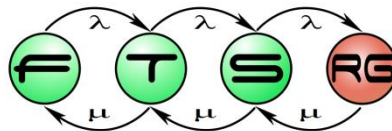


Állapot alapú modellezés

**Budapest University of Technology and Economics
Fault Tolerant Systems Research Group**



Tartalom

Ismétlés,
kitekintés

Állapotpartíció

Állapotgépek

Műveletek
állapotgépekkel

Szoftverdemó

Ismétlés,
kitekintés

Állapotpartíció

Állapotgépek

Műveletek
állapotgépekkel

Szoftverdemó

BEVEZETŐ

Ismétlés: felépítési vs. viselkedési

■ Felépítési (*structural*) modellek

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

Egy kosárban hat bájos kismacska van

A kismacskák, amikor ébren vannak, tudnak inni, nyávogni és karmolni

■ Viselkedési (*behavioral*) modellek

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

Egy kosárban hat bájos kismacska van, akik tudnak inni, nyávogni és karmolni

■ Nem fed le mindent, nem válik élesen szét...

Viselkedésmodellek fő kérdései

- Mit „csinál” a rendszer?



Esemény alapú modell
Folyamat alapú modell
...

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



Állapot alapú modell

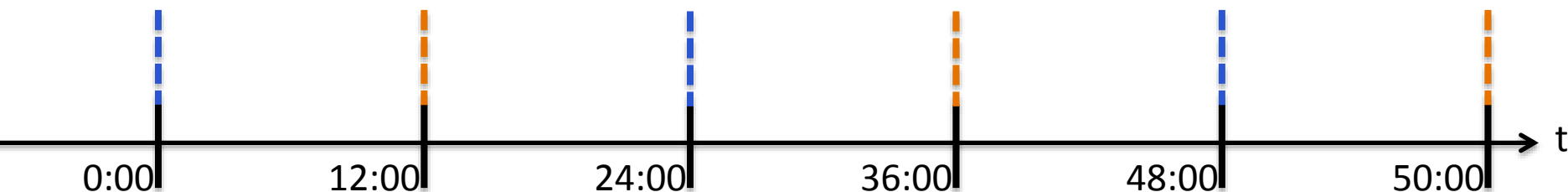
Alapozás: diszkrét eseménysor

■ Eseményfolyam

- Pl. input/output adatforrás
- **Eseménytér** –megengedett eseménytípusok halmaza
 - Beolvasható input értékek, kibocsátható output értékek
- Pillanatszerű események sora, egyszerre legfeljebb egy

Eseményfolyam: toronyóra ütései

Eseménytér: {éjfél, dél}



Ismétlés,
kitekintés

Állapotpartíció

Állapotgépek

Műveletek
állapotgépekkel

Szoftverdemó

AZ ÁLLAPOTPARTÍCIÓ

Kulcsfogalom: állapotpartíció

■ Állapotpartíció (más néven állapottér)

- Egymástól megkülönböztetett **rendszerállapotok** köre
- Példák
 - Pl. {hétfő, kedd, szerda, csütörtök, péntek, szombat, vasárnap}
 - Pl. kismacska állapotai: {alszik, éber}
 - Pl. mikró állapotai: {teljes fokozat, kiolvasztó mód, kikapcsolva}
- **DEF:** Az állapotpartíció egy halmaz, amelynek minden időpontban pontosan egy eleme jellemzi a rendszert.

■ Pillanatnyi állapot ???

- Pl. most szerda van, a kismacska alvó állapotban van
- **DEF:** Egy adott időpontban a rendszer pillanatnyi állapota az állapotpartíció azon egyetlen eleme, amelyik abban az időpontban jellemző a rendszerre.

Az állapotpartíció tulajdonságai

■ Teljesség

- Mindig legalább az egyik állapot fennáll
- Ellenpéldák (nem jó állapotpartíciók!)
 - {hétfő, kedd, csütörtök, szombat} nem teljes
 - kismacskára {alszik, iszik}

■ Kölcsönös **kizárólagosság**

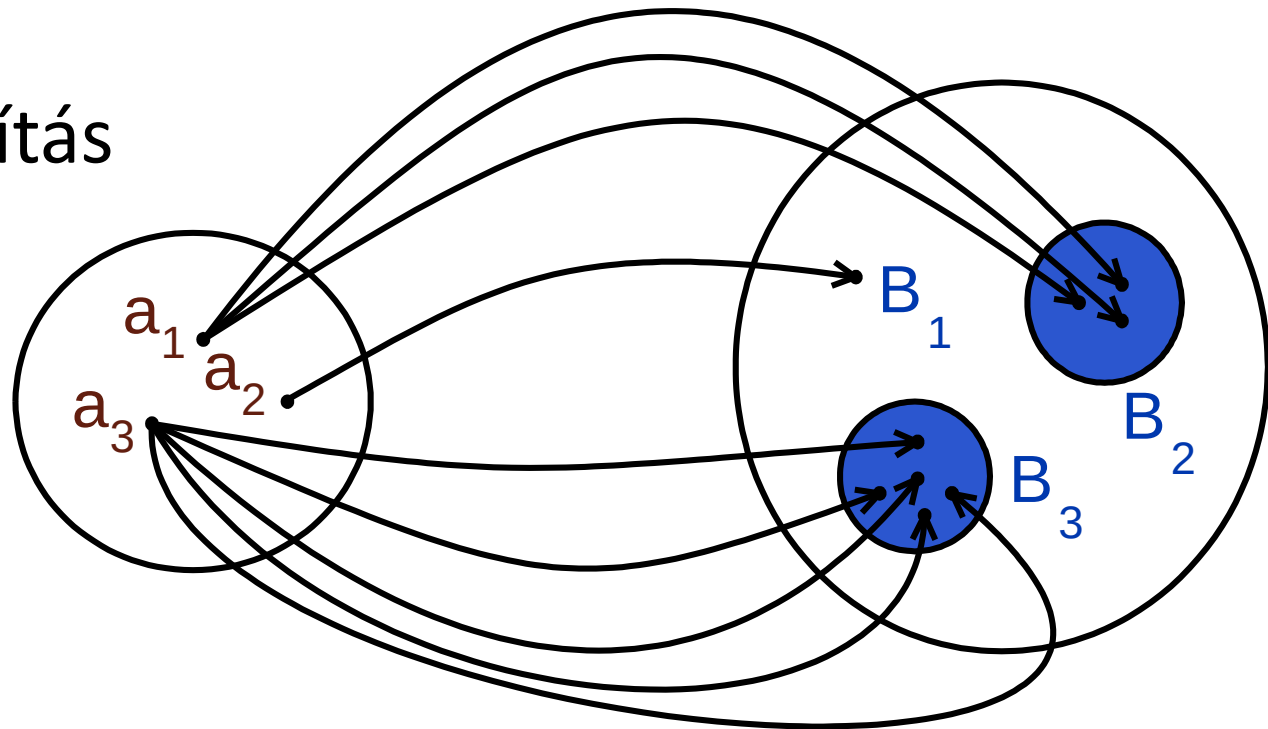
- Egyszerre csak egy állapot állhat fent
- Ellenpéldák (nem alkalmasak állapotpartíciónak!)
 - {hétköznapi, hétvége, délután} nem kizárólagos (holott teljes)
 - mikróra {ajtaja tárva, kikapcsolva}

■ Állapot alapú modellezésnél ezt tartsuk észben!

Állapotfinomítás, állapotabsztrakció

- Állapotfinomítás/absztrakció: az állapotpartícióon végzett halmazfinomítás/absztrakció

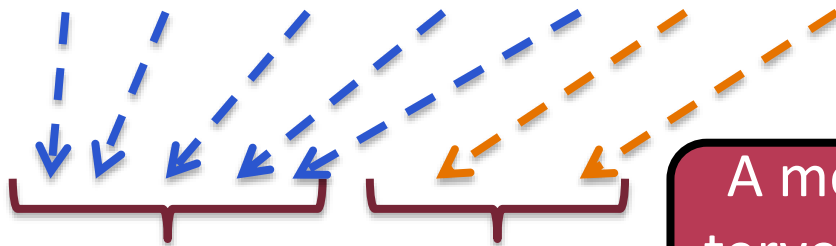
Ismétlés:
halmazfinomítás



Állapotfinomítás, állapotabsztrakció

- Állapotfinomítás/absztrakció: az állapotpartícióon végzett halmazfinomítás/absztrakció

- {Hé, Ke, Sze, Csü, Pé, Szo, Va}



- {hétköznap, hétvége}

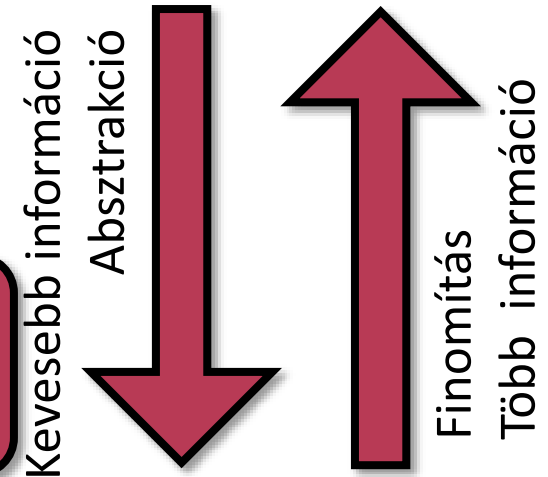
A menetrend tervezőjét csak ennyi érdekli

- Mikro állapotabsztrakciója

- {teljes, olvasztó, kikapcsolva} → {bekapcsolva, kikapcsolva}
- „kikapcsolva” helybenhagyva

- Kismacska állapotfinomítása

- {alszik, éber} → {alszik, eszik, iszik, játszik, felfedez, pihen}



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)
 - Pl.: a kosárban két kismacska van (Lukrécia, Szerénke)

■ Több információ, több tudás

- Ez vajon mindig jó?

