Szoftverfejlesztési technológiák 07-09

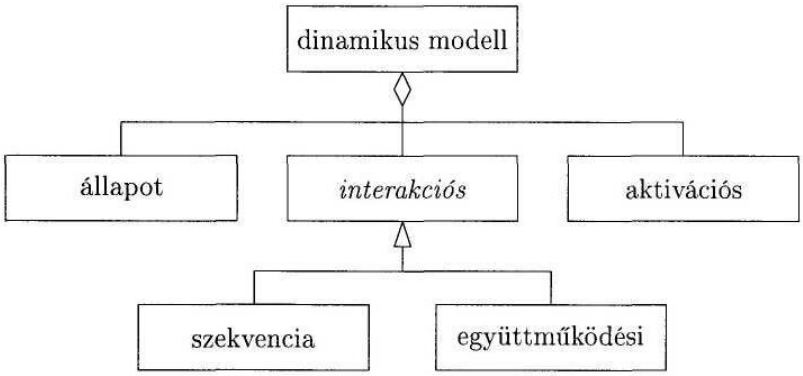
/óravázlat/

Állapotdiagram

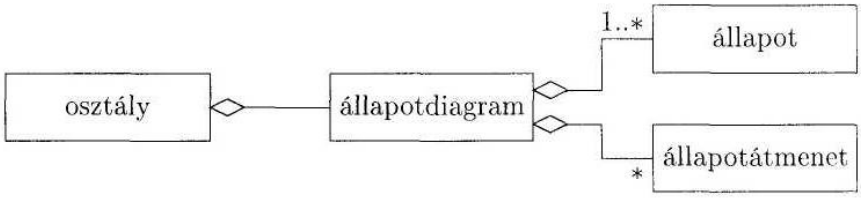
Az előzőekben megismerkedtünk a statikus modell alkotóelemeivel. Most az objektumelvű modellezés egy másik fontos elemének, a dinamikus modellnek a bemutatását kezdjük el. A dinamikus modell három fő részből tevődik össze: az állapotdiagramokból, az állapotok leírásából és az események leírásából. Azaz:

dinamikus modell = állapotdiagramok +
állapotok leírása +
események leírása.

Az események leírására szolgál a szekvencia-, az együttműködési és az aktivációs diagram. A szekvencia- és az együttműködési diagram az objektumok interakcióját (egymásra hatását) határozza meg, így a dinamikus viselkedést leíró diagramok csoportosítása:



Az osztály objektumainak dinamikus viselkedését a probléma megoldása során az állapotdiagram írja le. Az állapotdiagram egy állapotautomatát ábrázol. Az osztály dinamikus tulajdonságai alapján lényegében egy állapotautomata, mely állapotoknak és állapotátmeneteknek az összessége:



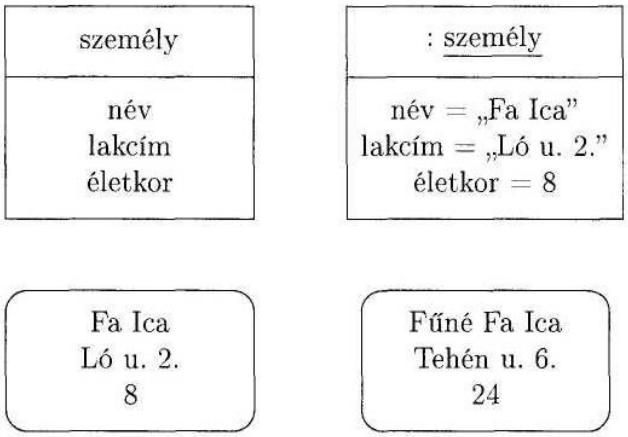
A dinamikus modell leírásának megértéséhez be kell vezetnünk néhány alapfogalmat és a hozzájuk tartozó jelöléseket. A bevezetendő alapfogalmak:

- állapot,
- esemény + jelölésrendszer,

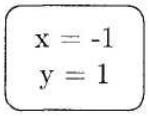
• állapotdiagram.

Minden objektumnak van egy életciklusa. Az objektum létrejön, aktív életet él, azaz más objektummal működik együtt különböző feladatok megoldásában, majd megsemmisül. Számos esetben azonban (kvázi) végtelen folyamatot vizsgálunk. Ilyenkor az objektum megsemmisülése nem játszik szerepet a vizsgálatban. Az objektum aktív állapotaiban különböző objektumokkal kerül kapcsolatba, és ennek eredményeként különböző *állapotokba* kerül.

Állapot: Az objektum állapotát az attribútumok konkrét értékeinek *n*-esével jellemezzük:



Ugyanígy, ha síkbeli pontokat koordinátáikkal jellemzünk, akkor egy pont lehetséges állapota:



Ez az állapot megváltozhat pl. eltolás vagy forgatás hatására.

Esemény: Eseménynek nevezzük azt a tevékenységet, történést, folyamatot, amely valamely objektum állapotát megváltoztatja.

Eseményre példa lehet az előző esetekben a házasságkötés vagy az eltolás, forgatás, illetve ha pl. az útkereszteződésnél a lámpa pirosra vált. (Ekkor ugyanis az útkereszteződésnek mint objektumnak az állapota megváltozik.)

Az objektum állapota rendszerint nem egy adott pillanatban képezi a vizsgálat tárgyát. A személyekre vonatkozó példa esetén az lehet érdekes számunkra, hogy az adott személy általános iskolás-e, azaz objektumunk olyan állapotban van-e. Ezt az állapotot egy időintervallummal jellemezhetjük az életkor attribútumra vonatkozóan:

$7 < \text{életkor} < 14$,

miközben a többi attribútum tetszőleges lehet.
Az ilyen értelemben vett állapot az ún. osztályhoz rendelt állapot.

Osztályhoz rendelt állapot: Azokhoz az objektumokhoz rendelt állapotokat, amely objektumok az eseményekre, (az események egy bizonyos halmazára) azonos módon reagálnak, egy halmazba vonjuk össze, és ezt a halmazt osztályhoz rendelt állapotnak nevezzük. *Az osztályhoz rendelt állapot tehát az állapotoknak egy részhalmaza, amely az állapotinvariánssal jellemezhető.*

Ilyen invariáns pl. az előbbi példában az, hogy az életkor 7 és 14 év között kell legyen, miközben a többi attribútum tetszőleges lehet.

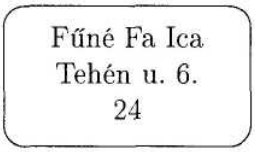
A következőkben állapot alatt mindig osztályhoz rendelt állapotot értünk.

Az állapotok és az események időhöz vett viszonyát a következőképpen jellemezhetjük:

- az állapothoz időintervallum tartozik,
- az eseményhez pedig időpont tartozik (általában).

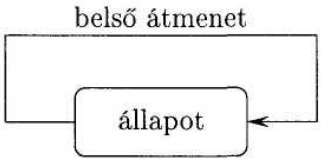
Az állapot a következő tulajdonságokkal rendelkezik:

1. Az állapotnak van azonosítója. Az állapotok egymástól megkülönböztethetők, például van nevük. Nem csak név azonosíthat egy állapotot. Lehet az azonosító egy vagy több attribútumának konkrét értéke, illetve ezen értékeket meghatározó állítás, feltétel, azaz az állapotinvariáns. Az állapot jelölése egy lekerekített sarkú téglalap, amelybe az azonosító kerül:

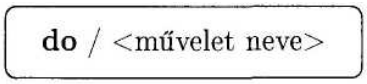


2. Az állapot általában esemény, eseménysorozat hatására jön létre. Ezeket az eseményeket megelőző eseményeknek (pre-events) nevezzük. Például az iskolás állapot a beiratkozás hatására jön létre. Egy speciális, rendszeren kívüli állapot a kezdeti állapot.

