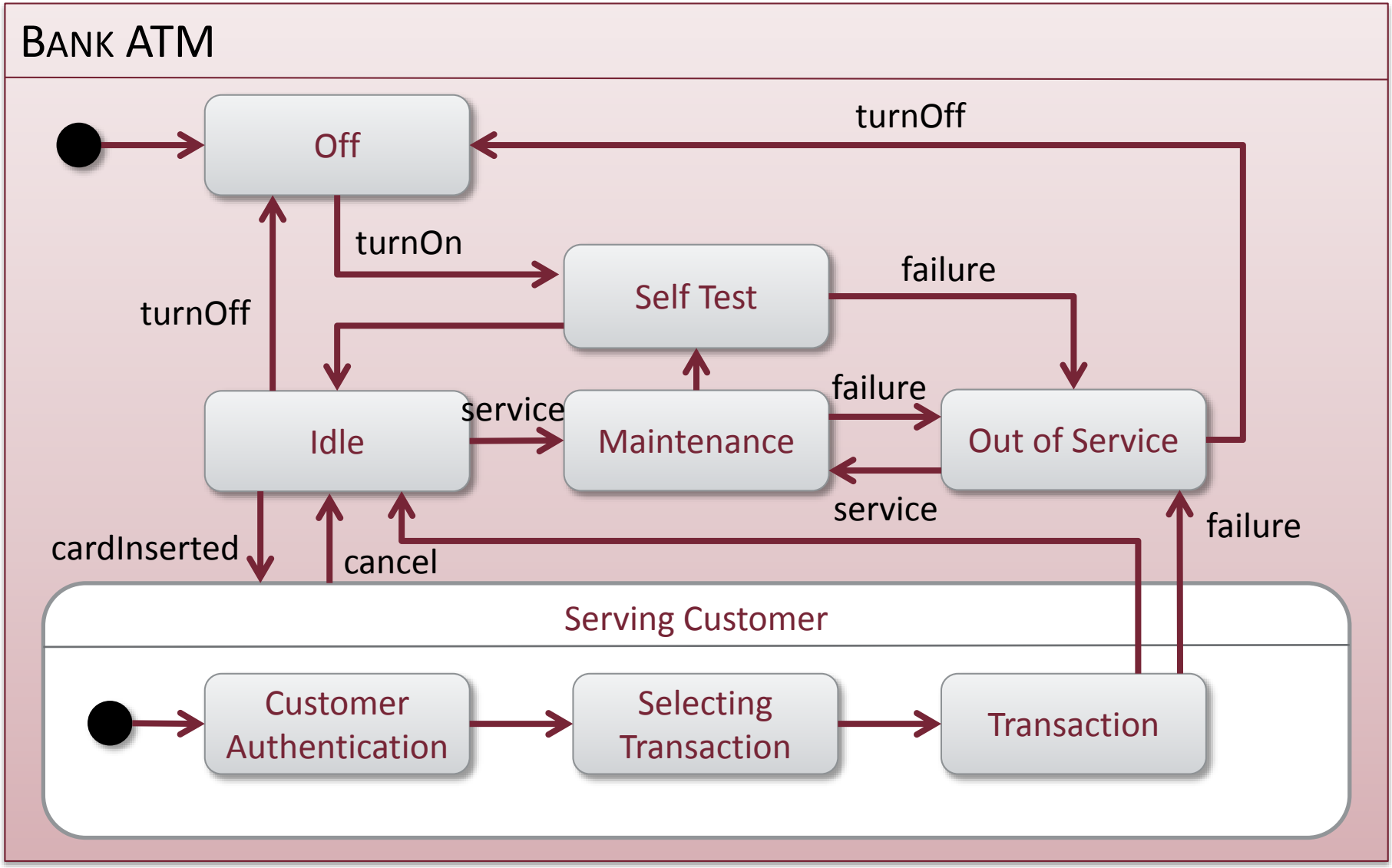
- Állapotkonfiguráció: $\{On, Reading\ operand1\}$