Állapotfinomítás, állapotabsztrakció

- Állapotabsztrakció: miért?
 - Akkor hasznos, ha az absztrakt állapotok „egységesek”
 - 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 állapotpartíció
- Pl.: a mikró két külön állapotpartíciója
 - üzemteljesítmény szerint
 - {teljes fokozat, olvasztó mód, kikapcsolva}
 - az ajtó nyitottsága szerint
 - {nyitva, zárva}
 - nem teljesen függetlenek: ha nyitva, akkor kikapcsolva
- Pl.: a kosárban két kismacska (Lukrécia, Szerénke)
 - {L. alszik, L. ébren van} ill. {Sz. alszik, Sz. ébren van}
 - részrendszer állapottere → a teljes rendszerállapot ismerete nélkül eldönthető, melyik áll fenn

Állapotterek (direkt) szorzata

■ Két állapotpartíció együttes figyelembevétel

- $S_1 = \{\text{teljes fokozat, olvasztó mód, kikapcsolva}\}$
 - $S_2 = \{\text{nyitva, zárva}\}$
- } állapot-változók

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

Állapotváltozók finomított kompozíciója

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

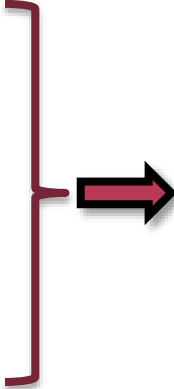


Állapotváltozók finomított kompozíciója

- „A szorzatnál ***finomabb*** kompozit állapottér”
 - ... mivel két kompozit állapot előfordulását kizártuk
 - Itt a finomított állapotpartíciónak 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
$$S_1 = \{teljes, \underline{olvasztó}, kikapcsolva\}$$
$$S_2 = \{nyitva, zárva\}$$
- A vetített állapotváltozók absztrakciók
- Miért dekomponálunk?
 - Állapotváltozók külön tárolhatóak
 - Absztrakció általános előnyei

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:

- szoftver memóriájában
- Adatbázisban (hosszabb távra)

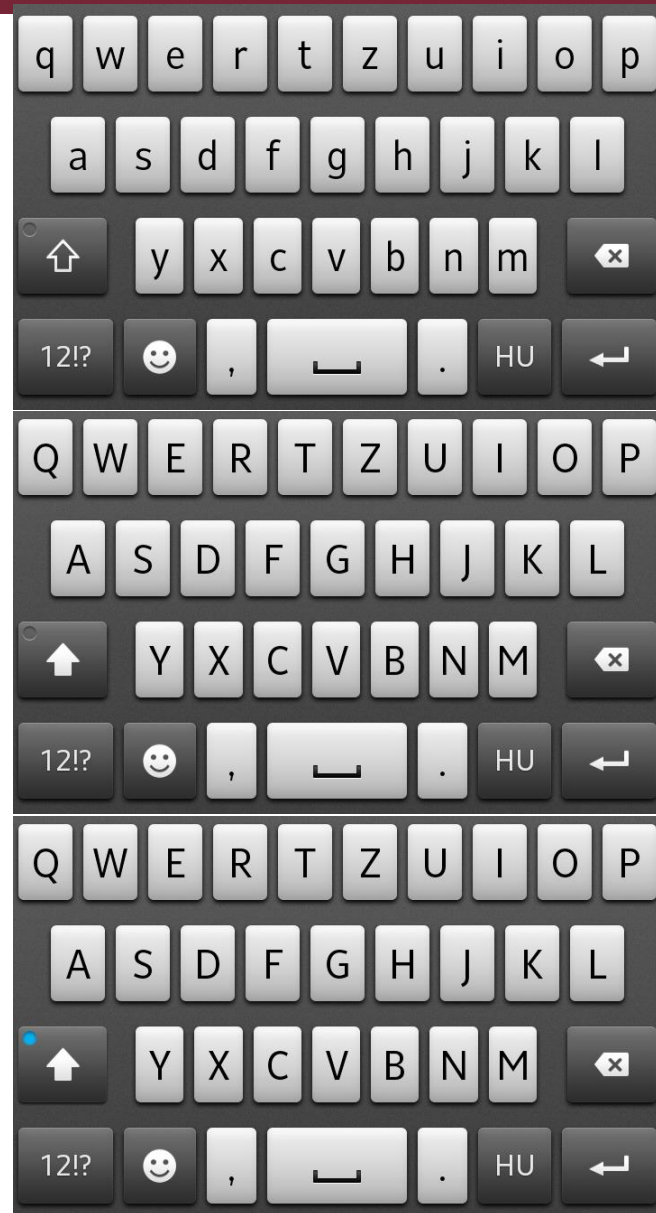
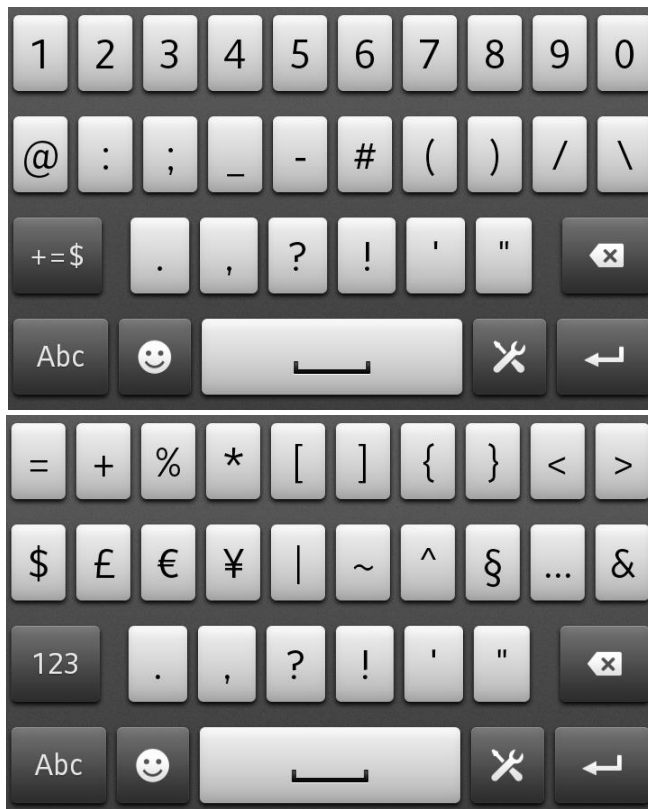
$S_1 \times S_2$		
	nem ismerősök	Jancsi bejelölte Juliskát
	Juliska bejelölte Jancsit	ismerősök