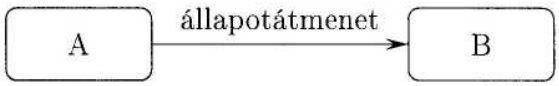
3. Az állapot időben mindaddig fennmarad, amíg az objektumok attribútumainak értékei kielégítik az állapothoz rendelt invariánst. Az állapot fennállása során belső átmenetek fordulhatnak elő. Ezek nem változtatják meg az objektum állapotát, azaz érvényes marad továbbra is az állapotinvariáns:



4. Az állapotokat gyakran nehéz megfelelően kifejező névvel ellátni. Ezért gyakran azonosítóként használjuk a belső tevékenységek, belső műveletek nevét. Ennek jelölése:



5. Az állapot megszűnése egy esemény hatására következik be. A megszűnéshez egy eseménysorozat kötődhet (rákövetkező események, post-events). Azaz az objektum egy külső állapotátmenet hatására egy másik állapotba kerül. Jele:



6. Az objektum megszűnése ugyancsak egy állapotátmenet hatására következik be. Ekkor az objektum egy rendszeren kívüli, úgynevezett befejező állapotba kerül.

E definíció alapján az állapotot a következő formáknak alávetett leírással adhatjuk meg:

state: <az állapot azonosítója>;

comment: <az állapot rövid, magyarázó leírása>;

pre-events: <az állapotot előidéző, megelőző események azonosítóinak listája>;

invariant: <az állapot invariánsának leírása>;

post-events: <az állapot megszűnéséhez kötődő, rákövetkező események listája>.

Példaként adjuk meg – ennek megfelelően – az ébresztőóra „csengetés” állapotának leírását:

state: ébresztő csengetés;
comment: amikor az előzetesen beállított időpont bekövetkezik, a csengő megszólal és 10 másodpercig csörög;
pre-events: idő beállítás (ébresztési idő); aktuális idő = ébresztési idő;
invariant: ébresztés bekapcsolva és ébresztési idő < aktuális idő < ébresztési idő + 10 mp;
post-events: aktuális idő = ébresztési idő + 10 mp.

Az eseményeket a következők jellemzik:

1. lehet paraméter nélküli (például az „enter” gomb megnyomása);

2. lehet paraméteres;

3. az események között sorrendiség állhat fenn, azaz beszélhetünk:

   • megelőző eseményről, illetve
   • rákövetkező eseményről;

4. az eseménynek előfeltétele is lehet.

Általában az események a mindennapi életben is sorrendben egymáshoz kötődnek, például:

• reggel kimegyek az utcára (megelőző esemény);
• hőmérséklet fagypont alatti (feltétel);
• felhúzom a kesztyűmet (esemény, egyben a következő esemény előfeltétele);
• a kezem felmelegszik (rákövetkező esemény).

Az esemény formáknak alávetett leírása a fentiek alapján a következő lehet:

event: <esemény neve, azonosítója>;
comment: <az esemény jelentésének rövid leírása>;
parameters: <a paraméterek listája>;
precondition: <az esemény bekövetkezését szükségszerűen megelőző állítás, feltétel>;
pre-events: <az eseményt megelőző esemény>.

Ennek alapján *az esemény általános formája:*

<esemény>(<paraméterek>)[<feltétel>]/<megelőző esemény>,

ahol a feltétel vonatkozhat

- paraméterekre (előfeltétel),
- a megelőző eseményekre (szinkronizációs feltétel).

Példaként tekintsük a MALÉV MA-416-os London-Prága-Budapest járatát, az esemény legyen a gép megérkezése Budapestre (áll a gép és szállnak ki az utasok). Ekkor:

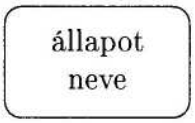
megérkezik[Londonból felszállt ^ Prágába leszállt ^ Prágából felszállt]/Budapesten leszállt.

Az állapotdiagram definíciója

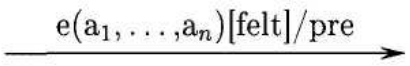
Az állapotdiagram egy összefüggő irányított gráf, amelynek csomópontjaihoz az állapotokat rendeljük, éleihez pedig az eseményeket.

Csak olyan állapotokat vizsgálunk, amelyek létrejönnek, azaz elérhetők a kezdő állapotból, ezért a gráf összefüggő. Ugyanakkor egy állapotátmenet több esemény hatására is létrejöhet, ezért két csúcsot több irányított él is összeköthet.

Az állapot jelölése:



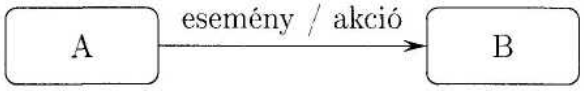
Az élekhez rendelt (átmeneteket okozó) események jelölése általános formában:



Ezenkívül fontos még a rendszeren kívüli állapotok jelölése:

Esemény és akció

Az eseményt mi úgy definiáltuk, hogy az egy időpillanat alatt játszódik le. Számos esetben azonban az esemény végrehajtása időben elhúzódhat. Ilyenkor célszerű megkülönböztetni az eseményt és az akciót egymástól. Ekkor az *akció* az, ami egy időpillanathoz kötődik. Az állapotok közötti átmenetet tehát esemény vagy akció eredményezheti:



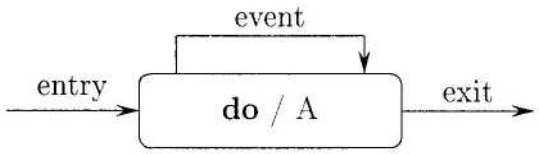
Ha a megelőző állapotból történő kilépés és a következőbe történő belépés ugyanaz az időpillanat, akkor akcióról beszélünk, ellenkező esetben eseményről, amely a hozzá köthető állapotban tartja az objektumot arra az időre, amíg az esemény zajlik.

Úgy is értelmezhetjük, felfoghatjuk, hogy az eseményeknek (általában) három fázisuk van, amelyek közül kettő akció:

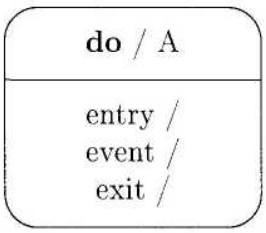
• Egy *entry fázis,* a belépés akciója, amely elindítja azt a történést, amelynek hatására létrejön egy eseményhez rendelt állapot.

• Egy *event fázis,* amely az adott állapothoz kötődik, azaz belső események sorozata, amely az adott állapothoz kötődő belső állapotokat jelenti.

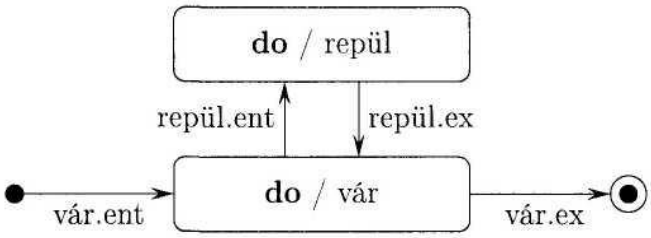
• Egy *exit fázis,* a kilépési akció, amely az esemény befejezését, a hozzá rendelt állapotból való kilépést eredményezi.



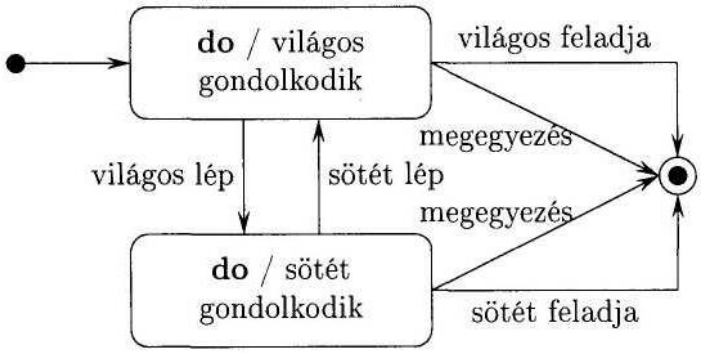
Az esemény három fázisának jelölése:



Példaként tekintsük a repülőjárat-nyilvántartásnak egy leegyszerűsített modelljét. A repülők a regisztráció után a felszállásra várnak, repülnek, a következő állomáson várnak, stb., az utolsó állomáson várnak, majd befejezik a tevékenységet, kijelentkeznek. A várakozás és a repülés időben elhúzódhat, azaz ezek események. A regisztráció felel meg a várakozás belépési akciójának, a kijelentkezés pedig a kilépési akciónak. A repül esemény entry fázisa a felszállás, exit fázisa pedig a leszállás. Az ábrában az entry fázist „ent”-ként, az exit fázist „ex”-ként rövidítettük. A továbbiakban is ezt a jelölési konvenciót használjuk.