Állapothierarchia



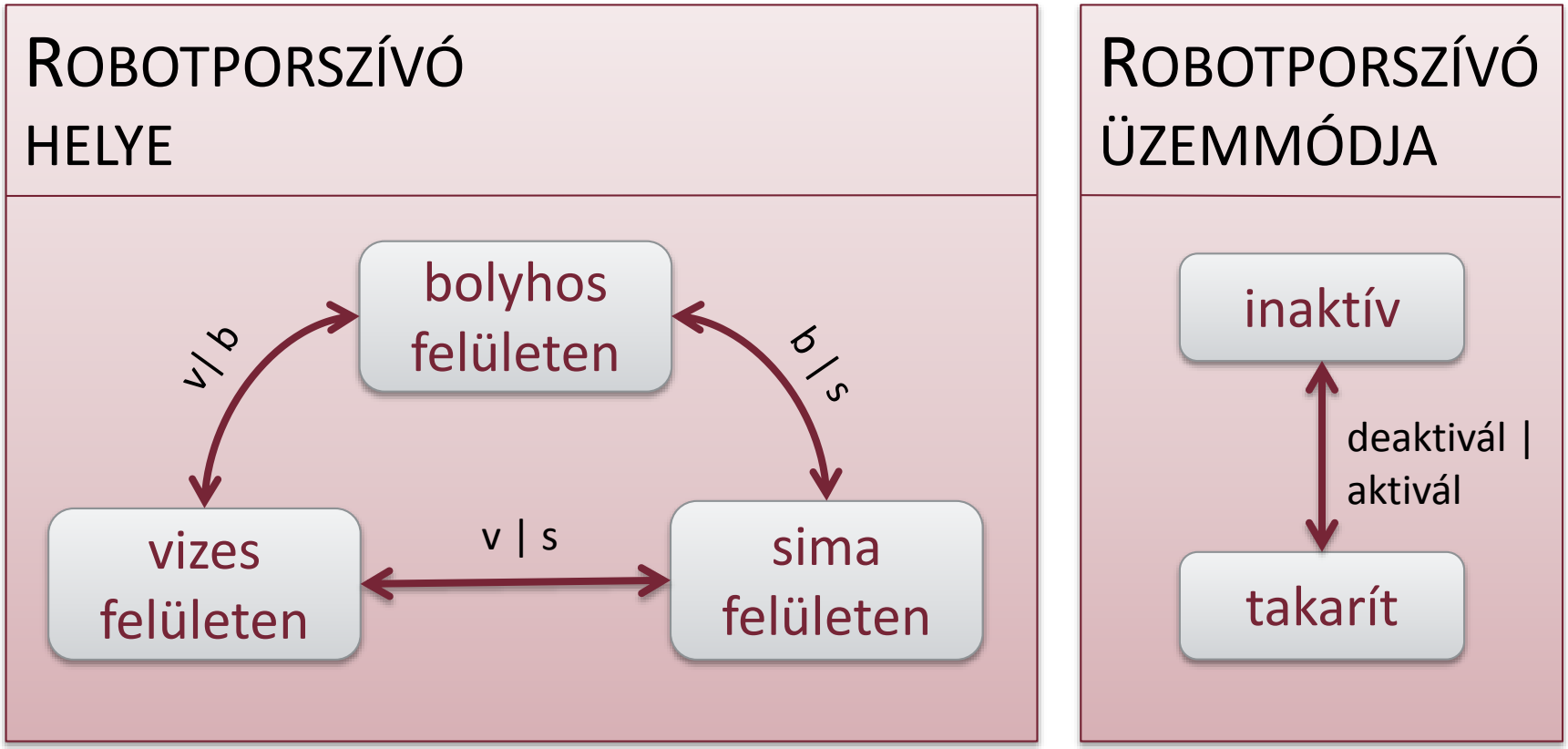
- Állapotkonfiguráció: $\{On, Reading\ operand2\}$