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
 - 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
 - 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;
```


Ismétlés,
kitekintés

Állapotpartíció

Állapotgépek

Műveletek
állapotgépekkel

Szoftverdemó

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

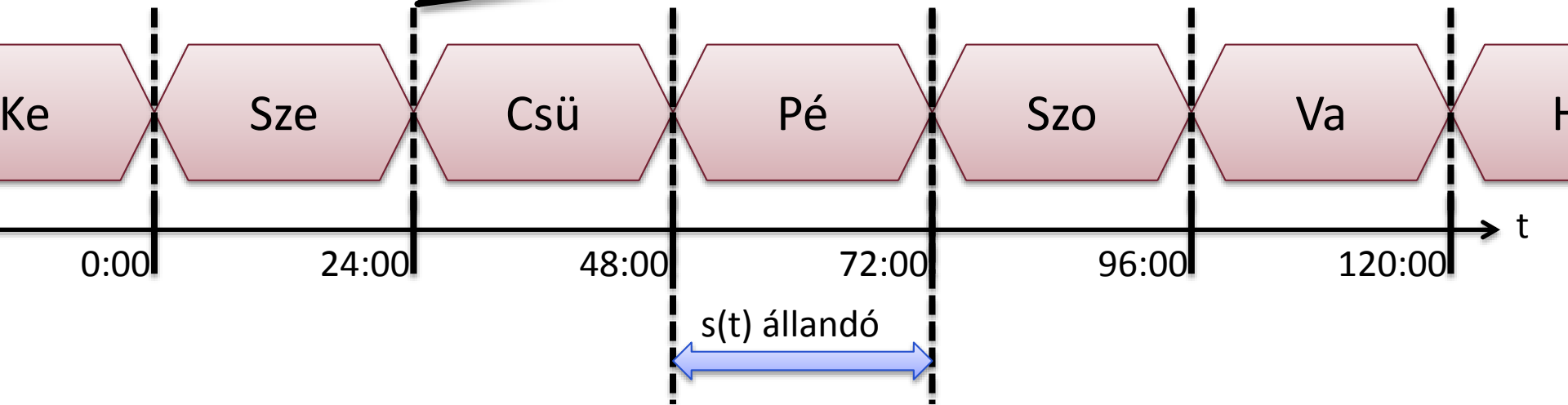
- Állapottér: S

- Pl. $S = \{\text{Hé}, \text{Ke}, \text{Sze}, \text{Csü}, \text{Pé}, \text{Szo}, \text{Va}\}$

- $s(t) \in S$

- A pillanatnyi állapot az idő függvényeként

Pillanatszerű állapotátmenet
(esemény)



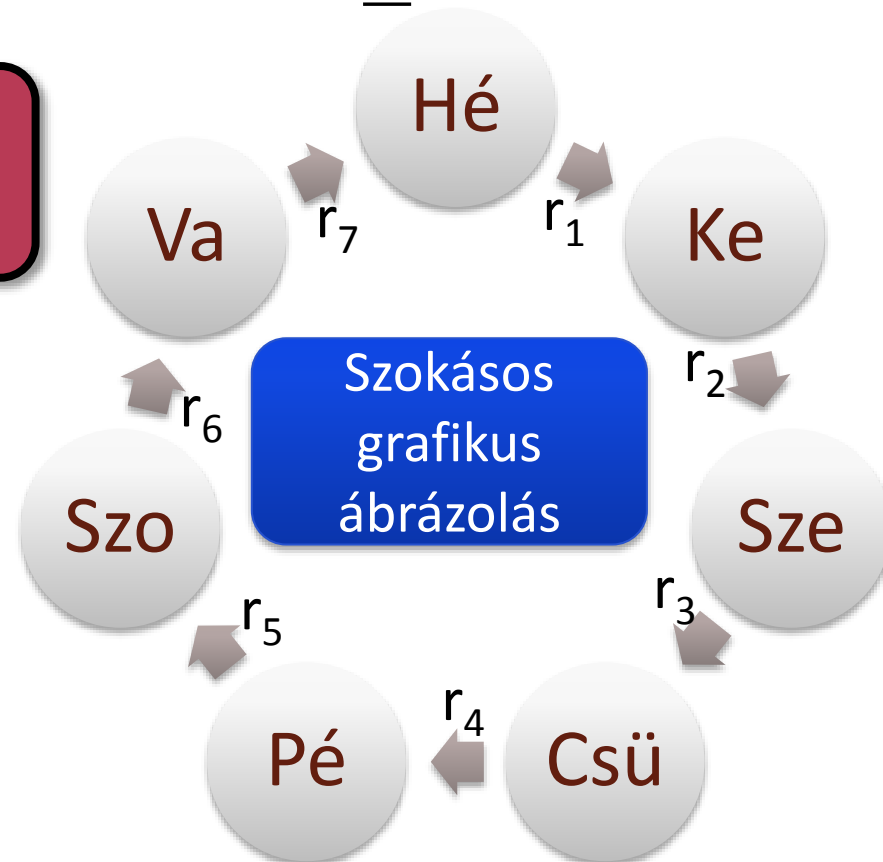
Állapotátmenetek

■ Mely állapotokat mely állapotok követhetnek?

- Állapottér $S = \{\text{Hé, Ke, Sze, Csü, Pé, Szo, Va}\}$
- Eseménytér: **állapotátmeneti reláció** $R \subseteq S \times S$

- Állapotátmeneti szabályok $r \in R$
- $r_1 = \langle \text{Hé, Ke} \rangle$
 - $r_2 = \langle \text{Ke, Sze} \rangle$
 - $r_3 = \langle \text{Sze, Csü} \rangle$
 - $r_4 = \langle \text{Csü, Pé} \rangle$
 - $r_5 = \langle \text{Pé, Szo} \rangle$
 - $r_6 = \langle \text{Szo, Va} \rangle$
 - $r_7 = \langle \text{Va, Hé} \rangle$

Csü-ből Pé-be
át tud lépni
a rendszer



- Más néven: **állapotgráf**

Észrevételek az állapotgráfról

■ Lehetséges, hogy...

- ... teljes gráf $\rightarrow$ minden átmenet engedélyezett
 - Pl. $S_{\text{kis macska}} = \{\text{alszik, játszik, iszik}\}$