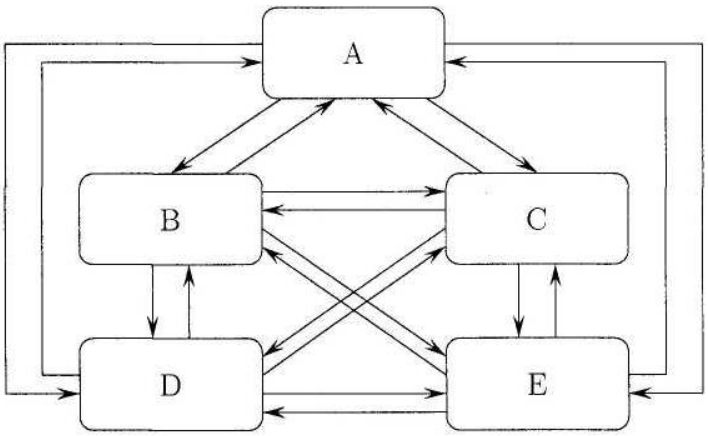


A következő példa egy sakkjátszma nyilvántartása; az állapotdiagram a játszma lefolyásának legabsztraktabb szintjét mutatja be:



Az állapotdiagram bonyolultsága

Az állapotdiagram viszonylag egyszerű esetekben is áttekinthetetlenné válhat. Tekintsük azt az állapotdiagramot, amelynél 5 állapot van, és mindegyik állapotból mindegyik másik állapotba történhet átmenet:



Az ábra már ebben az esetben is áttekinthetetlen, bonyolult. Általános esetben, *n* állapot esetén, ilyenkor az állapotátmenetek száma *n* * (*n* - 1). Noha ennek a szélső esetnek az előfordulási valószínűsége elenyésző, az állapotok számának növekedésével az állapotdiagram egyre áttekinthetetlenebbé válik. Ezért szükséges az így előálló bonyolultságot kezelnünk, nevezetesen egyszerűbbé és áttekinthetőbbé tenni az állapotdiagramot.

A bonyolultság csökkentésére két általános módszer létezik:

- az állapotok általánosítása és

• az állapotok aggregációja.

Mindkét esetben állapotokat vonunk össze, csoportosítunk egyetlen állapotba, és az állapotdiagramban ezt az összevont állapotot tüntetjük fel. Az összevont állapotot megadhatjuk akár függetlenül, külön diagramban is. Így egy adott szinten egyrészt az állapotok száma csökken, másrészt az összevont állapotok belső átmenetei sem jelennek meg.

Állapotok általánosítása

Az általánosított állapot a következő tulajdonságokkal rendelkezik:

1. Az általánosított állapot véges számú részállapot összessége.

2. A részállapotok örökölhetik az általánosított állapot tulajdonságait:

    a. attribútumait (állapotjellemzőit);
    b. eseményeit, akcióit.

3. Az objektum az általánosított állapotban mindig valamelyik részállapotban van.

4. A részállapotok lehetnek általánosított állapotok is.

5. Az általánosított állapothoz állapotdiagram tartozik, amelyben a részállapotok közötti átmeneteket tüntetjük fel.

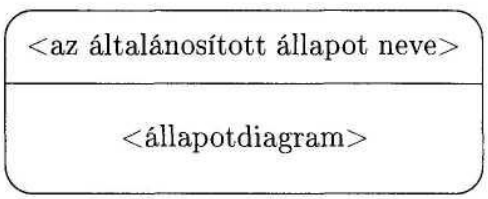
6. Az általánosított állapot állapotdiagramjában a részállapotokhoz azok az állapotátmenetek tartoznak, amelyek az általánosított állapotot nem változtatják meg.

7. Az állapotdiagramban kell lennie legalább egy olyan részállapotnak, amely az általánosított állapot *„entry”* akcióját örökli. (Azaz kell legyen belépési akciója legalább egy részállapotnak.)

8. Ha az általánosított állapot rendelkezik *„exit”* akcióval, akkor az állapotdiagramjában kell lennie legalább egy olyan részállapotnak, amely az általánosított állapot *„exit”* akcióját örökli. (Azaz kell legyen kilépési akciója legalább egy részállapotnak.)

9. Az általánosított állapot megfelelő objektuma az *„entry”* akció hatására létrejön, a megfelelő objektum pedig az állapotdiagram *„exit”* akciójának hatására semmisül meg.

Az általánosított állapot jelölése:



Állapotok aggregációja

Az aggregációval előállított állapot a következő tulajdonságokkal rendelkezik:

1. Az aggregációval létrejövő állapot egymástól független részállapotok egy véges halmaza.

2. Minden részállapothoz állapotdiagram tartozik.

3. A részállapotok lehetnek általánosított állapotok is.

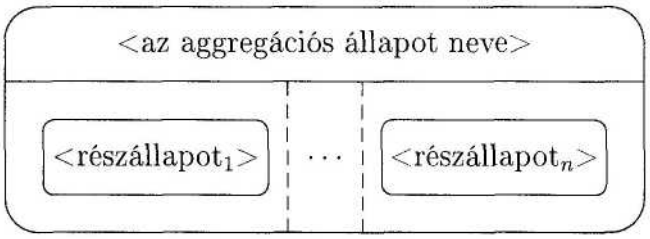
4. Minden részállapothoz tartozó állapotdiagramban kell lennie legalább egy olyan állapotnak, amelybe a részállapot az aggregátum *„entry”* akciójának hatására kerül, azaz örökli ezt az akciót. (Azaz minden részállapotnak kell legyen belépési akciója.)

5. Ha az aggregátum rendelkezik *„exit”* akcióval, akkor minden részállapothoz tartozó állapotdiagramban kell lennie legalább egy olyan állapotnak, amely az aggregátum *„exit”* akcióját örökli. (Azaz minden részállapotnak kell legyen kilépési akciója.)

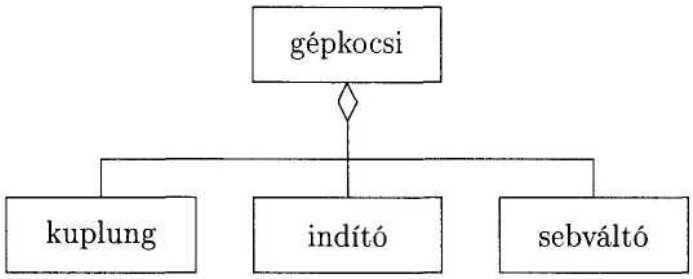
6. Az aggregációs állapot objektuma az aggregációt alkotó részállapotokban egyidejűleg van.