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

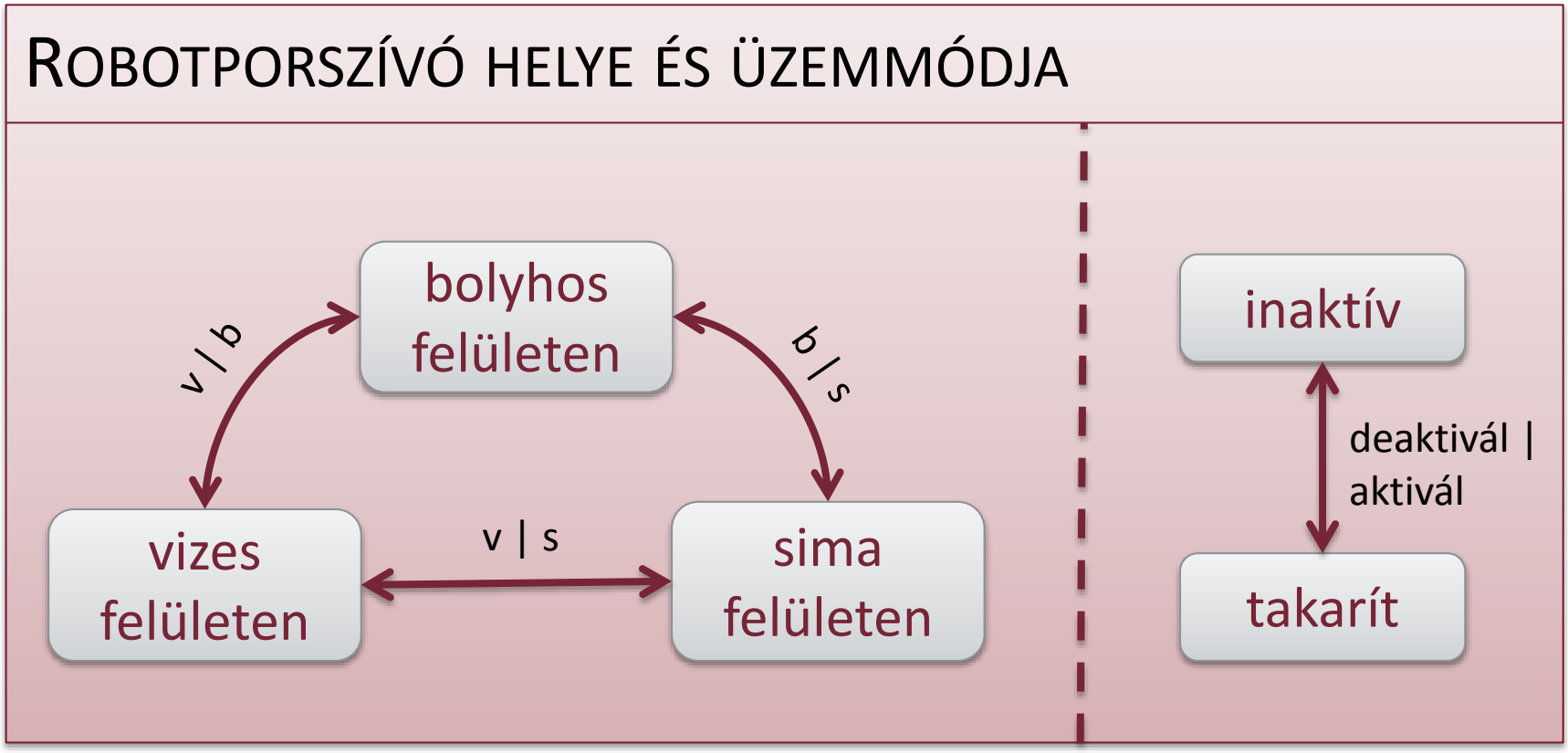
Példa: robotporszívó

Ortogonalitás