- ... nem minden állapotból érhető el az összes többi
 - Pl. $S_{\text{pohár}} = \{\text{üres, teli, törött}\} \rightarrow$ nincs törött $\leadsto$ 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

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)



- ... az állapotátmenet oka (prekondíció)



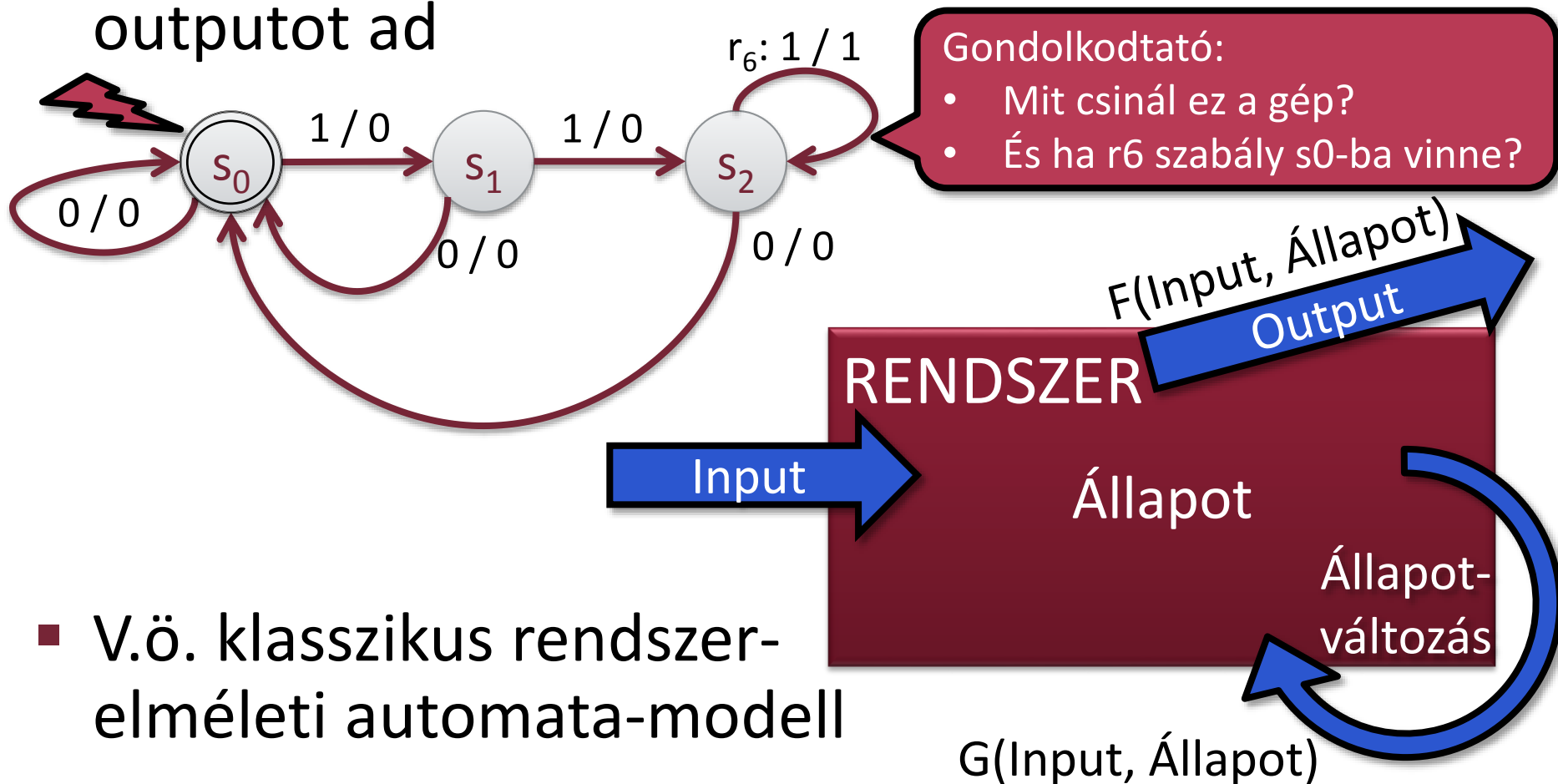
- Egy szabály címkézhető több eseménnyel is

- input beolvasás / output kiírás



Emlékeztető: Mealy véges automata

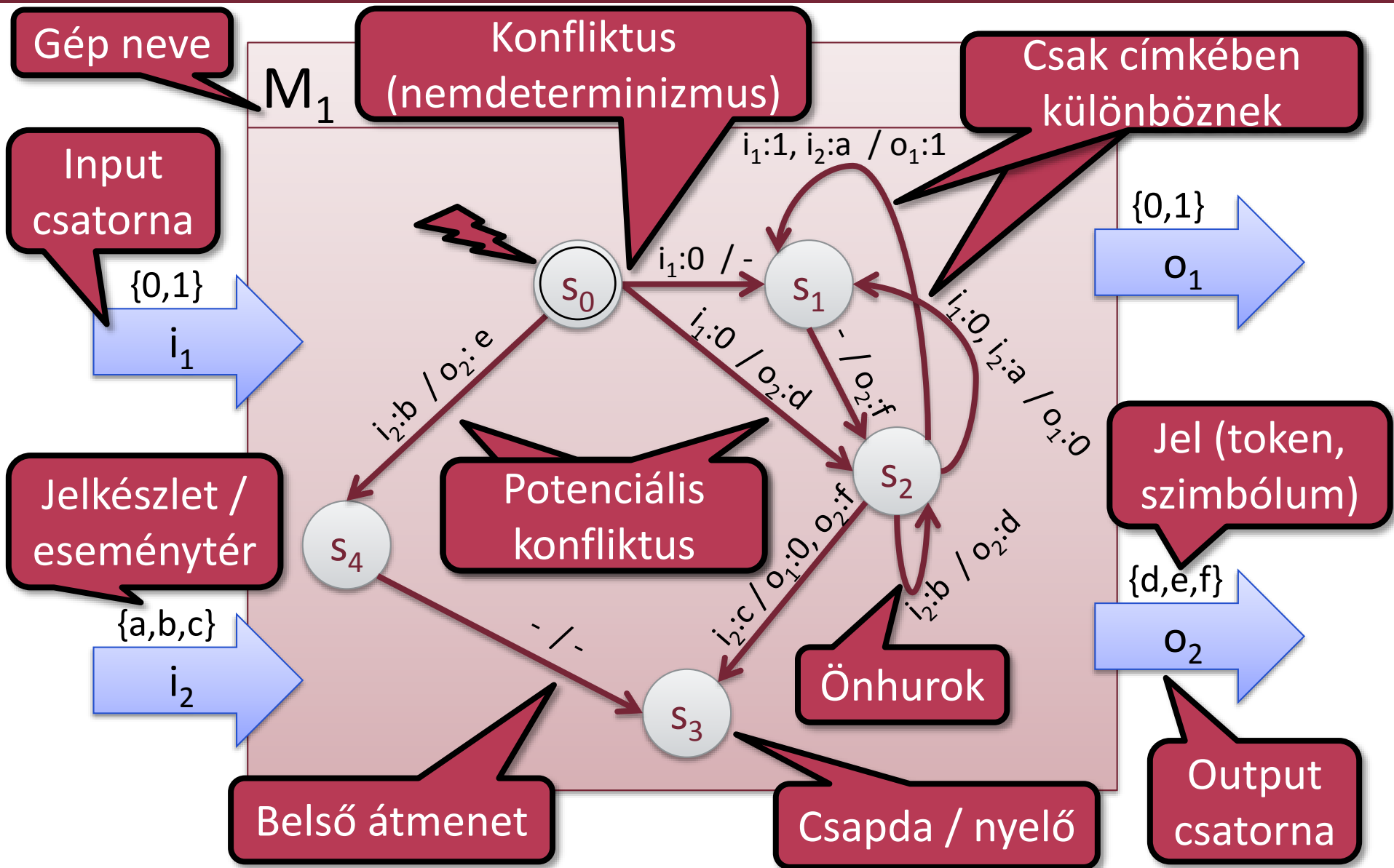
- Kijelölt kezdőállapot $\rightarrow s_0 = s(t=0)$
- Minden lépés determinisztikus, inputot olvas és outputot ad



A Mealy gép kiterjesztései

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

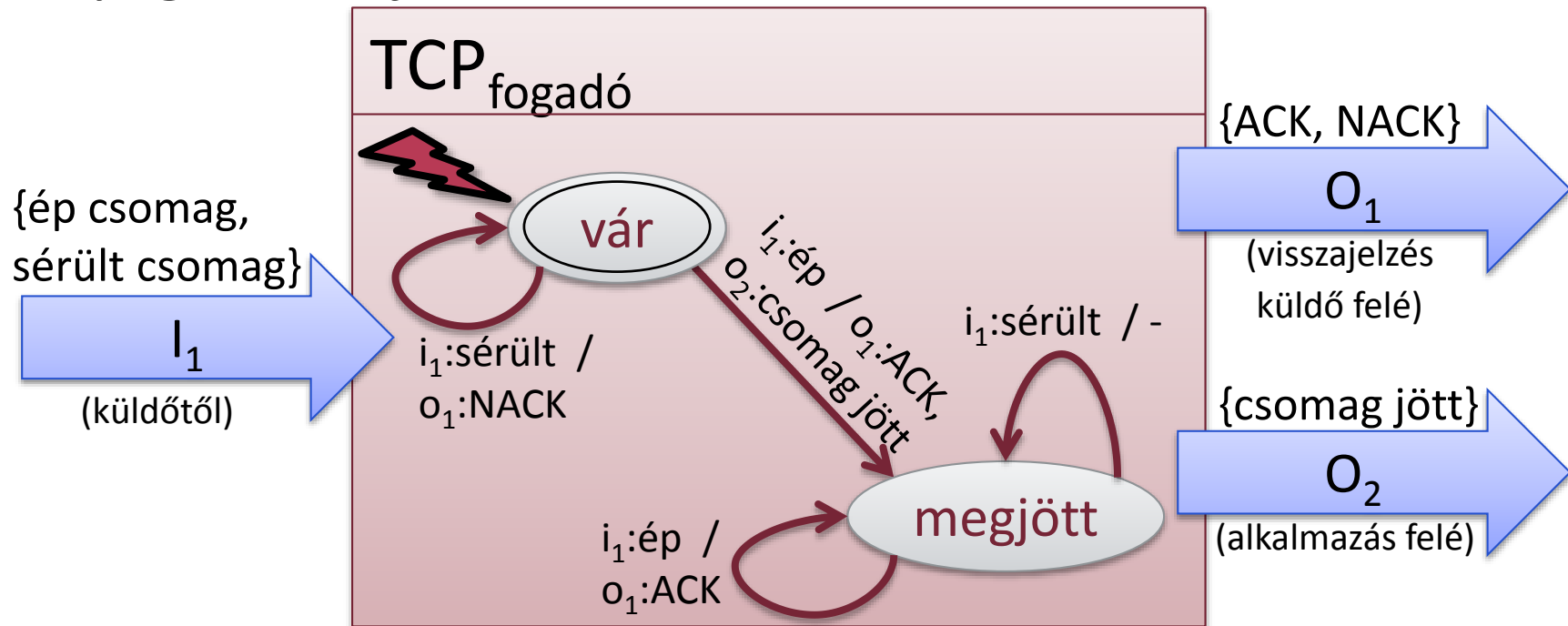


Specifikálatlan input

- 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
 - Az is érvénytelen, ha több szabály van (determinizmus kell)

Kitekintés: szakmai példák

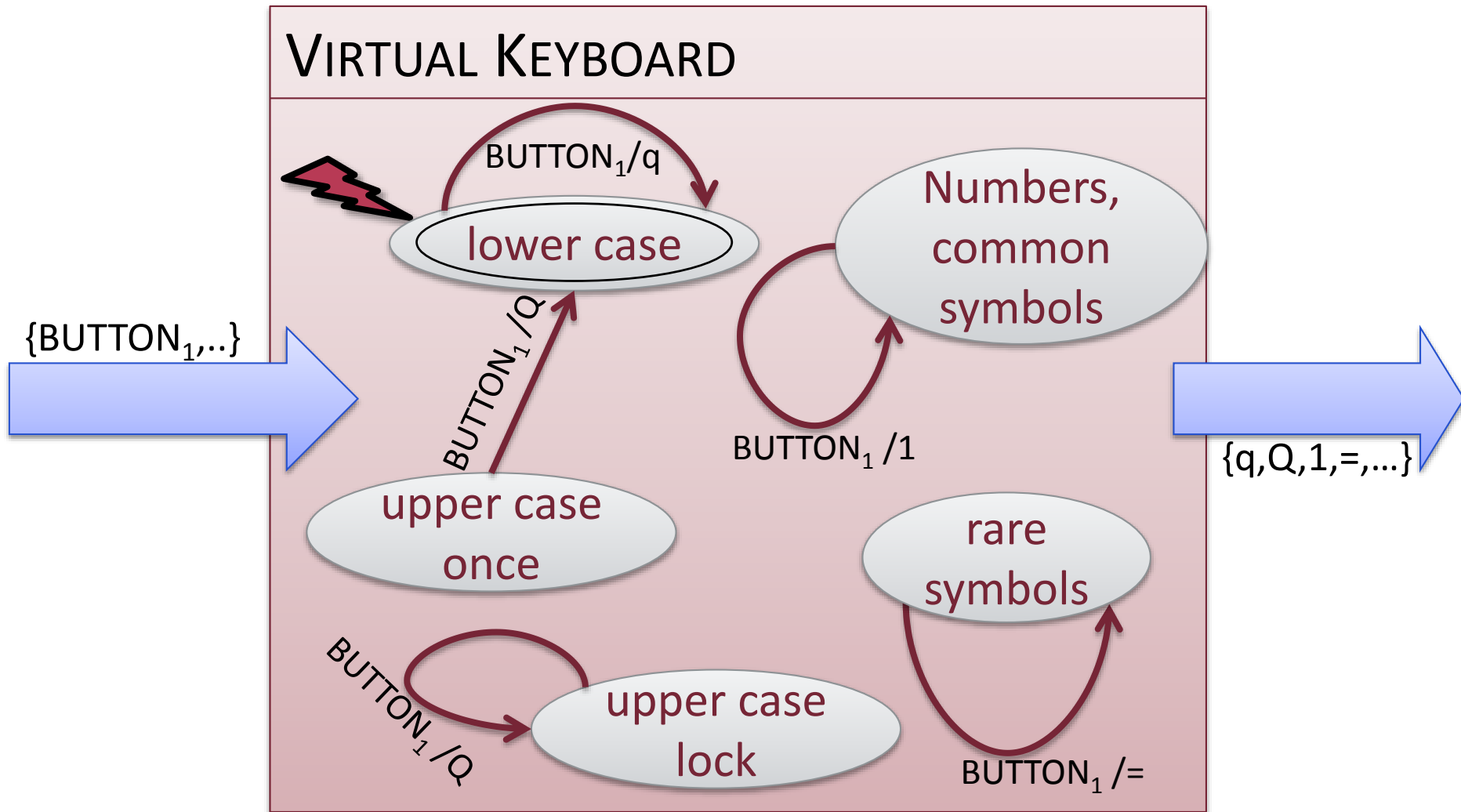
- Csomag alapú adatátviteli protokoll
 - Csomagok elveszhetnek, megsérülhetnek
 - Nyugtázás, újraküldés



- (Bővebben ld. Számítógép-hálózatok c. tárgy)

Kitekintés: szakmai példák

- Virtuális billentyűzet



Kitekintés: szakmai példák

■ Programozás: állapotgép megvalósítása

- 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

- Programozás: állapotgép megvalósítása
 - Táblázatos megoldás
 - Tömörebb, de rugalmatlanabb