7. Az állapotok aggregációja az állapoton belüli állapotdiagramok közötti párhuzamosság egy megjelenési formája.

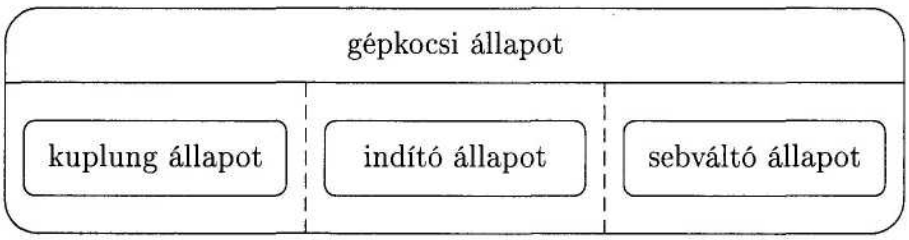
Az aggregációs állapot jelölése:



Egy lehetséges példa a gépkocsi, ami – leegyszerűsítve – álljon a következő egységekből: kuplung, indító és sebességváltó (röviden sebváltó):

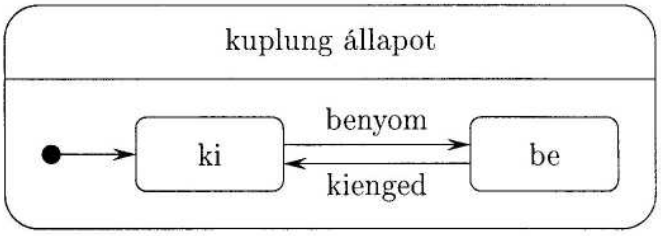


Ekkor a gépkocsi állapotát leírhatjuk ezen három összetevő állapotainak aggregációjaként:

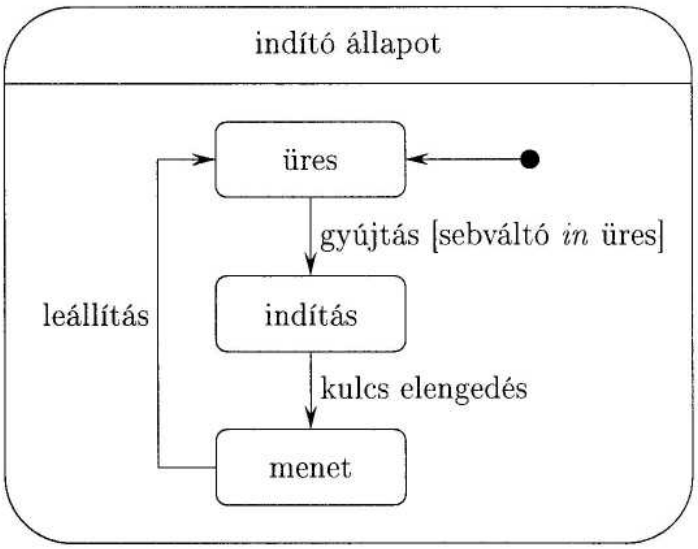


Ezek után csak az aggregációban szereplő általánosított állapotokat kell megadnunk, azaz:

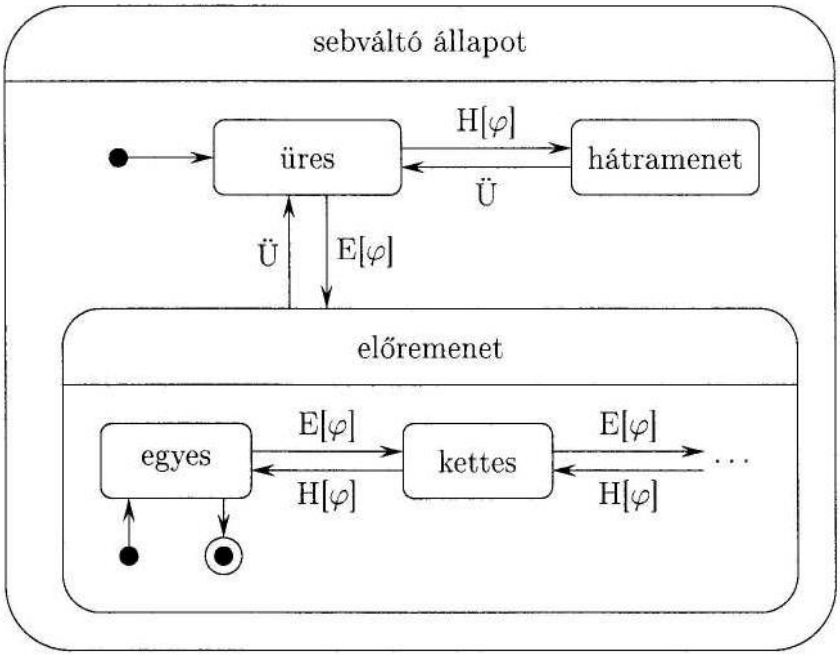
• a kuplung állapotát, amelynek konkrét állapotai a benyomott állapot és a kiengedett állapot:



• az indító állapotát, amelynek konkrét esetei legyenek például az üres, az indítás és a menet állapot:



• és a sebváltó állapotát, amely általában három állapot általánosítása: ezek az üres, a hátramenet és az előremenet állapotok (E = előre kapcsolás, H = hátra (vissza) kapcsolás, Ü = üresbe kapcsolás, φ = kuplung *in* be):



A sebváltó előremenet állapota szintén állapotáltalánosítás, mivel konkrét esetei az egyes, kettes, ... fokozatok. Az egyszerűség érdekében tegyük fel, hogy adott fokozat esetén csak a következő, illetve az előző fokozatba tudunk váltani. Természetesen nincs akadálya annak, hogy az összes előremeneti állapot örökölje az „előremenet" általánosított állapot belépési és kilépési eseményeit, noha az ábrán csak az egyes fokozat esetén tüntettük ezt fel. Sebességet csak benyomott kuplunggal lehet váltani, ezért minden ilyen esemény („E" és „H") feltétele: kuplung *in* be. A sebességváltót viszont fölengedett kuplunggal is üresbe lehet kapcsolni („Ü" esemény).

Az állapotdiagram bonyolultságának kezelésére bevezetett állapotáltalánosítás és állapotaggregáció miatt általánosabban kell megfogalmaznunk az állapot fogalmát, mint ahogy azt a korábbi definíció során tettük. Az ott leírtakat ki kell egészítenünk, és az eddigi rendszeren kívüli állapotokat is bővítenünk kell egy újabbal. Ezen rendszeren kívüli állapotokat a továbbiakban pszeudoállapotoknak nevezzük majd.

Az állapot általános fogalma:

1. Az állapotnak van azonosítója.

2. Esemény vagy akció hatására következik be.

3. Az állapot mindaddig fennmarad, amíg az objektumok attribútumai az állapot invariánsát kielégítik.

4. Az állapot megszűnéséhez esemény(sorozat) kötődik.

5. Az állapot lehet részállapotok általánosítása. Ilyenkor az állapothoz állapotdiagram is kötődik.

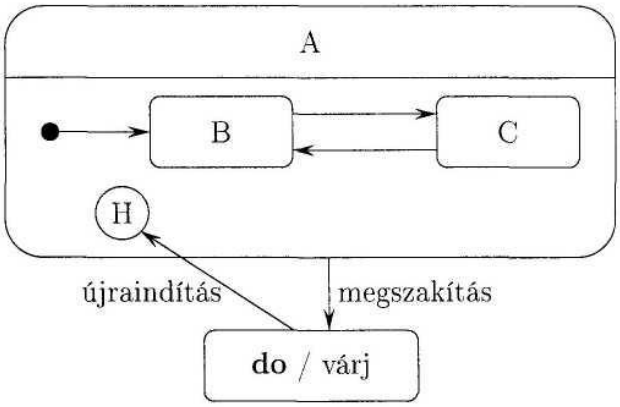
6. Az állapot lehet más állapotok aggregációja.

7. Az állapot lehet pszeudoállapot:

    a. Kezdéskor a külső állapot.