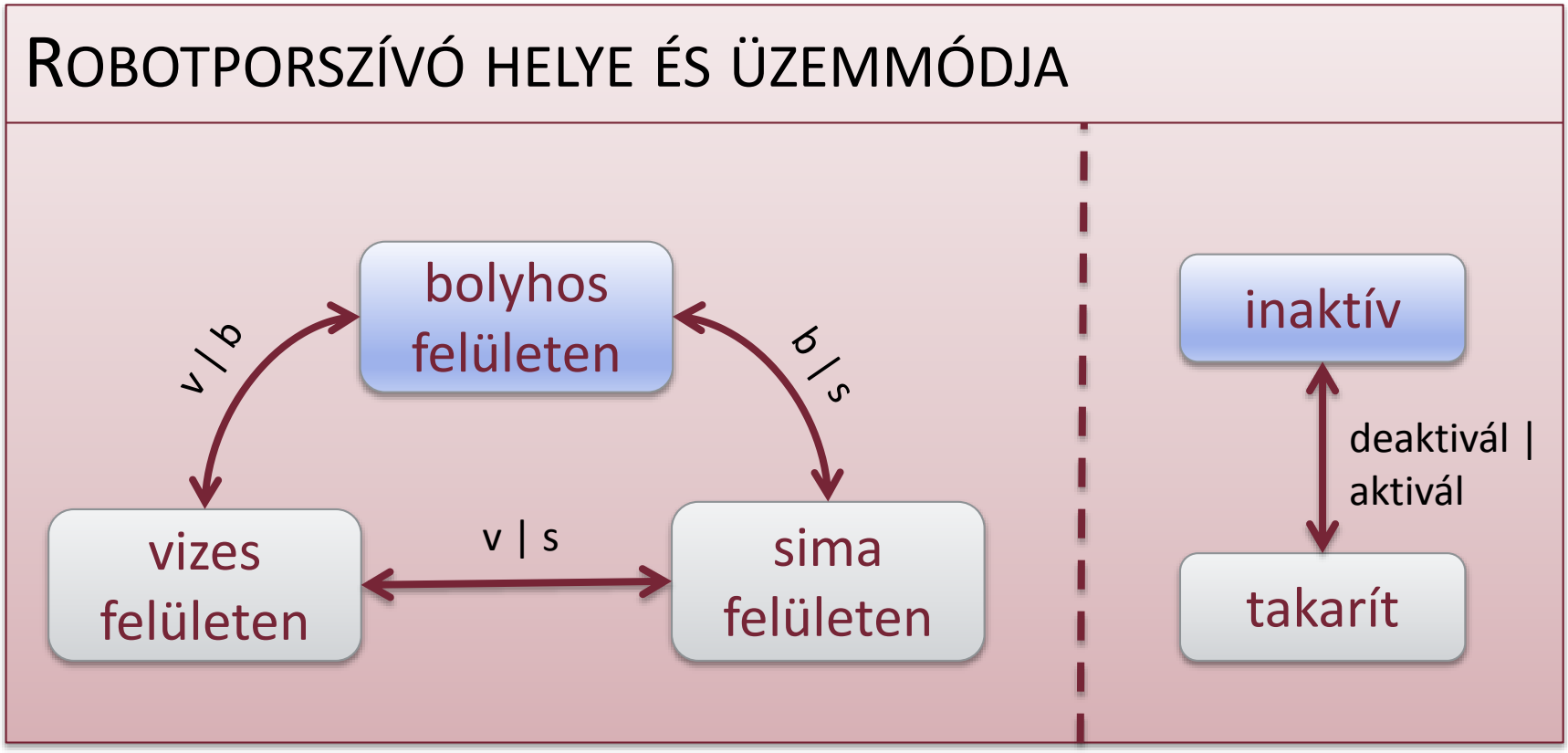


Megj: kétirányú nyíl nem szokásos, csak a tömör ábrázolás miatt használjuk…

Ortogonalitás