```
Character [][] output = {
    {'q', 'Q', 'Q', '1', '='},
    { /* ... */ }
    // ...
};
VirtualKeyboardState [][] jumpTable = {
    // ...
};
void handleKey(KeyCode input) {
    emit(outputTable[input.ordinal()][keyboardState.ordinal()]);
    keyboardState = jumpTable[input.ordinal()][keyboardState.ordinal()];
}
```

Kitekintés: szakmai példák

■ Programozás: állapotgép megvalósítása

○ Kompozit állapottér esetén

Állapottér-robbanás

- Többdimenziós táblázat

- Több állapotváltozó együttes olvasása

```
if (  
    keyboardFacet == VirtualKeyboardFacet.ALPHABETIC &&  
    capsState      == CapsState.LOWER_CASE  
) {  
    emit('q');  
}
```

- Sokat egyszerűsít, ha bizonyos kódrészletek csak bizonyos állapotváltozókat használnak (esetleg csak egyet!)
 - Pl. szám/szimbólum módban a Caps Lock állás nem érdekes
 - Ez ilyenkor egy jól sikerült dekompozíció

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

Ismétlés,
kitekintés

Állapotpartíció

Állapotgépek

Műveletek
állapotgépekkel

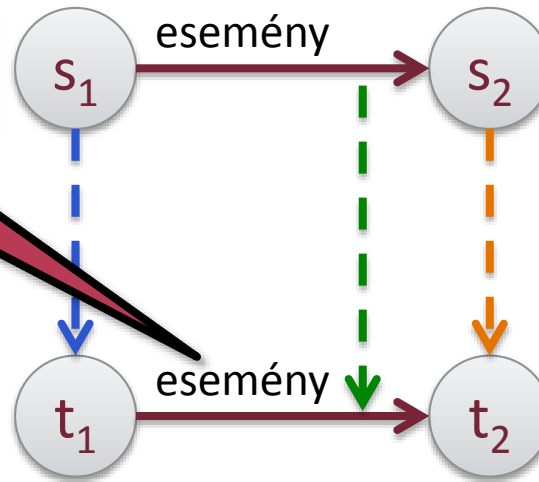
Szoftverdemó

MŰVELETEK ÁLLAPOTGÉPEKKEL

Állapotátmenetek vs. állapotabsztrakció

■ Állapotabsztrakció hatása átmenetre

Ha van címke, megmarad

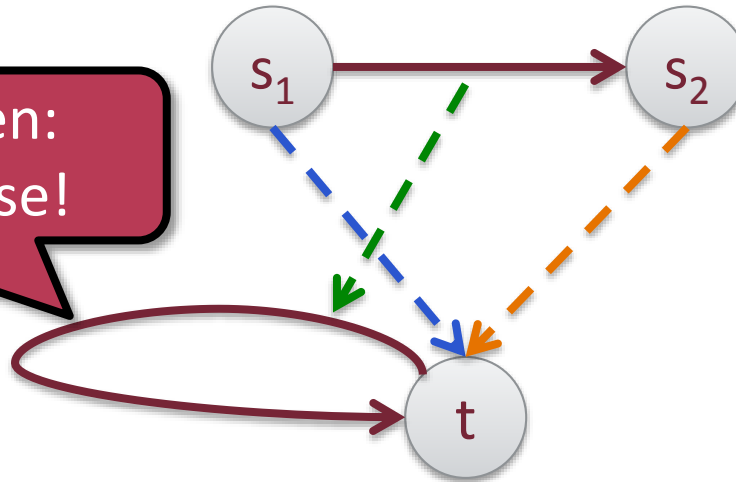


Kevesebb információ
Absztrakció

A large red arrow pointing downwards, indicating the direction of abstraction from a more detailed state to a more abstract one.

■ Speciális eset

Címkézetlen esetben:
ennek nincs jelentése!

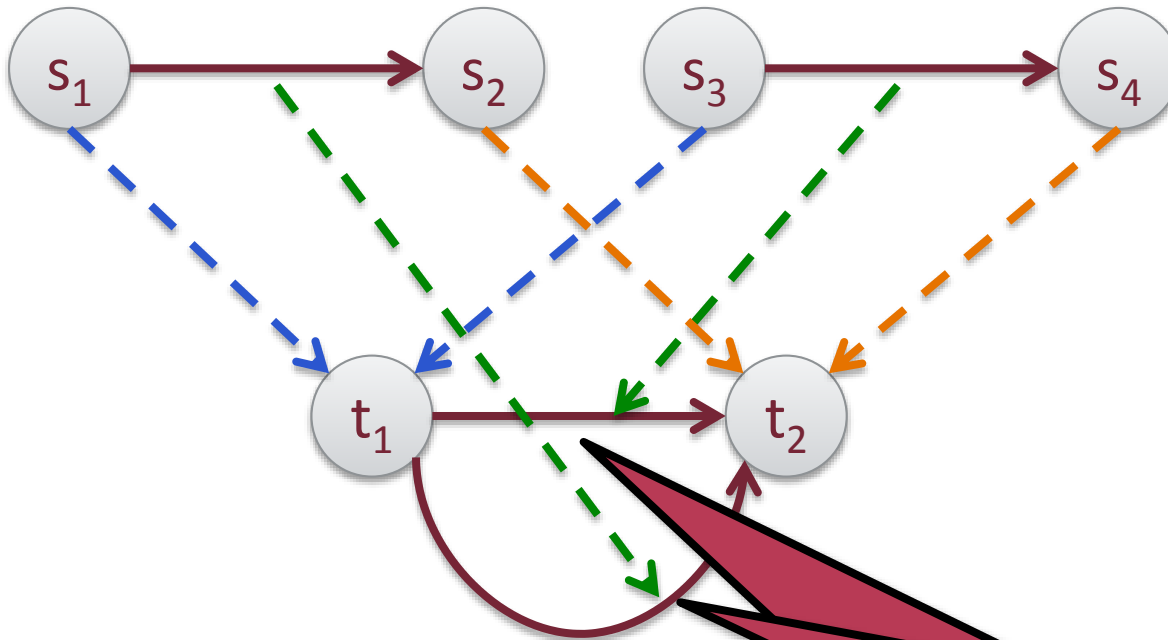


Kevesebb információ
Absztrakció

A large red arrow pointing downwards, indicating the direction of abstraction from a more detailed state to a more abstract one.

Állapotátmenetek vs. állapotabsztrakció

- Állapotabsztrakció hatása átmenetre

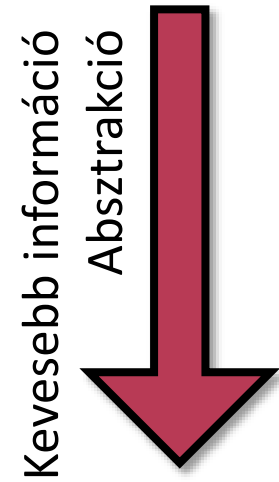
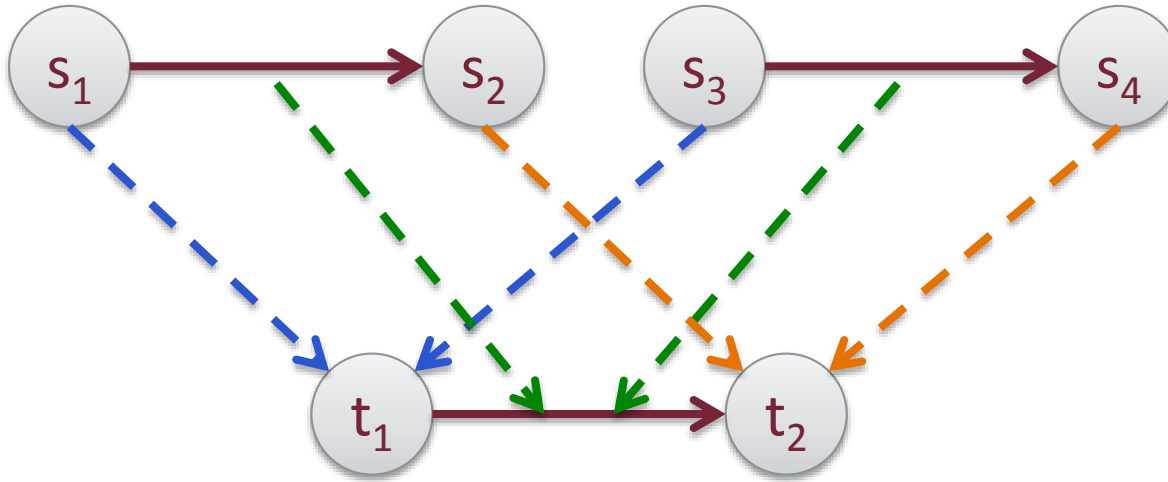


Kevesebb információ
Absztrakció

Ha azonos a címke (vagy nincs),
akkor ez valójában egy szabály

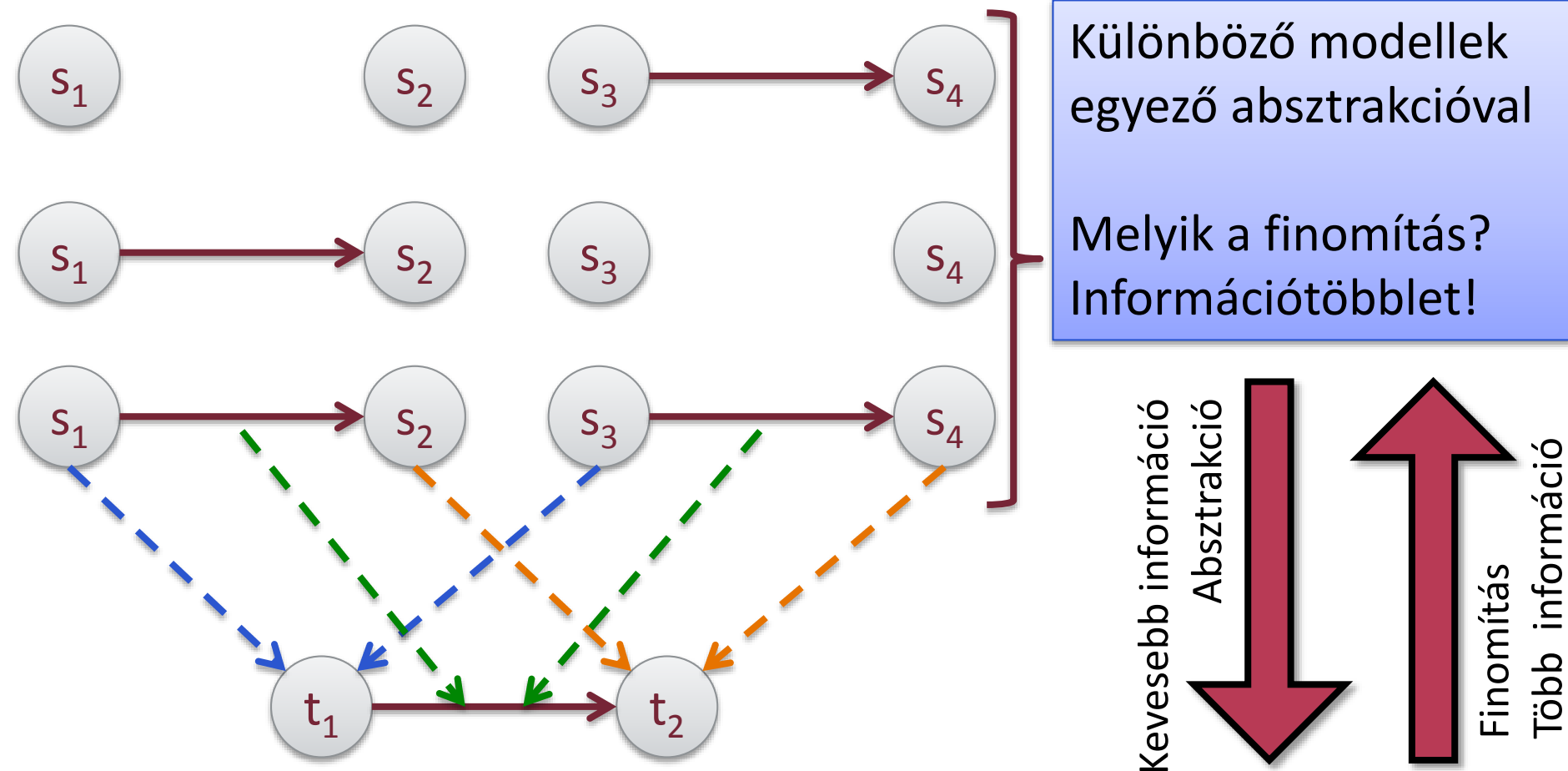
Állapotátmenetek vs. állapotabsztrakció

- Állapotabsztrakció hatása átmenetre

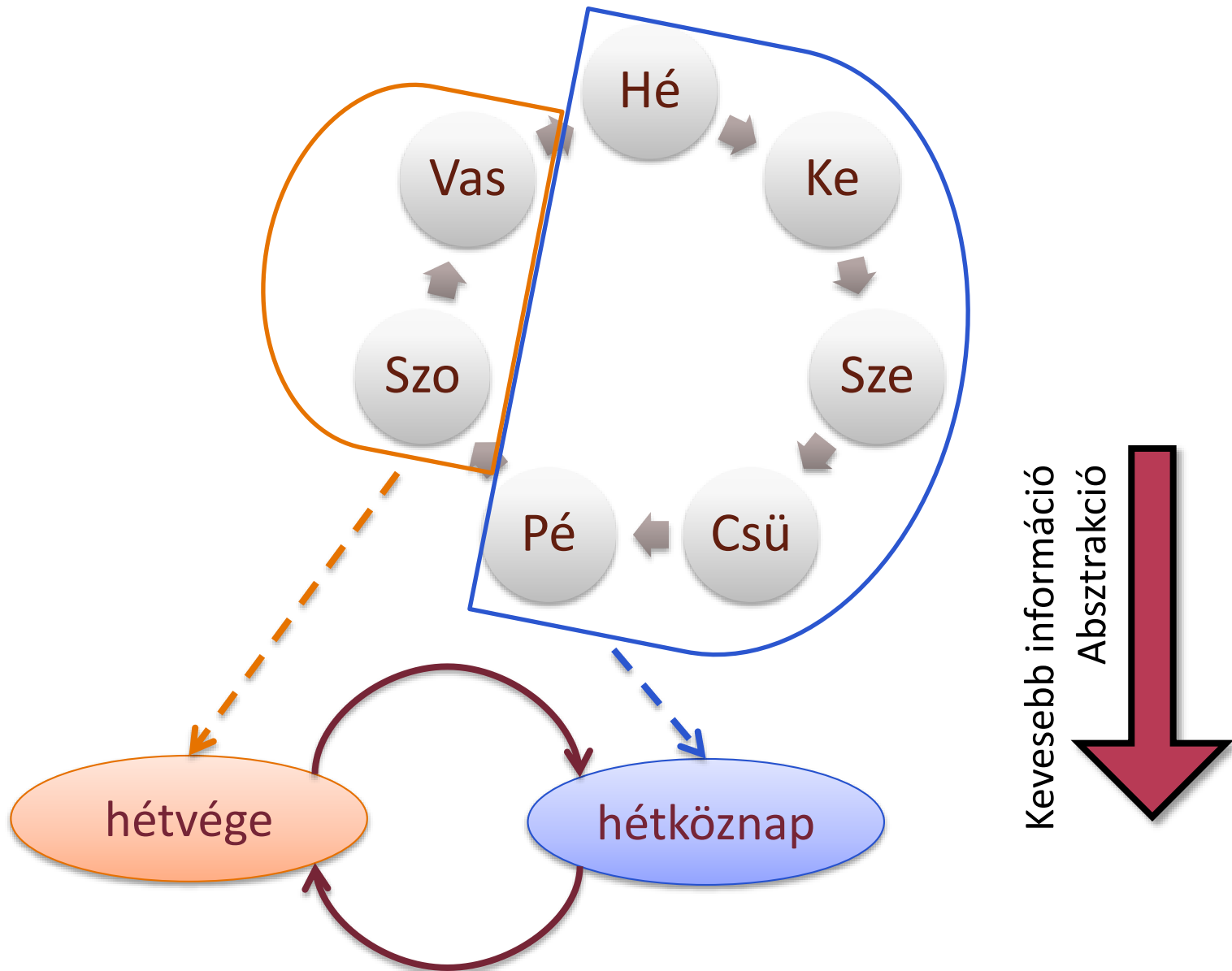


Állapotátmenetek vs. állapotfinomítás

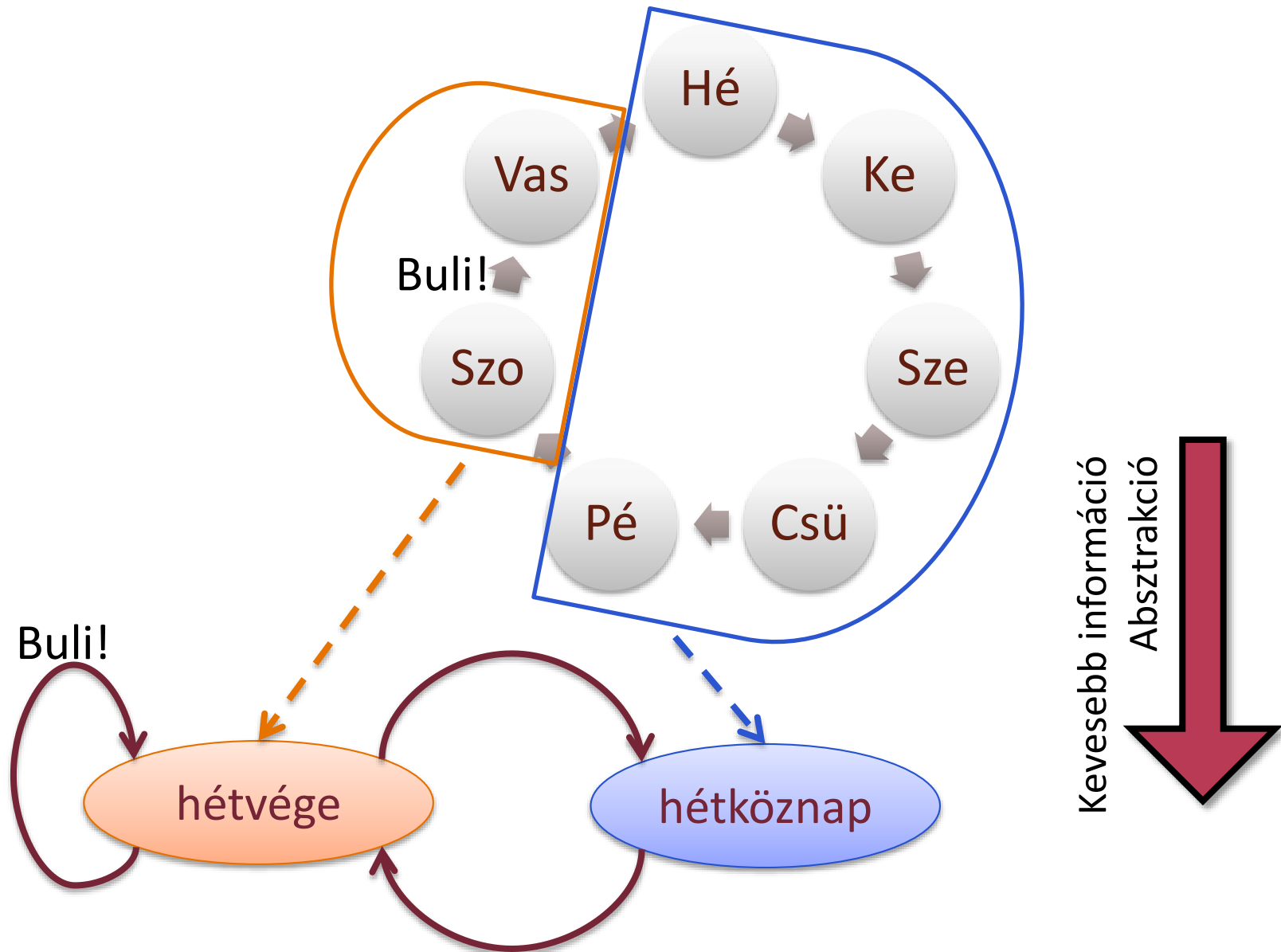
■ Állapotfinomítás hatása átmenetre



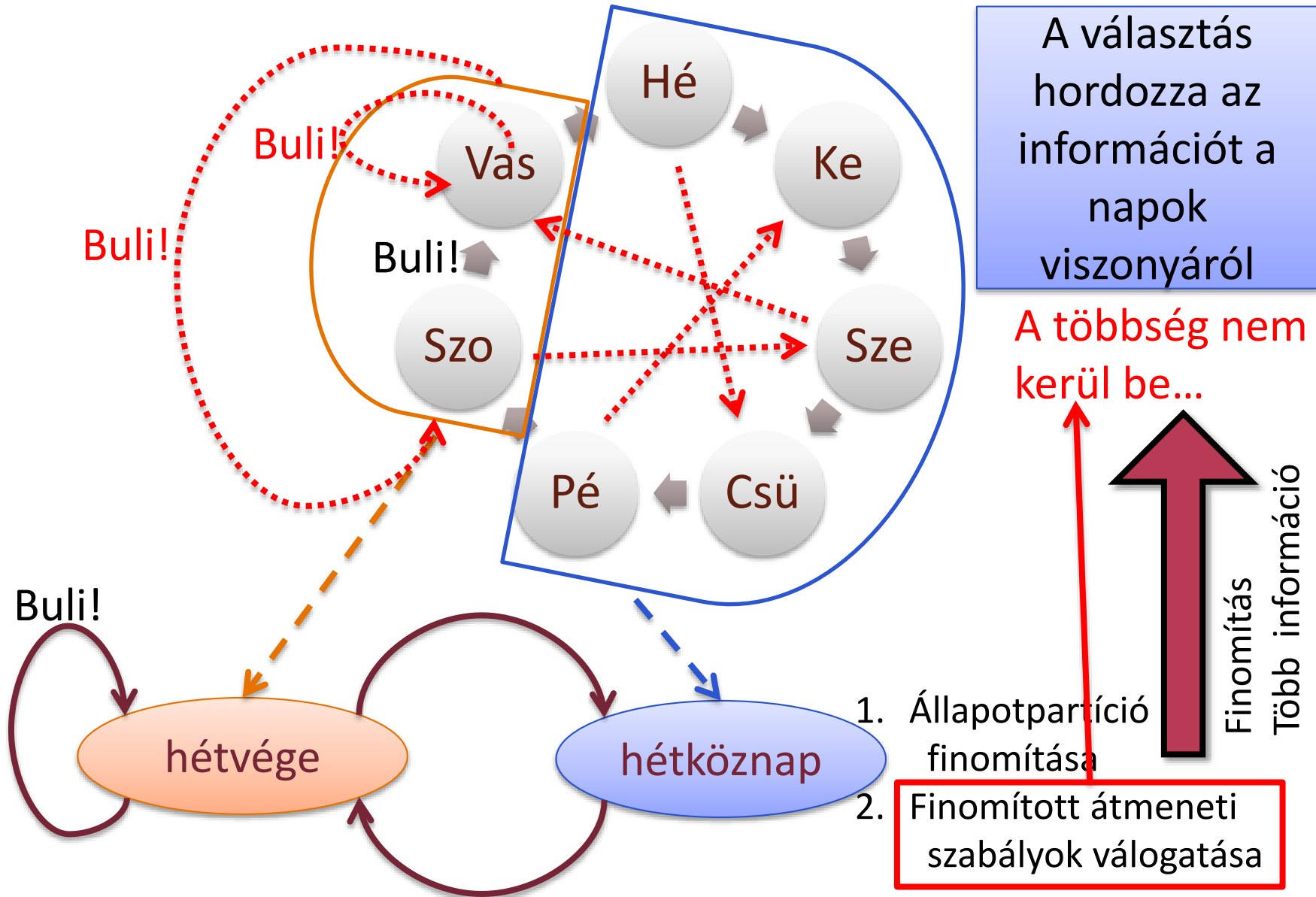
Állapotabsztrakció példa



Állapotabsztrakció példa



Állapotfinomítás példa



Szinkron / aszinkron szorzat

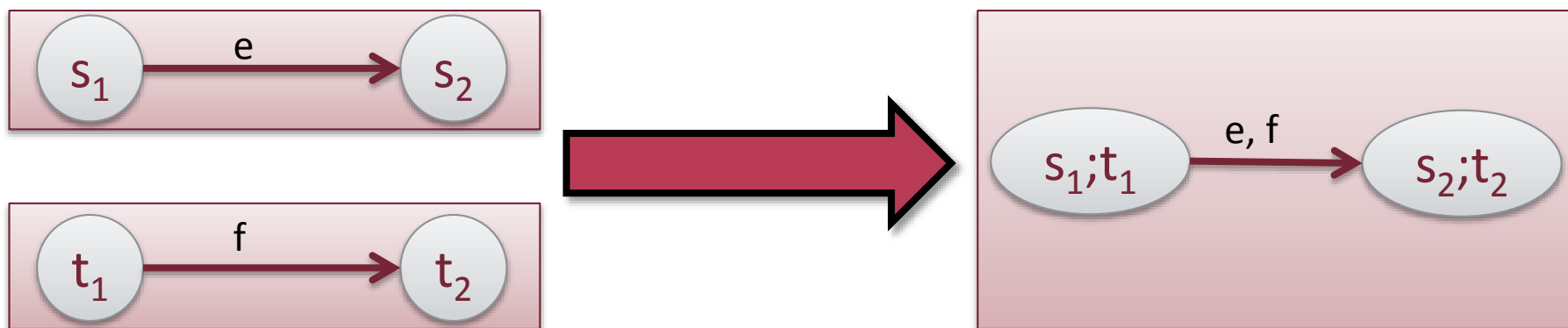
■ Állapotgépek szorzata

- Modell felépítése állapotváltozók állapotgépeiből
- Állapothalmaz: az állapotváltozók terének szorzata

Könnyebb először kisebb modellekben gondolkozni!

■ Átmenetek

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



- A két állapotváltozóra történő visszavetítés itt is absztrakció

Szinkron / aszinkron szorzat

■ Állapotgépek szorzata

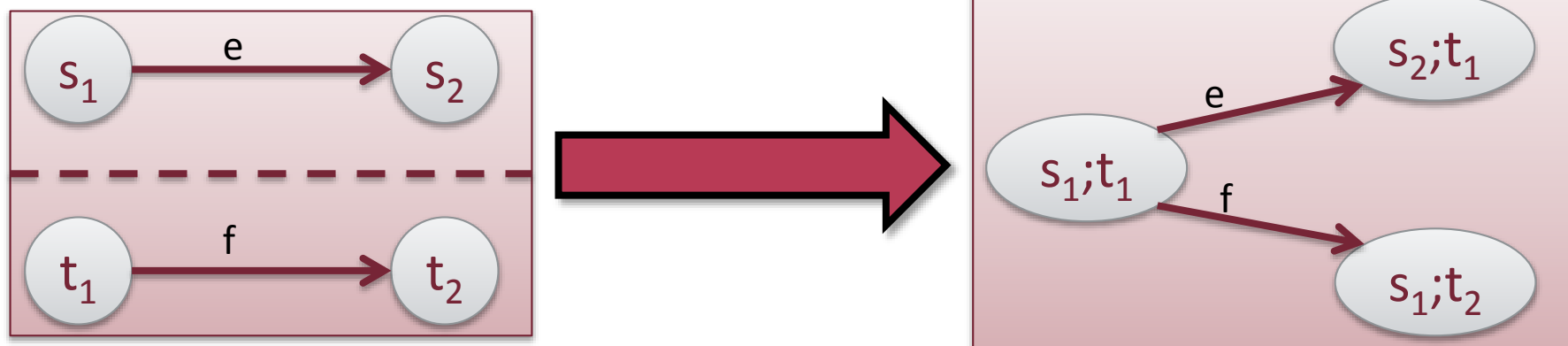
- Modell felépítése állapotváltozók állapotgépeiből
- Állapothalmaz: az állapotváltozók terének szorzata

Könnyebb először kisebb modellekben gondolkozni!

■ Átmenetek

Vegyes szorzat: aszinkron, de helyenként szinkron módon összeolvastva (gyakori!)

- **Aszinkron szorzat:** egyszerre egy állapotváltozó lép
 - Az átmenetek a szorzatba „másolódnak”



- A két állapotváltozóra történő visszavetítés itt is absztrakció

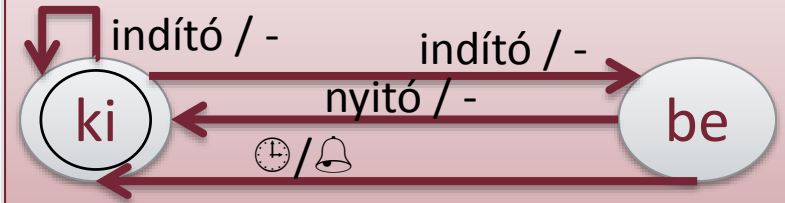
Szinkron szorzat példa

■ Mikró kompozit állapottere

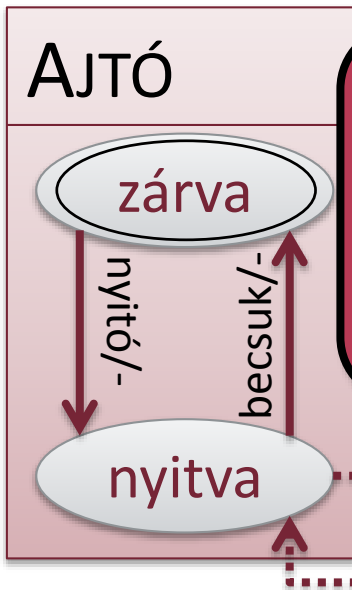
○ További finomítást igényel

- Átmenetek kieshetnek
- Állapotok így a kezdőállapotból elérhetetlenné válhatnak

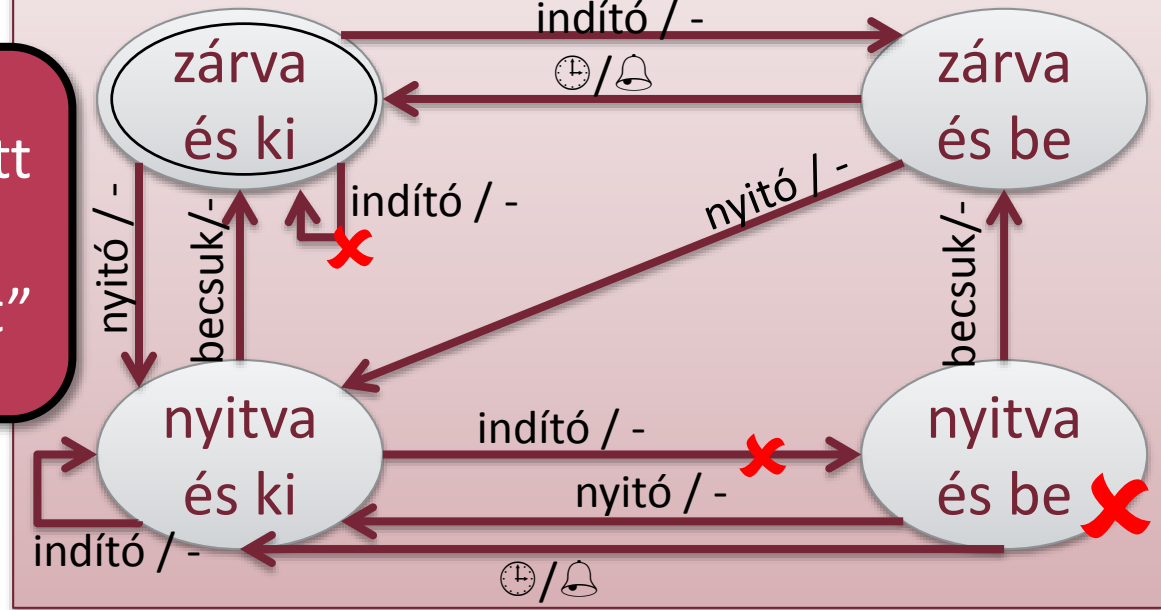
MAGNETRON



AJTÓ ÉS MAGNETRON



Figyelman kívül hagyott inputok: „odaképzelt” hurokél



Aszinkron szorzat példa

■ Kismacska kompozit állapota

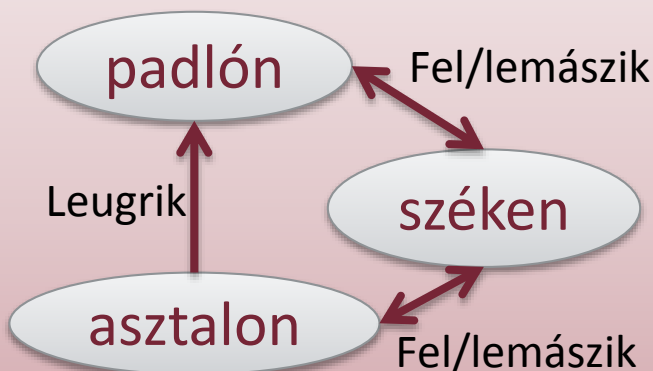
○ További finomítást igényel

- Átmenetek kieshetnek
- Állapotok így elérhetetlenné válhatnak

MACSKA ÉBRENLTÉTE



MACSKA HELYE

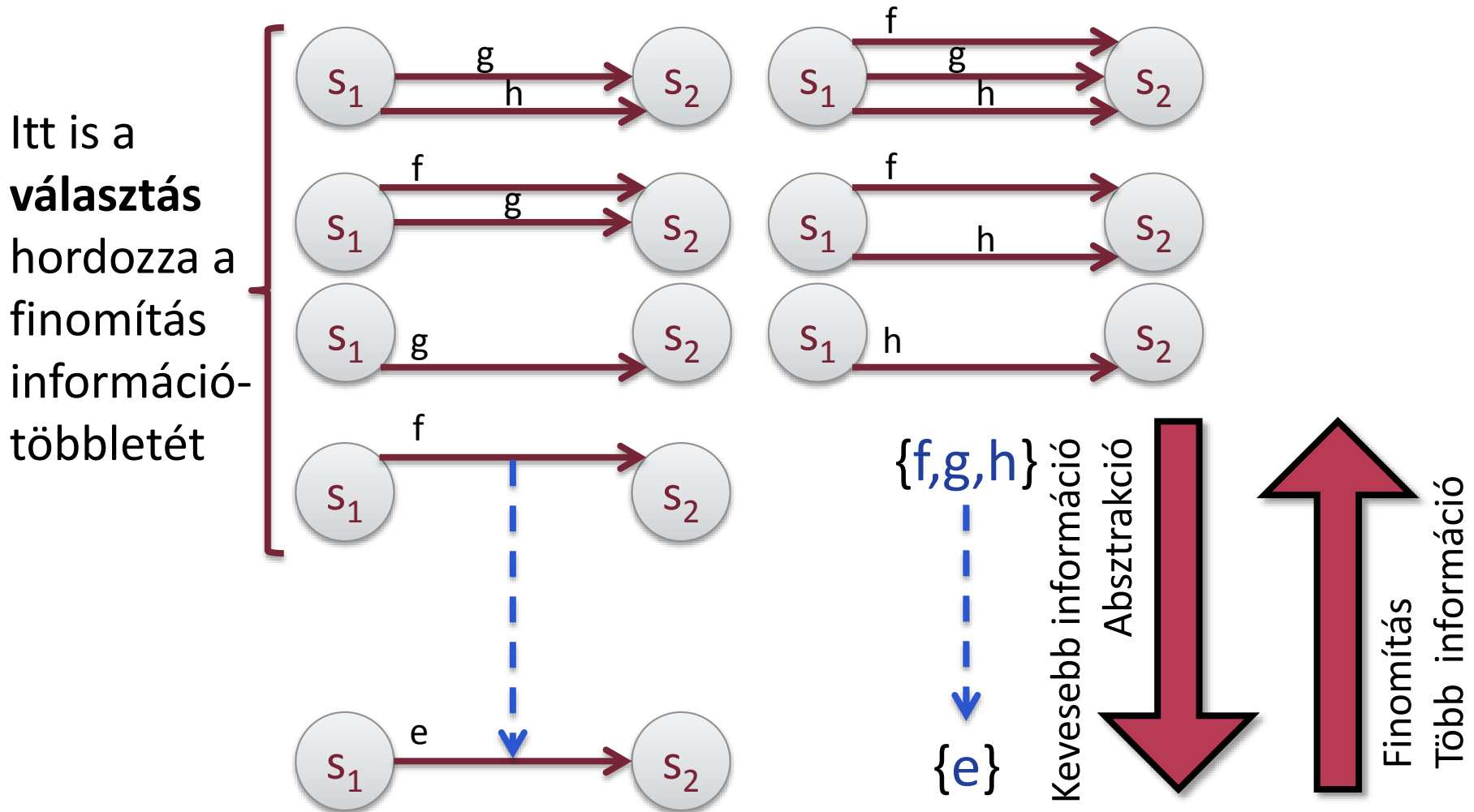


MACSKA HELYE ÉS ÉBRENLTÉTE



Eseményfinomítás és -absztrakció

- Esemény (input/output token) halmazfinomítása



Példa állapot- és eseményfinomításra

MAGNETRON



{olvasztó indító, teljes fokozat indító}

{indító}

Tokenfinomítás
Több információ

MAGNETRON

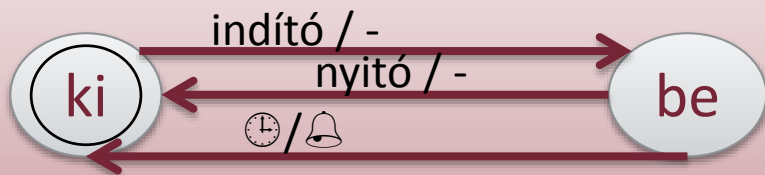


Állapotfinomítás
Több információ

{olvasztó, teljes}

{be}

MAGNETRON



Kitekintés: szakmai példák

- Csomag alapú adatátviteli protokoll (TCP)
 - A teljes üzenet (datagram) több csomagból áll!
 - Az se biztos, hogy eredeti sorrendben érkeznek meg
 - Kommunikációs tokenek megkülönböztetése
 - Csomag sorszáma fejlécben, pl. $ép_3 \rightarrow ACK_3$
 - Csomagonként külön állapotgép a fogadására
 - Teljes fogadó automata: **aszinkron szorzat**
 - (Bővebben ld. Számítógép-hálózatok c. tárgy)
- Ugyanitt tokenfinomításra példa:
 - Ép csomag adattartalma továbbítandó alkalmazás felé
 - Pl. „0” tartalmú ép csomag $\rightarrow$ „0” átadása alkalmazásnak

Ismétlés,
kitekintés

Állapotpartíció

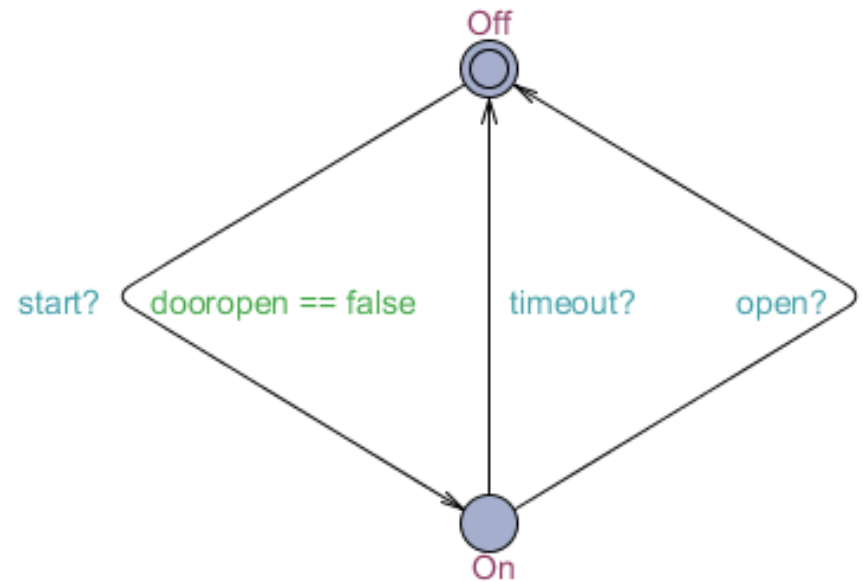
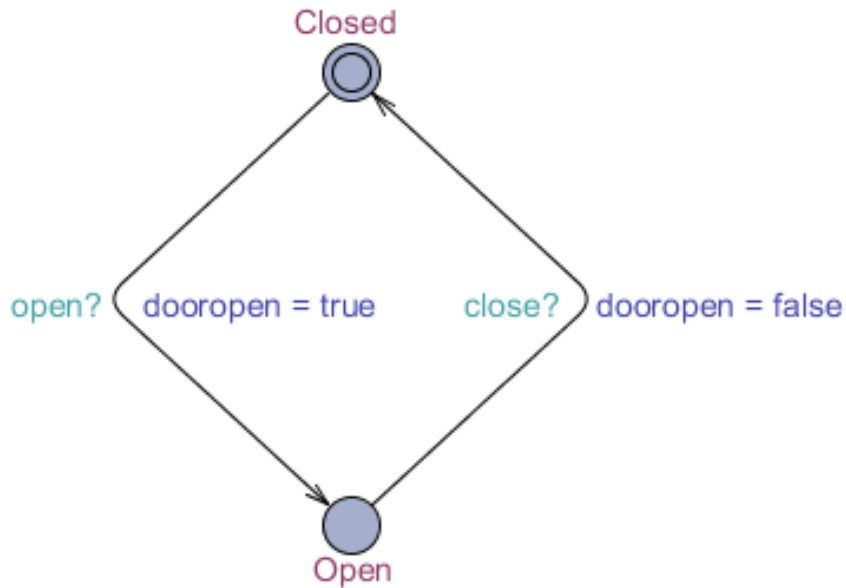
Állapotgépek

Műveletek
állapotgépekkel

Szoftverdemó

ESZKÖZTÁMOGATÁS (PÉLDÁK)

■ Állapotgépek vegyes szorzata



■ Kompozit állapotgép viselkedése...

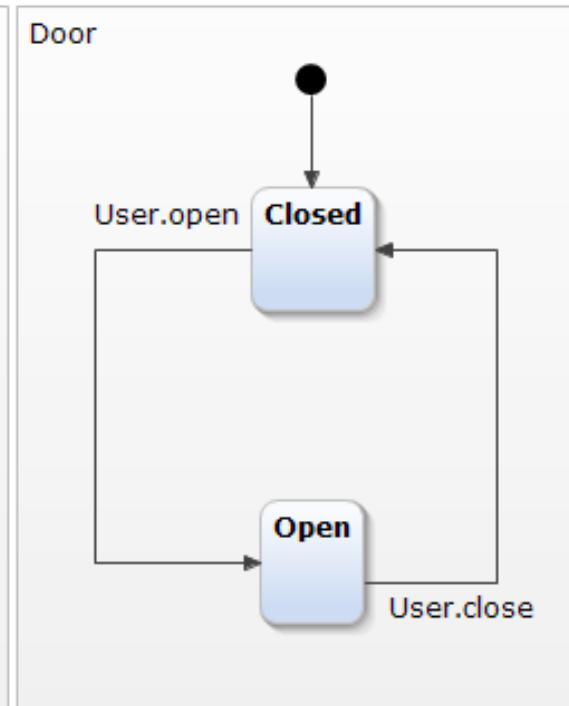
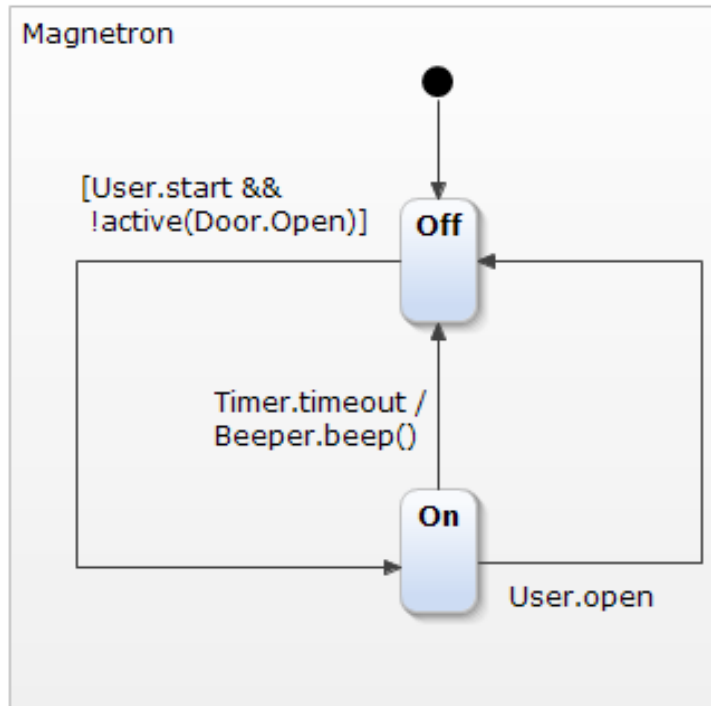
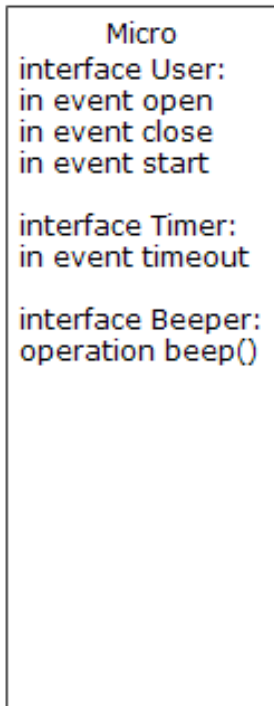
- ...szimulálható
- ...verifikálható

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



Yakindu Statechart Tools

- Kompozit állapotgépek szerkeszthetők 
 - (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 / Állapotpartíció	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