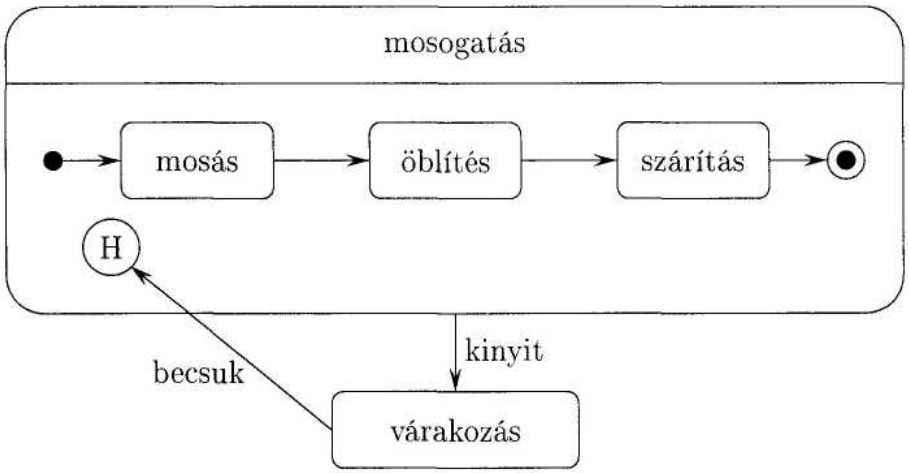
b. Befejezéskor a külső állapot.
c. Hisztorizációs állapot, melyhez hisztorizációs indikátor társul.

Az állapot hisztorizációs indikátorának, illetve a hisztorizációs állapotnak a jelölése:



Ez lehetőséget ad arra, hogy egy adott pillanatban felfüggesszük az állapotokat, majd bizonyos várakozás után újra a felfüggesztés pillanatától folytatódjon az állapotok változása.

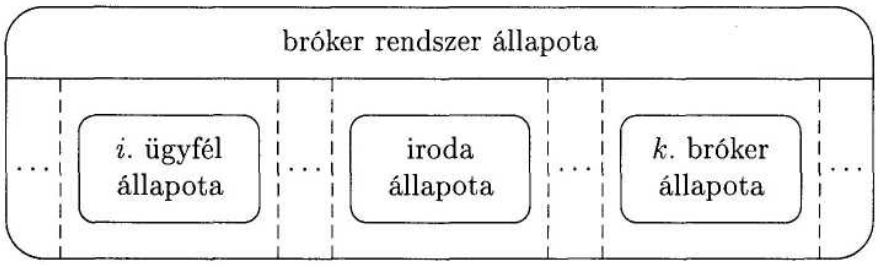
Példaként tekintsük a mosogatógép működését. A mosogatás állapota a mosás, öblítés és szárítás állapotok egymásutánja. Azonban ha kinyitjuk a gép ajtaját, akkor az aktuális tevékenység felfüggesztésre kerül, és ha egy bizonyos idő eltelte után becsukjuk az ajtót, akkor pontosan onnan folytatódik minden, ahol megszakadt:



Példa:
Készítsük el a következő leírás alapján a bróker rendszer állapotdiagramját, felhasználva az állapotaggregációt és -általánosítást. A brókerirodába egymás után érkeznek az ügyfelek, hogy megbízásokat adjanak le részvények vásárlására. A brókerek ezeket a megbízásokat folyamatosan teljesítik. A brókerirodában két alkalmazott veszi át a megbízásokat, és adja tovább a brókereknek. Az alkalmazott egyszerre vagy egy ügyféllel, vagy egy brókerrel foglalkozik. Amíg az alkalmazottól egy megbízást bróker nem vesz át, addig az alkalmazott újabb megbízást nem vesz át ügyféltől.

Megoldás:

A feladatban az ügyfelek és a brókerek mind az iroda szolgáltatásait veszik igénybe. Valójában itt egy termelő-fogyasztó rendszerről van szó, amelyben az iroda a raktár, ahol a megbízások várakoznak arra, hogy a brókerek átvegyék azokat. A raktár két megbízás tárolására alkalmas.
A bróker rendszer állapota ennek megfelelően egy aggregációs állapot, amelyet az ügyfelek állapota, a brókerek állapota és az iroda állapota együttesen határoz meg. Ezen a szinten tehát az alábbi állapotdiagramhoz jutunk:



Az ügyfélnek két állapota van: vagy várakozik, vagy kiszolgálják az irodában, azaz megbízást ad át. Az ügyfél akkor veheti igénybe a szolgáltatást, ha van szabad ügyintéző. A leírás alapján nem lényeges nyilvántartani azt, hogy az ügyfelet melyik alkalmazott szolgálja ki. A megrendelések végrehajtásának sorrendjére sincs semmilyen megkötés. A leírásban arról sincs szó, hogy az ügyfelek milyen sorrendben vehetik igénybe a szolgáltatást, ez tehát véletlenszerűen történik. Ezért képezhetnek egy kvázi végtelen ügyfélfolyamot.

A brókernek is két állapota lehet: vagy várakozik (megbízást teljesít), vagy kiszolgálják (megbízást vesz át). A bróker állapotdiagramja csak abban különbözik az ügyfél állapotdiagramjától, hogy ő akkor veheti igénybe az iroda szolgáltatását, ha van el nem intézett megbízás.

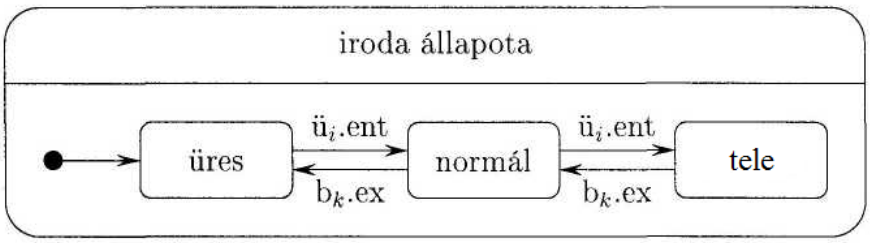
Az irodának három állapota lehet annak megfelelően, hogy hány alkalmazott szabad, azaz hány alkalmazottnál nincs megbízás.

Soroljuk fel ezek után az állapotokat, az állapotátmenetek eseményeit és az átmenetek feltételeit. Egyúttal az ábrázolás céljából vezessünk be ezek számára rövid jelöléseket.

Az iroda állapotai:
- üres: két alkalmazott szabad, azaz egyiküknél sincs megbízás;
- normál: egy alkalmazott szabad;
- tele: egyik alkalmazott sem szabad.

Az iroda állapotának változását előidéző események:
- $ü_i$.ent: egy ügyfél megkezdte a megbízást, eggyel kevesebb szabad hely lesz.
- b_k.ex: egy bróker befejezte a megbízás átvételét, eggyel több szabad hely lesz.

Ennek alapján az iroda állapotdiagramja:



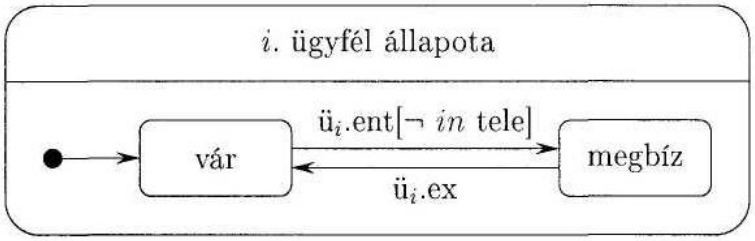
Az ügyfél állapotai:

- vár: az ügyfél arra vár, hogy legyen szabad alkalmazott;
- megbíz: az ügyfél átadja az alkalmazottnak a megbízást.

Az *i.* ügyfél állapotait megváltoztató események és feltételeik:

- $ü_i$.ent[¬ *in* tele]: az *i.* ügyfél akkor adhat át megbízást, ha van szabad alkalmazott;
- $ü_i$.ex: az ügyfél a megbízás átadása után feltétel nélkül távozhat (várakozhat).

