2. gyakorlat – Állapot alapú modellezés – Megoldások

1. Közlekedési lámpa

Közlekedési lámpát vezérlő elektronikát tervezünk.

a) Készítsük el egy egyszerű piros–sárga–zöld közlekedési lámpa olyan állapotterét, amely kellően finom ahhoz, hogy a lámpák vezérlését ez alapján lehessen végezni! Győződjünk meg arról, hogy az állapottér kizárólagos és teljes!

Megoldás

{ *piros, sárga, zöld, piros-sárga, villogó sárga, kikapcsolt* }, röviden: $S = \{p, s, z, ps, \hat{s}, \circ\}$. Ez kizárólagos és teljes. A kezdeti állapot: p.

b) A három égőnek külön-külön mi az állapottere? Milyen absztrakciós viszony áll fent a lámpa és az egyes égők állapottere közt? Hogy viszonyul a lámpa állapottere a három állapotváltozó direkt szorzatához?

Megoldás

$S_p = \{p_1, p_\circ\}$, $S_s = \{s, \hat{s}, \circ\}$, $S_z = \{z, \circ\}$.

Milyen absztrakciós viszony áll fent a lámpa és az egyes égők állapottere közt? Az absztrakciós viszony a *vetítés*. (Szemléletesen: pl. betesszük a lámpát egy dobozba, amiből csak a piros égő látszik ki, akkor az S_p állapotteret látjuk.) Egy komponens állapottere mindig állapottere a teljes rendszer állapotterének (hiszen a többi komponens állapota „elhagyható”, azaz összevonható egy állapotba).

Direkt szorzat: $S_p \times S_s \times S_z$, $2 \times 3 \times 2 = 12$ elemű lesz. A direkt szorzat elemei háromtagú vektorok (más néven n-esek, tuple-ök), pl.: $\langle p, s, \circ \rangle$, amit most ps formában fogunk rövidíteni. Ebből természetesen nem minden állapot fordul elő, ezért $S \subset S_p \times S_s \times S_z$.

A $S_p \times S_s \times S_z$ Descartes-szorzat táblázatban:

		S_z	
S_p	S_s	z	$\circ$
p	s	psz	ps^*
	$\hat{s}$	$p\hat{s}z$	$p\hat{s}$
	$\circ$	pz	p^*
$\circ$	s	sz	s^*
	$\hat{s}$	$\hat{s}z$	$\hat{s}^*$
	$\circ$	z^*	$\circ^*$

A *-gal állapotok az eredeti állapottérben is megjelennek.

c) Mik az érvényes állapotátmeneti szabályok? Készítsük el az (egyszerű) állapotgráfot!

Megoldás

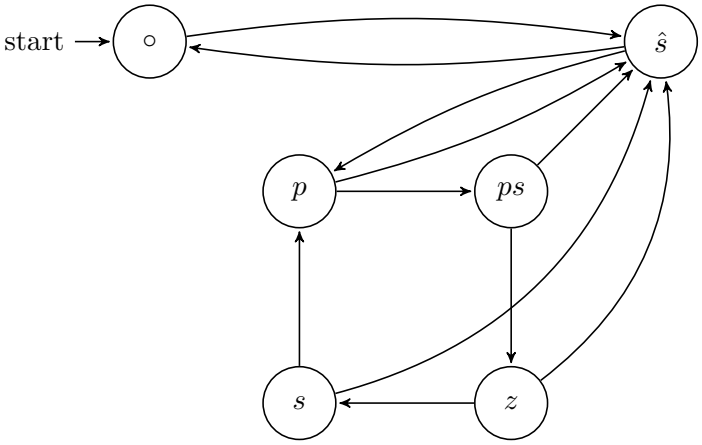
A cél az érvényes állapotátmeneti szabályok megállapítása a hat állapot között. Sok tervezői döntéssel szembesülünk, ezekre egy-egy választ adtunk zárójelben:

- Milyen állapotba kerül bekapcsoláskor? (villogó sárga)
- Csak a szabályos működést modellezzük? (igen)
- Ha elmegy az áram, az hibás működés? (igen, vegyük úgy, hogy csak tervezett áramszünetek vannak)
- Pirosból pirosba mehet? (nem rajzoljuk fel, mert semmi megfigyelhető nem történik)
- Piros-sárgából mehet rögtön pirosba, pl. baleset esetén? (nem)

A KRESZ[1] könyvben található állapotgép-jellegű ábra, de csak a piros, piros-sárga, zöld és sárga állapotokat tartalmazza, a villogó sárga és a kikapcsolt állapotokról csak a szövegben esik szó. Az állapotgép bárhonnan kerülhet villogó sárga ($\hat{s}$) állapotba.

Egy lehetséges megoldás:

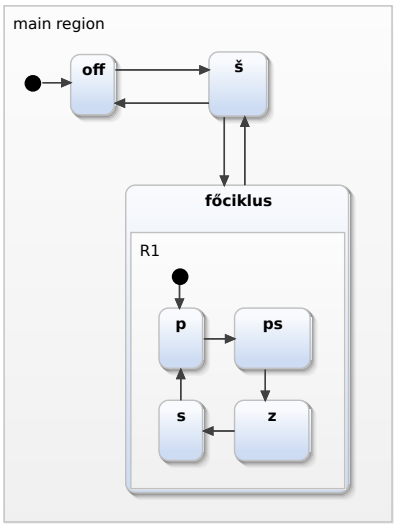
[1] Közúti Rendelkezések Egységes Szabályozása



d) Hogyan fejezhető ki ugyanez a működés tömörebben hierarchikus állapotokkal?

Megoldás

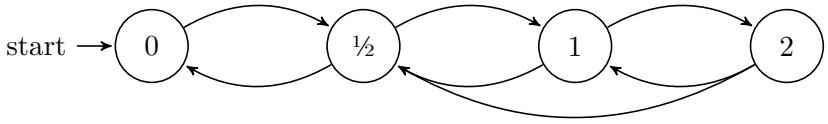
A lámpa bármely bekapcsolt állapotból kikapcsolható – ha tehát a $\{p, ps, s, z, \hat{s}\}$ állapothalmazra közös absztrakcióként bevezetjük az *üzemben* hierarchikus állapotot, akkor a 5 helyett 1 állapotátmeneti szabállyal kifejezhető, hogy üzemi állapotból kikapcsolható a lámpa. Hasonlóan a $\{p, ps, s, z\}$ állapotok is összefoglalhatóak egy *főciklus* hierarchikus állapottá (amely a $\hat{s}$ mellett az *üzemben* másik alállapota lenne), hogy 4 helyett egyetlen szabállyal fejezzük ki azt, hogy a főciklusban üzemelő lámpa a konkrét állapotától függetlenül villogó sárgába kapcsolható.



e) Amikor a lámpa elektromos fogyasztását vizsgáljuk, csak az érdekel, hogy a három égőből hány ég egyszerre. Absztraháljuk az állapotgépet úgy, hogy az állapotokat csak a fogyasztásuk különböztesse meg!

Megoldás

Vegyünk fel egy 4 állapotú állapotteret, az égők fogyasztása alapján: 0, ½, 1, 2 (egyszerre mindhárom égő nem világíthat). Az 1-es állapotra rajzolhatnánk hurokélet (pl. zöld → sárga → piros esetén mindig csak egy lámpa világíthat), de ezt nem tesszük meg, mivel nem figyelhető meg kívülről *ezen az absztrakciós szinten* (ha betesszük a lámpákat egy dobozba, amiből csak annyi látszik ki, hogy 0, ½, 1 vagy 2 égő világít, és nem látjuk a pozíciójukat és a színüket, akkor nem tudunk különbséget tenni például a p, a s és a z állapotok között). Többszörös éleket (ha nincs rajtuk különböző bemenet/kimenet) nem rajzolunk be.



f) (Emelt szintű feladat.) Vegyünk egy olyan eseményszekvenciát, amelynek során az eredeti (absztrakció előtti) modell összes állapota legalább egyszer előfordul. Rajzoljuk fel egy-egy idődiagramon az eredeti és az absztrahált modell viselkedését ugyanezen konkrét végrehajtás során! Hogy viszonyul egymáshoz a két idődiagram tartalma?

g) A piros jelzés végén van egy olyan időszak, amikor a merőleges gyalogosátkelő zöld lámpája már villog. Finomítsuk úgy az (absztrakció előtti) állapotgráfot, hogy ez az állapot elkülöníthető legyen!

Megoldás

Bontsuk fel a piros állapotot két állapotra: piros és a gyalogosoknak folyamatos zöld (p_{fz}), piros és a gyalogosoknak villogó zöld (p_{vz}). Az állapotgráfra is vigyük át a változtatást: a két állapot között $p_{fz} \to p_{vz}$ átmenet van, a többi értelemszerűen berajzolandó.

Itt tovább lehet gondolni azt, hogy szükség van-e olyan állapotra, amikor az autósoknak és a gyalogosoknak is piros a lámpa. Ha az előző részben úgy döntöttünk, hogy a lámpának kötelezően pirosnak kell lennie bekapcsolás után, akkor itt is érdemes ezt megvalósítani: ebben az esetben a piros állapotot három állapotra kell felbontani: p_p, p_{fz}, p_{vz}.

h) Egy út mentén 10 jelzőlámpa található, egyenként 4 állapottal. Legfeljebb hány állapota lehet a teljes rendszernek? Várhatóan kell-e minden állapotot kezelnünk?

Megoldás

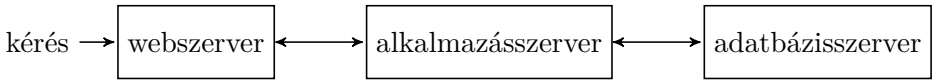
$4^{10} = 2^{20} = \left(2^{10}\right)^2 \approx 10^6$.

Természetesen nem biztos, hogy minden állapot elérhető is lesz, hiszen a jelzőlámpák állapotátmenetei nem biztos, hogy függetlenek egymástól a különböző hangolási kényszerek miatt. Például a zöldhullám beállítása miatt van egy „ritmusa” a 10 jelzőlámpából álló rendszernek.

Gyakran az is hasznos lehet, ha csak azt modellezzük, hogy egy szakaszba összevonunk több kereszteződést és azt vizsgáljuk, hogy az adott szakaszon mehetnek-e autók vagy sem. Pl. a forgalomirányításnál. Ebben az esetben a szakaszok meghatározásához arra van szükségünk, hogy ismerjük rendszer felépítését, azaz legyen strukturális modellünk a rendszerről.

2. Háromrétegű architektúra

Egy informatikai rendszert szeretnénk modellezni, melyet *háromrétegű architektúra* valósít meg az alábbiak szerint:



Háromrétegű kiszolgáló infrastruktúránk viselkedésének modellezésére megfelelő állapotterek-e az alábbiak?

a) { Webszerver dolgozik, Alkalmazásszerver dolgozik, Adatbázisszerver dolgozik }
b) { Leállítva, Tétlen üzemel, Aktívan dolgozik }
c) $\mathbb{N}$ (mint a pillanatnyilag feldolgozás alatt álló kérések száma)
d) { A kérés feldolgozása még nem kezdődött el, A szerverek épp dolgoznak a kéréssel, A kérés kiszolgálása befejeződött }
e) { Igaz } $*$

Megoldás

A használt fogalmak:

- **Adatbázisszerver**: hosszútávon tároljuk az adatokat
- **Alkalmazásszerver**: az üzleti logikáért felelős alkalmazást futtatja
- **Webszerver**: megjelenítést felelős, generálja a HTML oldalakat

Válaszok:

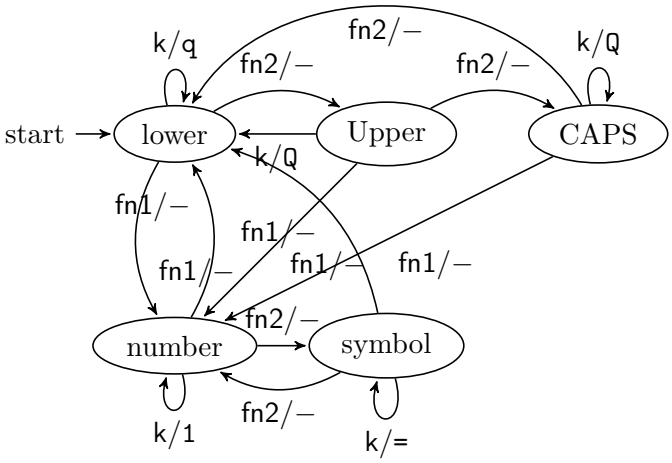
a) Nem. Egyszerre több gép is dolgozhat.
b) Igen. Természetesen elképzelhetők további állapotok is, pl. *hibernált.*
c) Igen. Fontos azonban, hogy ez egy végtelen állapotteret eredményez – bizonyos esetben egy ilyen modellre van szükségünk a rendszer elemzéséhez (ld. még a Teljesítménymodellezés témakört). A kizárólagosság mellett fontos vizsgálni, hogy teljes-e. Pl. 3,5 vagy -9 kérés nem lehet a rendszerben, ezért teljes. Ezzel persze a kérések eloszlását nem tudjuk megnézni.
d) Igen, ha egyértelmű, mit jelent „a kérés”, ugyanis a rendszerünknek természetesen sok kérést kell kiszolgálnia az idő múlásával. Ha teljesen egyértelműen egy adott, jól beazonosítható kérésről beszélünk (pl. Gipsz Jakab hétfő reggel 10:03-kor beérkezett kérése), akkor szigorúan ezen kérés életútjának vizsgálatára teljesen megfelel ez az állapottér, elhanyagolva az összes többi kérés sorsát.

e) Igen, minden állapottér ennek a finomítása.

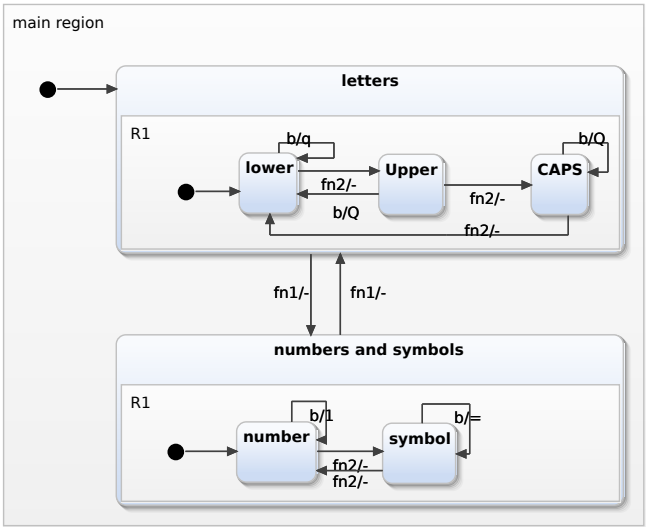
3. Érintőképernyős billentyűzet

Modellezzük állapotgéppel egy mobiltelefon érintőképernyőjére tervezett, az előadáson is bemutatott virtuális billentyűzetet! A billentyűzeten egyszerre vagy a kisbetűk, vagy a nagybetűk, vagy a számok és fontosabb szimbólumok, vagy ritkább szimbólumok láthatóak. Az elsődleges üzemmódváltó gomb a betűk és a számok/szimbólumok beírása között vált, a másodlagos üzemmódváltó pedig ezen kategóriákon belül. Létezik továbbá egy olyan nagybetűs állapot is, amely egy betű leütése után automatikusan kisbetűsre vált. Vegyük figyelembe a bal felső gombot (q/Q/1/=), ill. a két üzemmódváltó gombot mint bemenetet, és a szövegmezőbe begépelt karaktereket mint kimenetet!

Megoldás
A billentyűzet képernyőképeken:
Fontos, hogy csak egy billentyűt kell feltüntetnünk az állapotgépen, ez segíti az áttekinthetőséget is. Jelöljük a bal felső gombot k-val (key), az fn helyén lévő gombot fn1-gyel, a shift helyén lévő gombot fn2-vel. Egy lehetséges megoldás az alábbi:



A hierarchikus állapotgép:

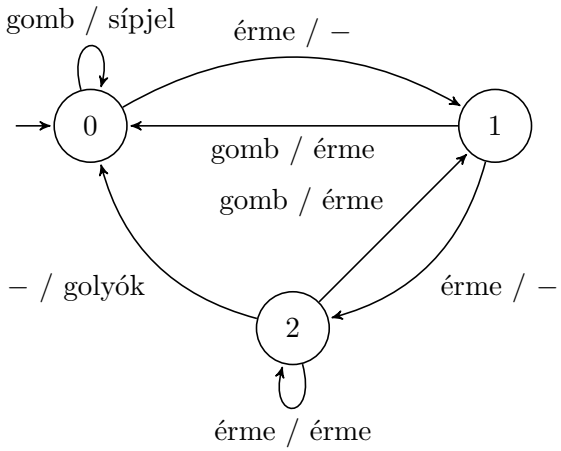


A jelenlegi megoldásnál a *number* és a *symbol* állapotokból az fn megnyomására mindig a lower állapotba kerül az automata. A valóságban a fejlettebb implementációk képesek megjegyezni azt, hogy a lower, Upper és CAPS állapotok közül melyikben voltak és abba lépnek vissza.

Ez az állapotgépben úgy valósítható meg, hogy finomítjuk a „number” és a „symbol” állapotokat (numbers with lower, numbers with Upper, numbers with CAPS, …), valamint *emlékező állapotot* (history state-et) vezetünk be. Erről a *Szoftvertechnológia* tárgyban lesz részletesen szó.

4. „Mit ír ki?”

Tekintsük az M állapotgépet!



Megjegyzés: a „−” jel a bemeneti jel pozíciójában spontán állapotátmenetet jelöl, a kimenet helyén pedig kimenet hiányát, egyik esetben sem a digitális áramkörtervezésben használt *don't care* szimbólum.

a) Milyen valós rendszer lehet az M állapotgép mögött, hogyan működik?

Megoldás

Az állapotgép egy fizetős biliárdasztal érmebedobó modulját („érmeszerkezet”) modellezi, amely két érme (pl. 100 forintos) bedobása után kiadja a játékhoz szükséges golyókat. A golyó kiadása, pénz bedobása, stb. mind pillanatszerű eseményként van modellezve (ez egy modellezés során alkalmazott egyszerűsítés). A rendszer állapotai az adott pillanatban az érmeszerkezetben található érmék darabszámát tükrözik. A gomb egy bedobott érmét ad vissza, ha a gépben van érme, egyébként sípjelet hallat. A rendszer nem engedi, hogy kettőnél több érme legyen a gépben. Amennyiben két érme van a gépben, rövid várakozás után egy *spontán állapotátmenet* keretében az érmeszerkezet időleges tárolójából átkerülnek az érmegyűjtőbe, az asztal pedig ezzel egyidejűleg a játékosok rendelkezésére bocsájta a golyókat. Ha a játékosok meggondolják magukat, a golyók kiadása előtt gyorsan visszakérhetik még az érméiket a gomb benyomásával.

b) Determinisztikus-e ez a viselkedésmodell? Hozzávehető-e, ill. elhagyható-e egyetlen állapotátmeneti szabály, hogy ez megváltozzon?

Megoldás

Mikor determinisztikus egy állapotgép? Akkor, ha „legfeljebb egy kezdőállapota van, valamint bármely állapotban, bármely bemeneti esemény bekövetkeztekor legfeljebb egy tranzíció tüzelhet.” Ez úgy dönthető el, ha végignézünk minden állapotot és a kimenő élekre ellenőrizzük, hogy minden bemenet esetén csak egy él megy-e ki, ill. nincs-e spontán átmenet. Itt minden állapotból csak egy él megy ki, azonban van olyan állapotátmenet, amelynél „−” karakter van a bemeneten (tehát bármely bemenet helyett választható ez), ezért az állapotgép *nem determinisztikus*. Erről teljes bizonyosságot szerezhetünk a következő gondolatmenettel: ha bedobunk 2 érmét, majd 5 másodperc múltán megnyomjuk a gombot, sípjelt vagy érmét kapunk-e vissza? Ez attól függ, hogy a gép feldolgozta-e a két érmét a gombnyomás előtt (tehát végbemegy-e ez a spontán átmenet).

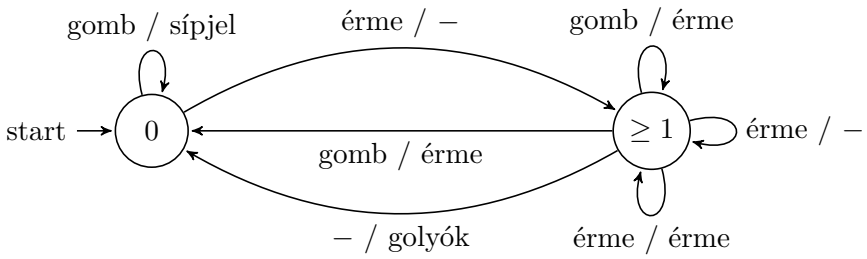
Megállapíthatjuk, hogy a modell nem tartalmazza azt az információt, amely alapján előre eldönthetnénk, hogy a két lehetőség közül ebben a helyzetben melyik fog megtörténni; nem határozza meg (determinálja) teljesen a viselkedést. Ennek egyik lehetséges oka, ha maga az érmeszerkezet konstrukciójából adódóan véletlenszerűen dönti el, lejárt-e a türelmi idő; a nemdeterminizmus másik lehetséges oka a modellezés során alkalmazott absztrakció (pl. egy időzítő határozza meg, mikor történjen meg az itt spontánként modellezett állapotátmenet).

Determinisztikussá tehető-e az állapotgép? Igen, a „2” és a „0” állapotok közötti tranzíció törlésével.

c) Absztraháljuk az M állapotgráfot a $\{\{0\},\{1,2\}\}$ állapotpartíció szerint!

Megoldás

A megoldás az M' állapotgép, melynek állapotgráfja:



Az M' állapotgép az M állapotgépből 1 és 2 állapotok összevonásával keletkezik.
A kapott állapotgép az eredeti állapotgép egy absztrakciója, mely az eredeti állapotgép végrehajtássorozatain kívül számos más végrehajtássorozatot is megenged. Ezek közül néhány akár valós rendszer viselkedése is lehet: az absztrakt rendszer a közös absztrakciója az n érmével ($n > 1$) működtethető asztaloknak. Az absztrakt állapotgép ugyanakkor olyan végrehajtássorozatokat is megenged, amelyeket nem egy valós biliárdasztal viselkedését modellezik.

d) Hol és milyen jellegű nemdeterminizmus figyelhető meg az így kapott absztrakt modellen?
Megoldás
Az absztrakt állapotgépben a spontán állapotátmeneten kívül is jelenik meg nemdeterminizmus: ≥ 1 állapotban a gomb, illetve érme bemeneti események több tranzíciót is kiválthatnak. A most újonnan bevezetett nemdeterminizmus forrása az absztrakció: elhanyagoljuk azt a tudást, amely alapján determinisztikusan dönteni lehetne (ez jó is lehet, pl. ha sok rendszerről egyszerre akarunk beszélni).