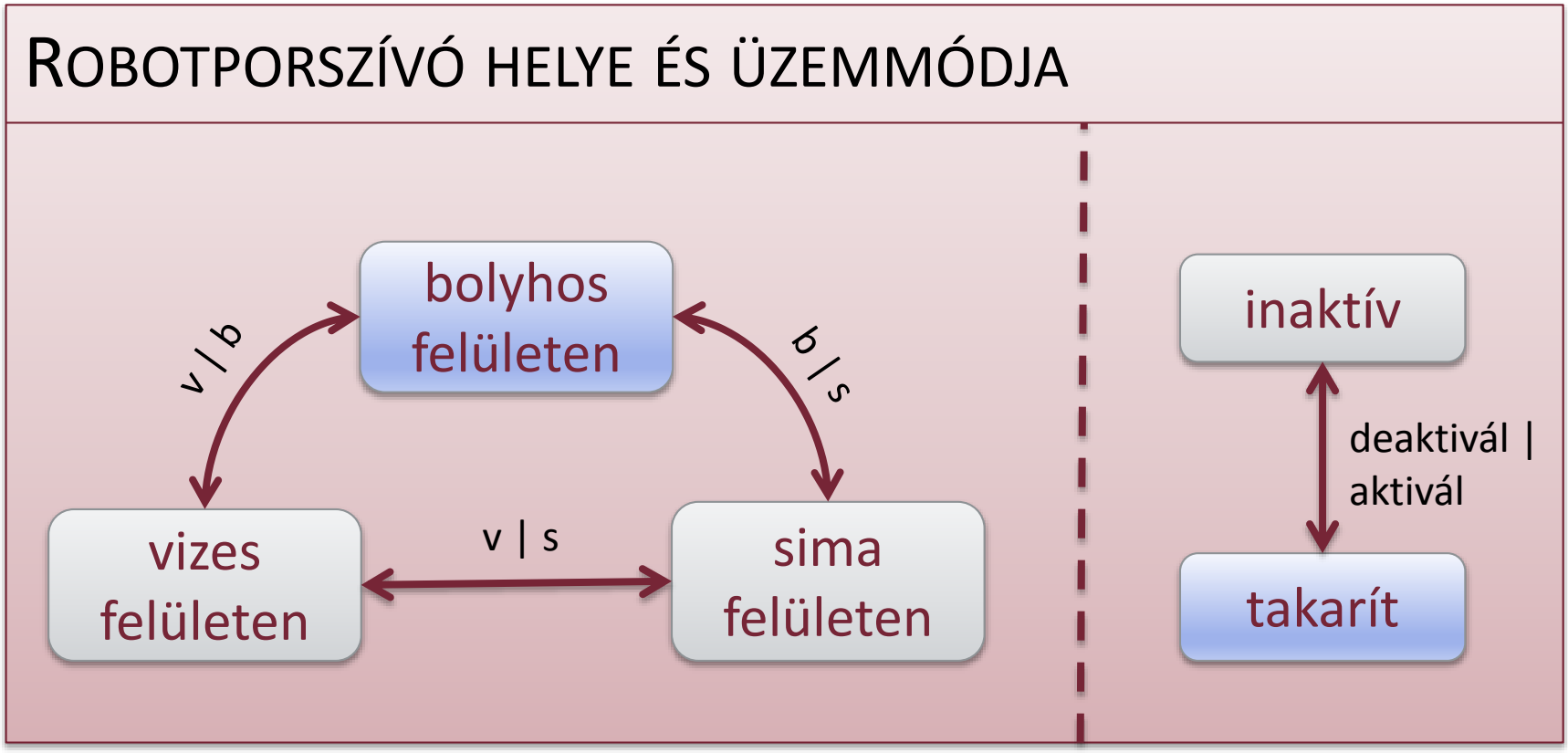


Ortogonalitás



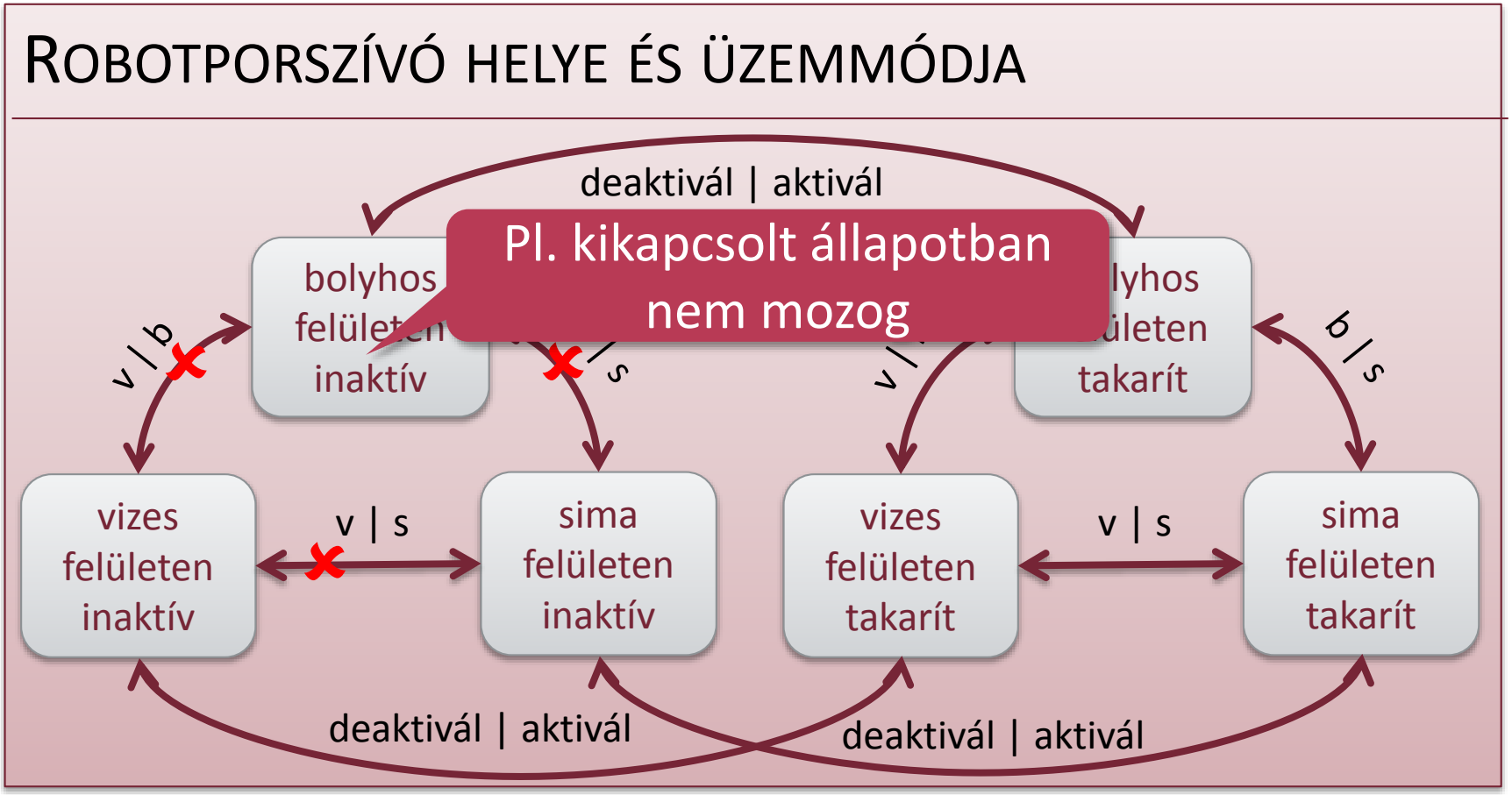
- Állapotkonfiguráció: {*bolyhos felületen*, *inaktív*}

Ortogonalitás



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

Aszinkron szorzat

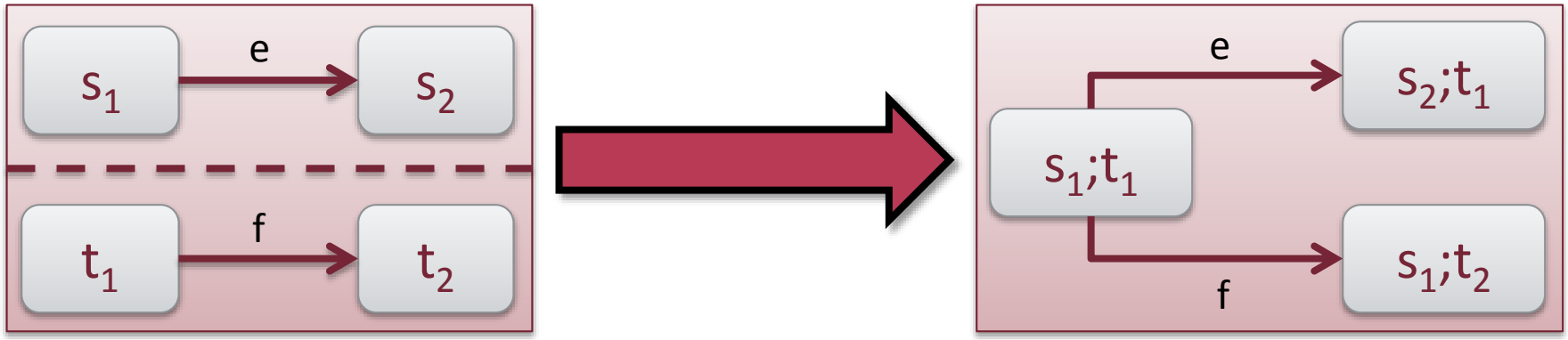


- További finomítást igényel: átmenetek kieshetnek
  → Állapotok így elérhetetlenné válhatnának

Aszinkron szorzat

- Állapotgépek szorzata
  - Modell felépítése állapotrégiók állapotgépeiből
  - Állapothalmaz: az állapotrégiók állapotainak szorzata
- Átmenetek
  - **Aszinkron szorzat**: egyszerre egy állapotrégió lép
    - Az átmenetek a szorzatba „másolódnak”

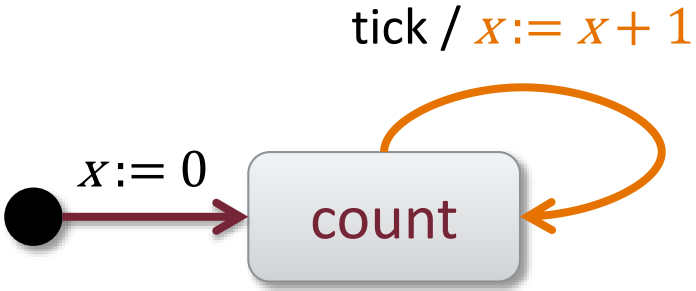
Könnyebb először kisebb modellekben gondolkozni!



Változók

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

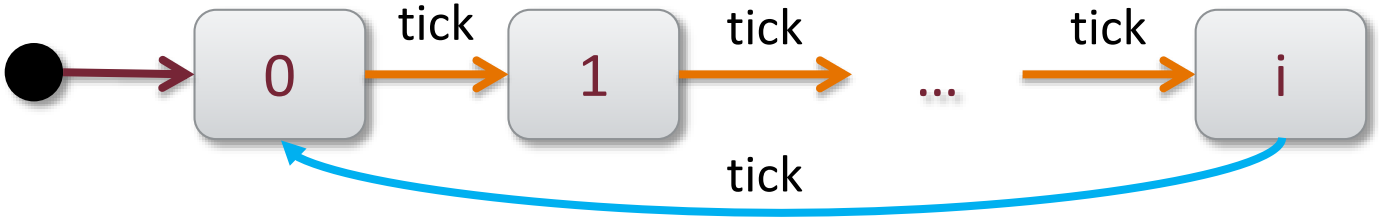
- Változó bevezetése: x



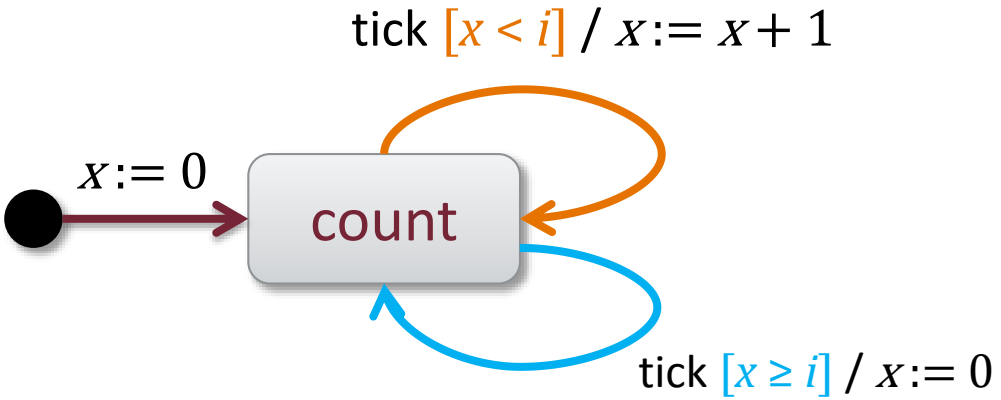
$$(count, \{x \mapsto 0\}) \rightarrow (count, \{x \mapsto 1\}) \rightarrow \ \ldots$$

Változók + őrfeltételek

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

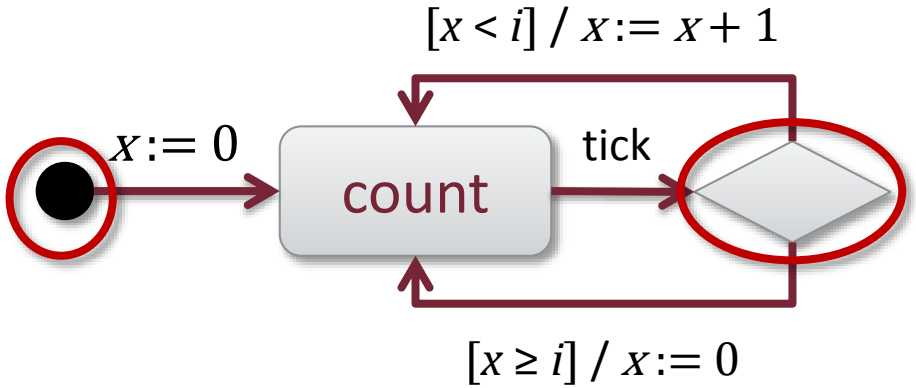


- Őrfeltételekkel:



Pszeudoállapotok

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

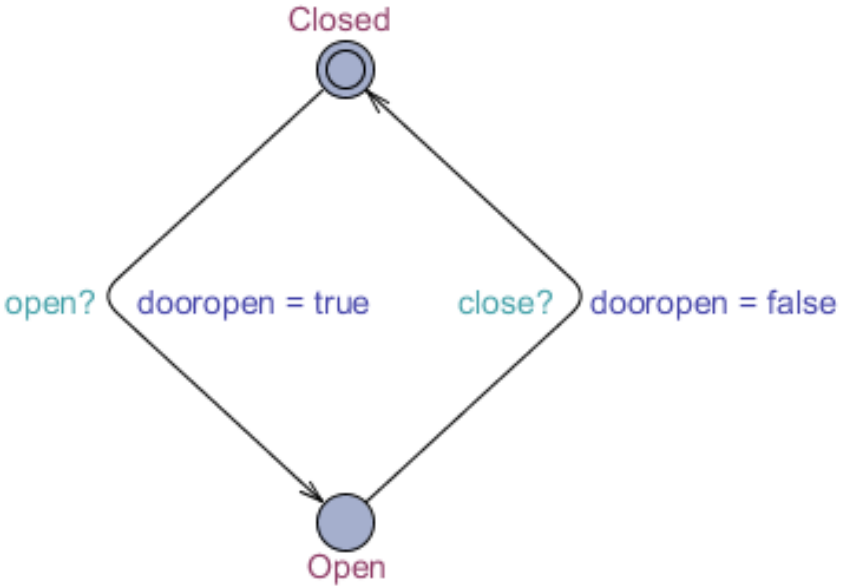


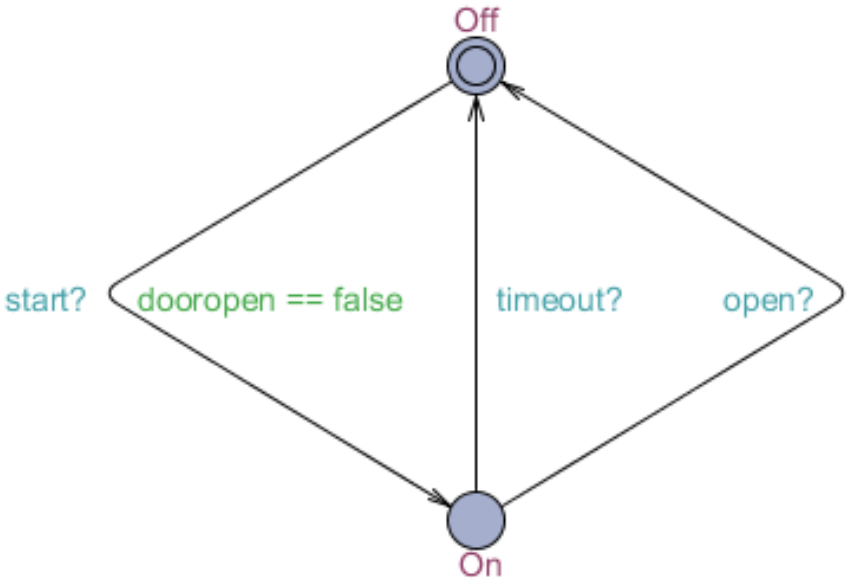
Tartalom

- Ismétlés, kitekintés
- Állapottér
- Állapotgépek
- Állapotgépek kiterjesztései
- **Szoftvertámogatás**

UPPAAL

- Állapotgépek vegyes szorzata



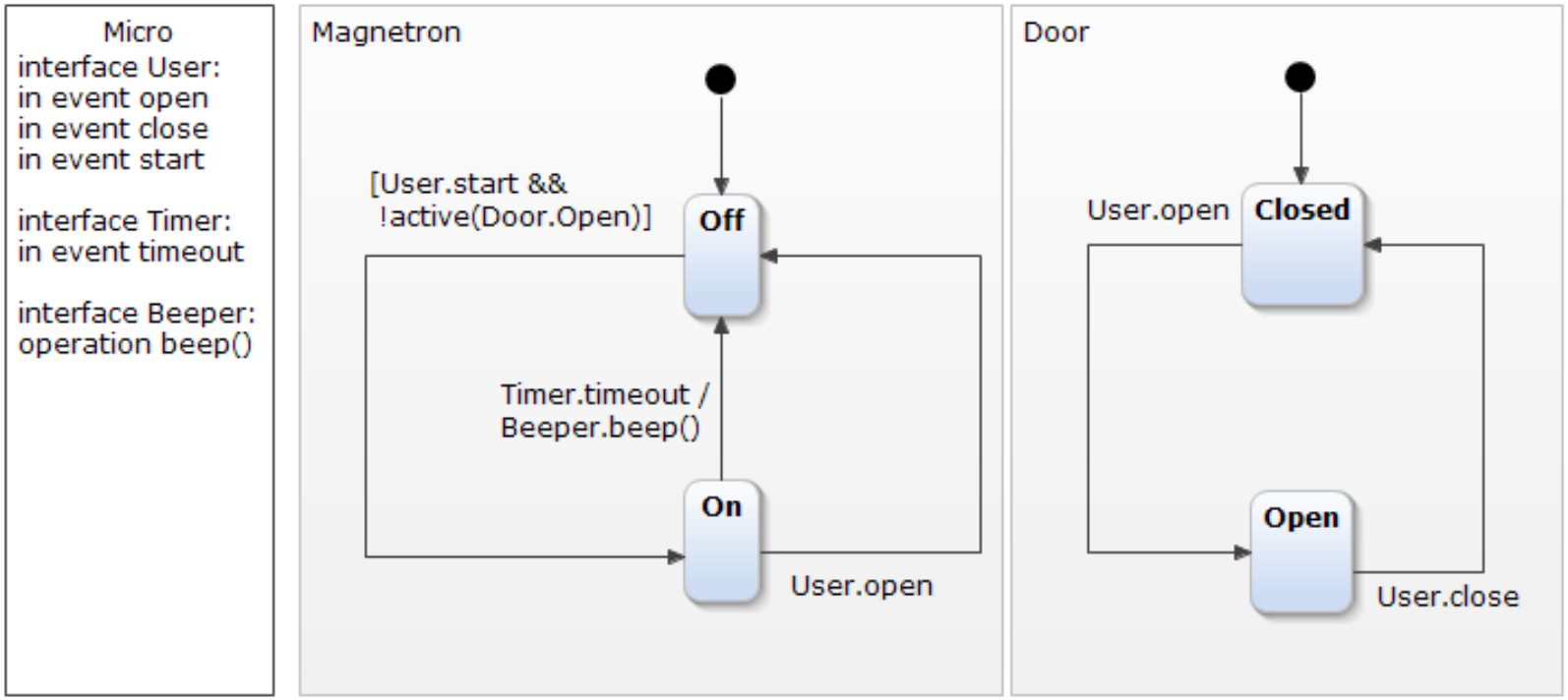


- Kompozit állapotgép viselkedése…
  - …szimulálható
  - …verifikálható

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

Yakindu Statechart Tools

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



Yakindu Statechart Tools

- Elkészült állapotgépből 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;
        }
    }
}
```

SZÓSZEDET

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