Így az *i.* ügyfél állapota:



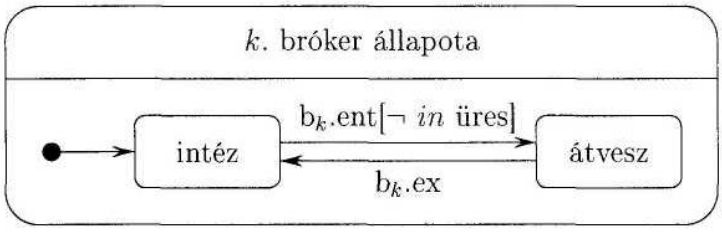
A brókerfolyam állapotai:

- átvesz: megbízást vesz át;
- intéz: megbízást teljesít.

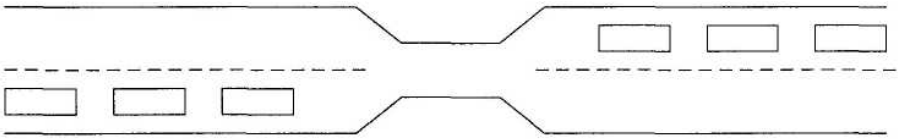
Az állapotokat megváltoztató események:

- b_k.ent[¬ *in* üres]: akkor vehet át megbízást, ha van megbízás az alkalmazottnál;
- b_k.ex: a megbízást feltétel nélkül elviheti intézni.

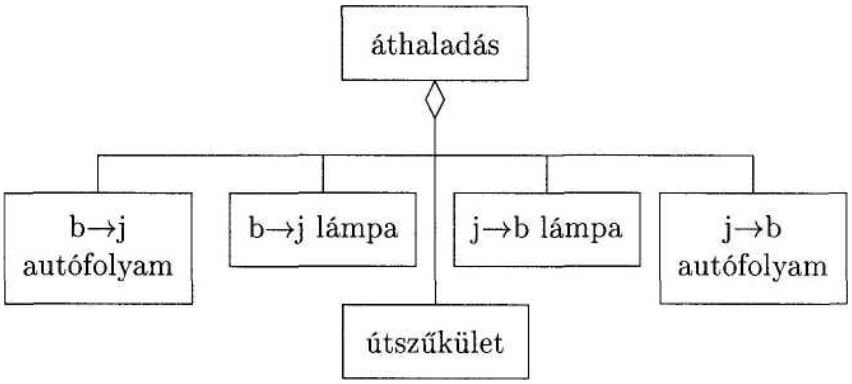
A *k.* bróker állapotdiagramja:



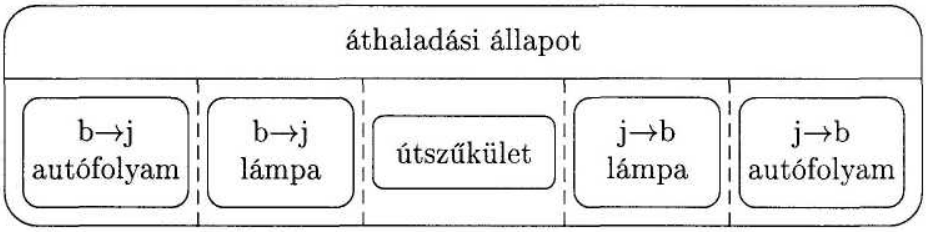
Példa:
Tegyük fel, hogy programot kívánunk készíteni a forgalom lámpával történő vezérlésére egy útszűkületben a következő leírás alapján. Egy útszűkületben csak egy sávban közlekedhetnek a járművek, felváltva jobbról balra, illetve balról jobbra. A forgalmat mindkét oldalon lámpa irányítja. A lámpa zöldre akkor vált, ha az ellenkező irányból a lámpa piros. A zöld színt a ciklusidő lejártakor sárga váltja fel. Sárgáról pirosra akkor vált a lámpa, ha az útszűkületben már nincs jármű. A probléma vázlata:



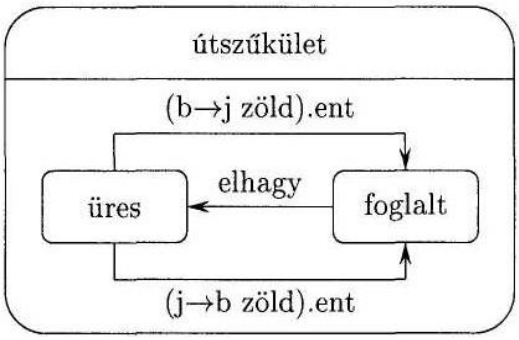
Megoldás:
A leírás alapján az útszűkület olyan erőforrás, amelyet a járművek vagy csak az egyik irányból, vagy csak a másik irányból vehetnek egyszerre igénybe. A két igénybevétel között irányváltás történik, amikor az útszűkületben már nem tartózkodhat jármű. Az útszűkületben történő *áthaladás* tehát a balról jobbra haladó járműfolyamból, a jobbról balra haladó járműfolyamból, a hozzájuk tartozó lámpákból, valamint az útszűkületből mint az igénybe vett erőforrásból áll. Ennek alapján az osztálydiagram:



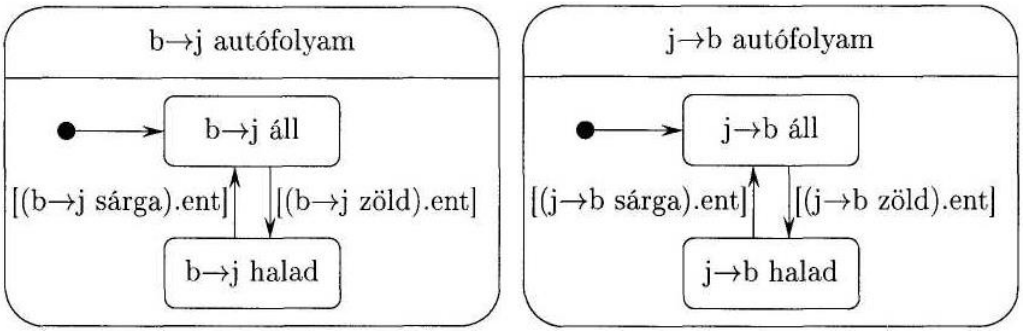
Így az áthaladási állapot (az osztálydiagrammal szinkronban) egy állapotaggregáció lesz:



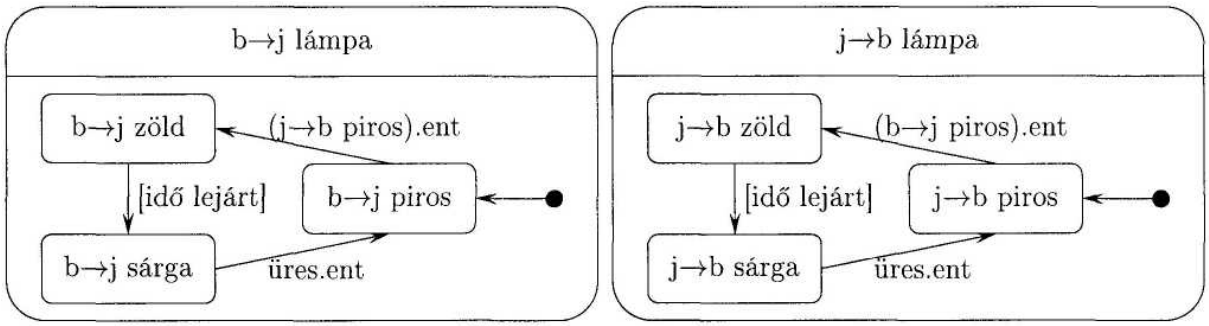
Az útszűkületnek két állapota lehet: tartózkodik benne jármű (foglalt), vagy nincs benne jármű (üres). Ha valamelyik lámpa zöldre vált, akkor foglalttá válik, és üres lesz, ha az utolsó jármű is elhagyja:



A gépkocsifolyamok állapotát a megfelelő lámpa állapota határozza meg. Az autók a lámpa előtt állnak, vagy haladnak, kezdetben állnak. Ha a lámpa zöldre vált, akkor haladnak, és ha a lámpa sárgára vált, akkor megállnak:



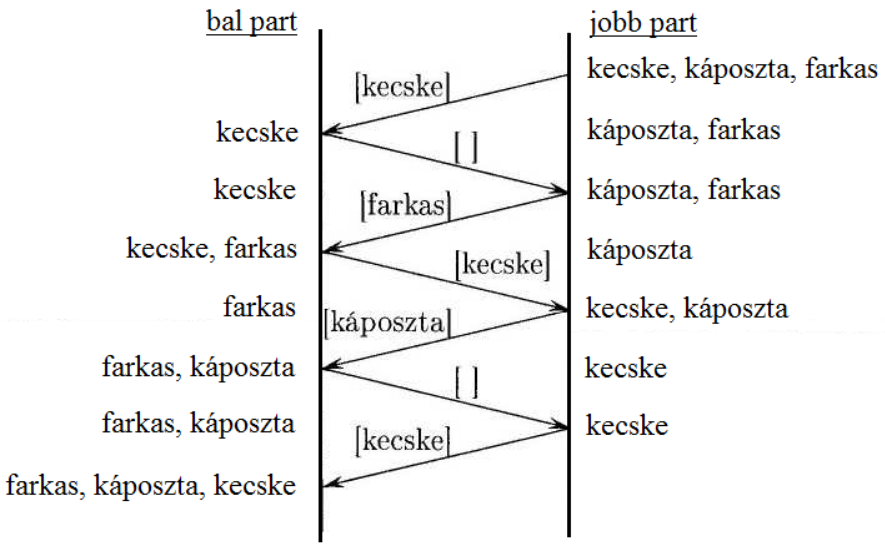
A lámpának három állapota lehet, az aktuális színnek megfelelően. Csak akkor válthat zöldre a lámpa, ha az ellenkező lámpa pirosra váltott. Sárgára az idő letelte után vált, és piros lesz, ha az útszűkület kiürül:



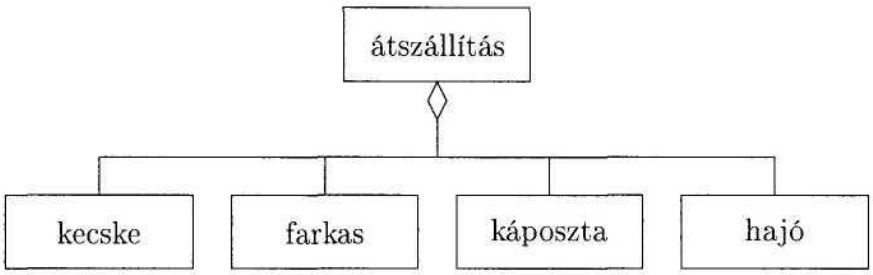
Példa:
Oldjuk meg a kecske, farkas és káposzta folyón történő átszállításának jól ismert problémáját. A szabályok a következők:

1. A csónakban egyszerre csak egy utast (tárgyat) tud szállítani a hajós.
2. A kecske a káposztával nem maradhat azonos parton, ha a hajós nincs jelen, mert a kecske megeszi a káposztát.
3. A farkas nem maradhat azonos parton a kecskével, ha a hajós nincs jelen, mert a farkas megeszi a kecskét.

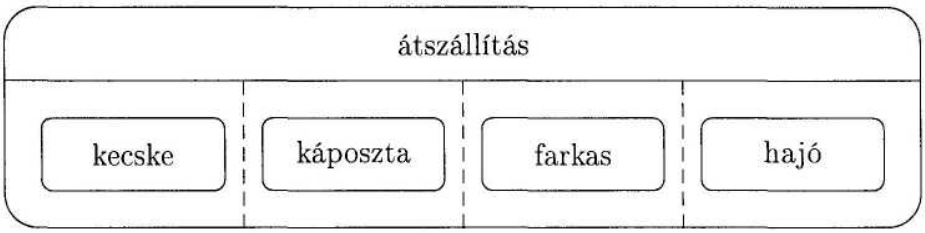
Megoldás:
A probléma egy lehetséges megoldása:



A leírás alapján az objektumok, illetve az osztályok a következők lesznek: átszállítás, kecske, káposzta, farkas, hajó. Az átszállítás az a rendszer, amelyet a kecske, káposzta, farkas és hajó objektumok alkotnak. Ez a rendszer tehát egy aggregátum, osztálydiagramja:



Az átszállítás állapotát a kecske, a káposzta, a farkas és a hajó állapota együttesen határozzák meg. Ezért az átszállítás állapota a felsorolt állapotok aggregációja:



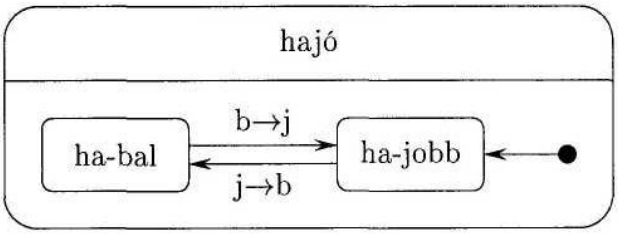
A hajó állapota: A hajó a folyó két partja között ingázik. A probléma megoldása szempontjából az a lényeges, hogy melyik parton kötött ki.
Ennek megfelelően az állapotok:

- hajó a bal parton kikötött : ha-bal,
- hajó a jobb parton kikötött : ha-jobb.

Az állapotot megváltoztató események pedig:

- átkelés balról-jobbra : b→j,
- átkelés jobbról-balra : j→b.

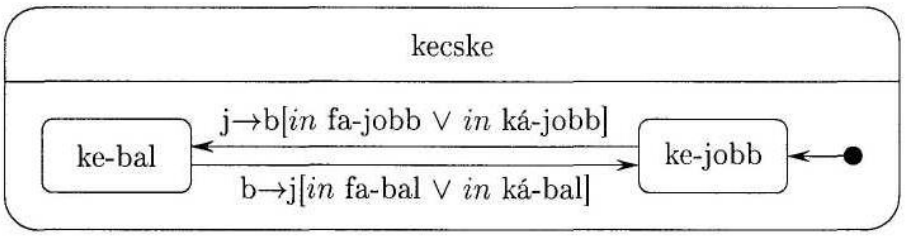
Az átkelés valójában időben elhúzódó esemény, azonban a modellben ezt egyszerűsíthetjük, és akcióként is tekinthetünk rá. Így a hajó állapotai:



A kecske állapota: A kecskének is két állapota van, attól függően, hogy melyik parton tartózkodik. Az állapotokhoz invariánsokat rendelünk, amelyekkel kifejezzük, hogy a kecske csak akkor tartózkodhat az egyik parton, ha a hajó ott van, vagy nincs azon a parton sem a farkas, sem a káposzta. Az invariánsokkal kifejezzük, hogy a hajót akkor használja a kecske, ha a farkas mellől kell menekülnie, vagy ha a káposzta kerül felügyelet nélkül veszélybe a jelenléte miatt. Az állapotok és invariánsaik a következők.

- A kecske a bal parton van: ke-bal,
  invariánsa: *in* ha-bal V *(in* ká-jobb ^ *in* fa-jobb).
- A kecske a jobb parton van: ke-jobb,
  invariánsa: *in* ha-jobb V *(in* ká-bal ^ *in* fa-bal).

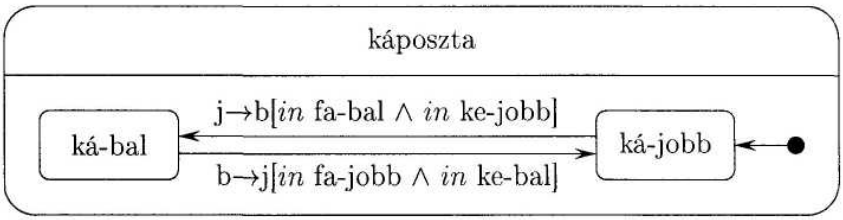
Az állapotok közötti átmenetet az átkelések adják, az egyetlen megszorítás, hogy az invariánsoknak fenn kell állniuk. Ezt biztosíthatjuk, ha a kecskét mindig átszállítjuk, amikor a farkas vagy a káposzta vele azonos parton van:



A káposzta állapota: A kecskéhez hasonlóan a káposztának is két állapota van, amelyeket invariánsokkal jellemezhetünk. Az invariánsok kifejezik, hogy a kecske és a káposzta csak akkor lehet ugyanazon a parton, ha a hajó is ott van:

- A káposzta a bal parton van: ká-bal,
  invariánsa: *in* ha-bal V *in* ke-jobb.
- A káposzta a jobb parton van: ká-jobb,
  invariánsa: *in* ha-jobb V *in* ke-bal.