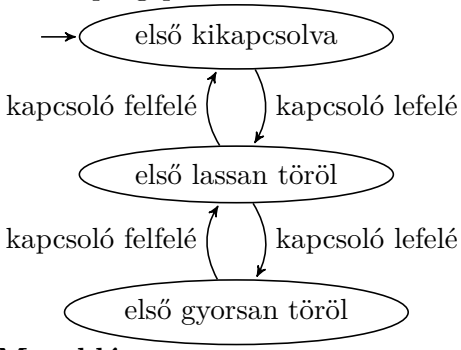
e) (Emelt szintű feladat.) Vegyünk egy olyan eseményszekvenciát, amelynek során az eredeti (absztrakció előtti) modell összes állapotátmeneti szabálya legalább egyszer végbemegy. Rajzoljuk fel egy-egy idődiagramon az eredeti és az absztrahált modell viselkedését ugyanezen konkrét végrehajtás során! Hogy viszonyul egymáshoz a két idődiagram tartalma?

5. Ablaktörlő

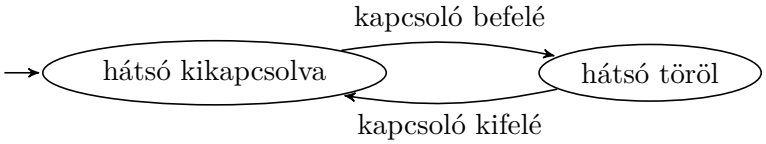
Egy autóban az első ablaktörlőnek három állapota van (*első kikapcsolva*, *első lassan töröl*, *első gyorsan töröl*), a hátsó ablaktörlőnek kettő (*hátsó kikapcsolva*, *hátsó töröl*). Az első ablaktörlő működését az M_1 állapotgép, a hátsóét az M_2 állapotgép modellezi.

a) Készítsük el az M_1 és M_2 állapotgépek aszinkron szorzatát!
b) Hány állapota és átmenete van az így kapott modellnek?
c) (Kiegészítő feladat) Kifejezhető-e a kapott állapotgépen, ill. a komponensre vetített modellek segítségével olyan kocsi, ahol a hátsó ablaktörlő csak akkor kapcsolható be, ha megy az első is?

M_1 állapotgép:

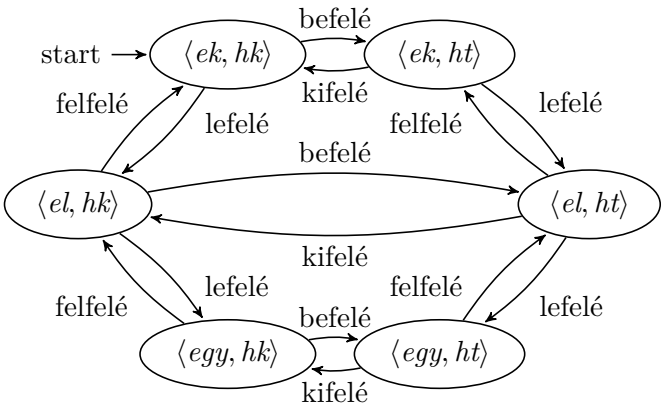


M_2 állapotgép:



Megoldás

a) Az aszinkron szorzat:



b) Az állapotok száma a komponensek állapotszámainak szorzata. Az átmenetek száma $|S(M_1)| \cdot |T(M_2)| + |S(M_2)| \cdot |T(M_1)|$.

c) A teljes rendszert leíró állapotgép nyilván módosítható megfelelően – egyszerűen csak elhagyjuk a nem megengedett átmeneteket. A komponensekre vetítéskor viszont bajba jutunk, hiszen a két komponens működése már nem független: a hátsó ablaktörlő szóban forgó átmeneteit csak akkor szabad használni, ha az első ablaktörlő állapota megfelelő. Ilyen megszorítást *őrfeltétellel* fejezhetünk ki.

Az aszikron szorzatban mindkét állapotgép explicit megjelenik, mindkettő annyiszor, ahány állapota van a másiknak. Intuitívan: az egyik állapotgépet „elvihetjük” egy tetszőleges állapotába, és ott „lejátszhatjuk” a másik összes viselkedését, és ugyanez igaz fordítva.