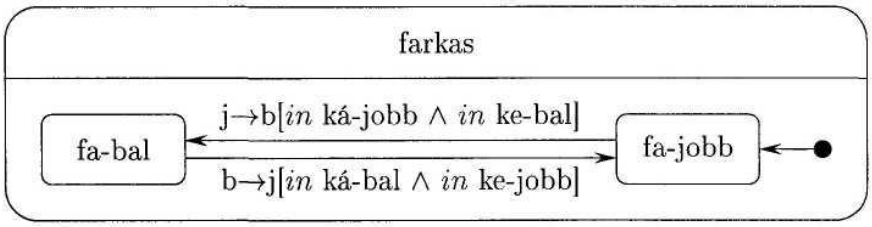
Az állapotok közötti átmeneteket az átkelések adják, amelyekhez feltételt rendelve biztosíthatjuk, hogy a kecske és a káposzta állapotainak invariánsa is fennálljon. Ezért a káposztát akkor kell átszállítani, ha a kecske azonos parton van, és nincs azon a parton a farkas:



A farkas állapota: Az eddigiekhez hasonlóan eljárva meghatározhatjuk az állapotokat és invariánsaikat:

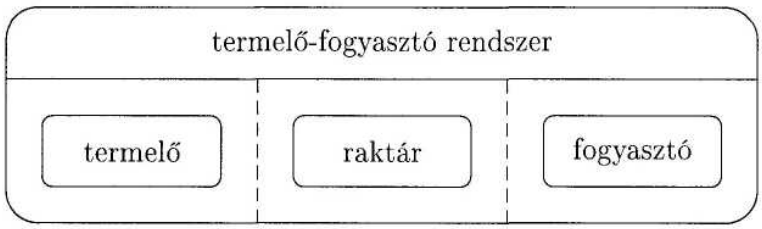
• A farkas a bal parton van: fa-bal,
invariánsa: *in* ha-bal V *in* ke-jobb.
• A farkas a jobb parton van: fa-jobb,
invariánsa: *in* ha-jobb V *in* ke-bal.

Az invariánsok alapján kapjuk, hogy farkast át kell vinni a kecskéhez a káposzta mellől:



Példa:
Készítsük el a következő leírás alapján a termelő-fogyasztó rendszer állapotdiagramját. A termelő-fogyasztó rendszer raktárból, termelőből és fogyasztóból áll. A termelő az általa előállított árut akkor helyezheti el a raktárban, ha van üres hely. A raktárban *n* darab üres hely van. A fogyasztó akkor szállíthat ki a raktárból árut, ha a raktárban van áru.

Megoldás:
A termelő-fogyasztó rendszer állapotát a termelő állapota, a fogyasztó állapota és a raktár állapota együttesen határozza meg. A termelő-fogyasztó rendszer állapota tehát ezek aggregációja:



A raktár állapota egy általános állapot. Ennek konkrét eseteit a hozzáférés szempontjából a következőképpen határozhatjuk meg:

1. Üres állapot, amikor a raktárban az üres helyek száma (*hely*) megegyezik a raktár kapacitásával ($n > 0$):

$$\textit{üres}[\textit{hely} = n] .$$

2. Normál állapot, amikor a raktár már nem üres, és még nincs tele:

$$\textit{normál}[0 < \textit{hely} < n] .$$

3. Tele állapot, amikor az üres helyek száma 0:

$$tele[hely = 0] .$$

A raktárban állapotváltozásokat a következő események okoznak:

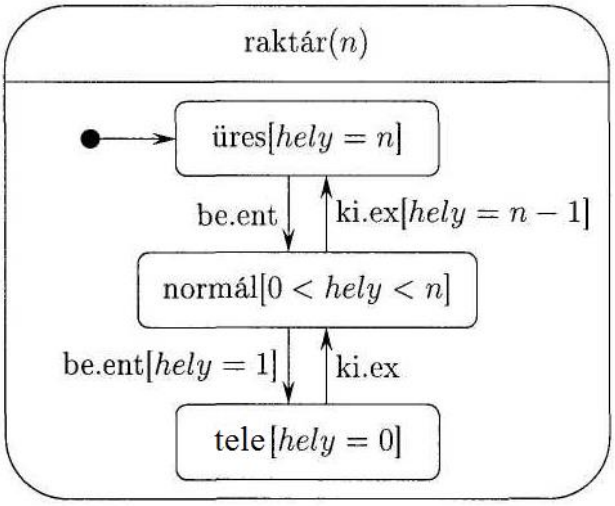
1. Áru elhelyezése az üres raktárban, aminek eredményeként a normál állapot jön létre $n > 1$ esetén. Legyen ez az akció: *be.ent*.

2. Áru elhelyezése a normál állapotú raktárban, aminek eredményeként az megtelik (üres helyek száma 0 lesz): $be.ent[hely = 1]$.

3. Áru kivétele a teli raktárból, aminek eredményeként a normál állapot jön létre $n > 1$ esetén. Legyen ez: *ki.ex*.

4. Áru kivétele a normál állapotú raktárból, aminek eredményeként az kiürül (üres helyek száma n lesz): $ki.ex[hely = n - 1]$.

Ennek eredményéül a raktár állapotdiagramja:

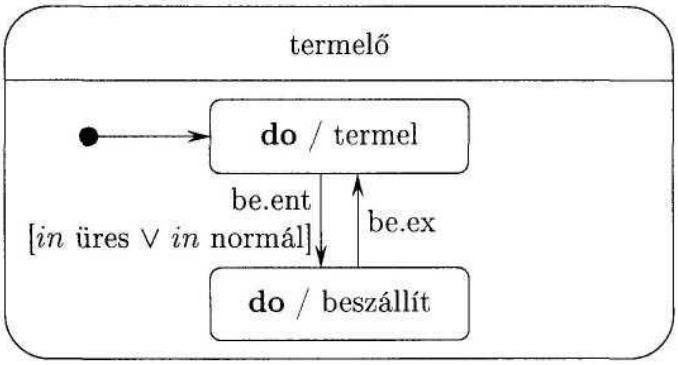


A termelő állapotai:

1. Az az állapot, amikor a termelő az árut előállítja. Legyen ez: do / termel. Ebbe az állapotba a termelő feltétel nélkül átmehet.

2. Az az állapot, amikor a termelő az árut elhelyezi a raktárba. Legyen ez: do / beszállít. Ebbe az állapotba akkor mehet át a termelő, ha van üres hely a raktárban, azaz ha a raktár állapota *üres* vagy *normál*. A beszállítás eseményéhez tartozó belépési és kilépési akciókra röviden „be.ent” és „be.ex” néven hivatkozunk.

A termelő állapota tehát az következő diagram lesz:



A fogyasztó állapotai:

1. Az az állapot, amikor az árut kiszállítja a raktárból. Legyen ez: do / kiszállít. Ebbe az állapotba akkor léphet be, ha van áru a raktárban, azaz: ki.ent[*in* normál V *in* tele]. (A kiszállít eseményre a belépési és kilépési fázisban röviden „ki" néven hivatkozunk.)

2. A kiszállított áru feldolgozásának állapota: do / fogyaszt. Ebbe az állapotba feltétel nélkül átmehet az áruval.

A fogyasztó állapotdiagramja: