4. gyakorlat – Modellek fejlesztése – Megoldások

Figyelem: Jelen anyag belső használatra készült megoldási útmutató, melyet a ZH felkészülés segítése érdekében publikáltunk. A feladatok részletesebb megoldása magyarázattal gyakorlaton hangzott el.

1. feladat

Elakadásjelző háromszögeket előállító gyárunkat adatfolyamhálóval modellezzük. A háló kezdetben két csomópontot tartalmaz. Az első csomópont egy gép, amely fényvisszaverő oldallapokat állít elő, és a futószalagra helyezi őket. A második csomópont az összeszerelő gép, amely a futószalagról felveszi a lapokat; ezen kívül időnként egy összeszerelt háromszöget bocsájt ki az egész háló kimenetén.

a. Készítsük el a feladat adatfolyamháló modelljét. Szorítkozzunk állapotgép jellegű csomópontokra.
b. Finomítsuk a modellt a következőképp: az első gép időnként deformált oldallapokat gyárt.
c. Finomítsuk tovább a modellt a következőképp: az összeszerelő gép az eredeti funkcionalitás elé kapcsolva tartalmaz egy bevizsgáló berendezést is, amely képes kidobni a deformált lapokat (az ép lapokat továbbengedve).
d. Végül finomítsuk tovább a modellt a következőképp: az összeszerelő gép (a selejtes lapok kiszűrése után) mindig bevár három fényvisszaverő lapot, és belőlük szerel össze egy háromszöget.

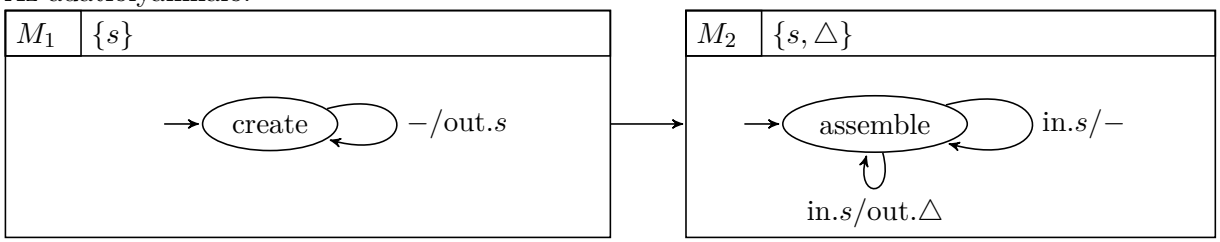
Megoldás

Azt érdemes pluszban a b-c-d lépésekben alaposan végiggondolni, hogy mi is a finomítás következménye:

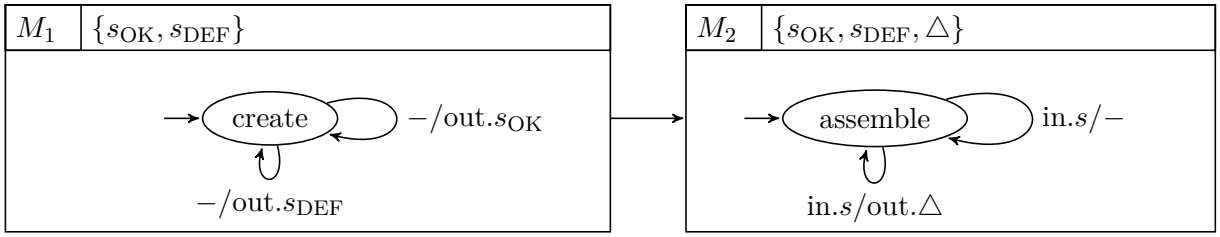
- A finomított modell "ugyanúgy" működik, mint az absztrakt; ha megfigyeljük a működését, de olyan szemüvegen keresztül, amely "röptében" visszacsinálja az absztrakciót (pl. összevonja a szétválasztott tokenfajtákat), akkor nem tudjuk megkülönböztetni az absztrakt modell viselkedésétől. Ezért helyes az absztrakciós viszony a két modell között.
- A finomított modell több információt tartalmaz, ugyanis finomítás közben választási lehetőségeink vannak. Pl. a (d) résznél sokféleképpen lehet az állapotok szétvágása után az állapotátmeneti szabályokat úgy felvenni, hogy megmaradjon az absztrakciós viszony, de ezek közül csak egy az, amely a feladatban leírtaknak megfelelően működik; ennek az egy megoldásnak a kiválasztása teszi bele az információtöbbletet a finomításba.

Ezeket a tanulságokat persze a lenti feladatmegoldás közben lehet felvezetni.
Az egyes állapotgépek be- és kimenetei lapok (s) és háromszögek ($\triangle$) lehetnek.

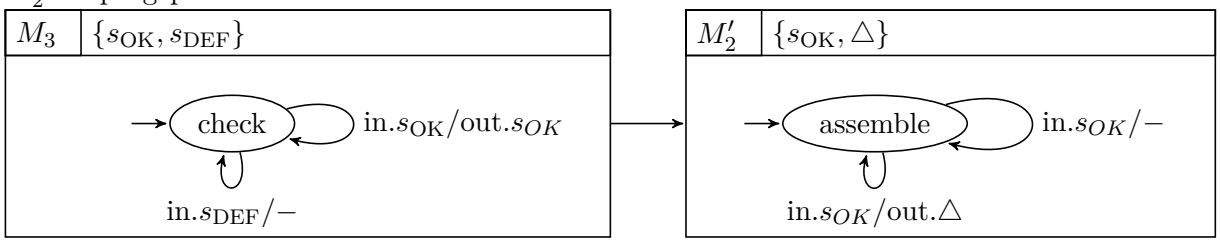
a. Az adatfolyamháló:



Megjegyzés a megoldásokhoz: a szövegnek jobban megfelelne, ha M_2 nem rögtön a lap felvételekor adna outputot, hanem csak úgy néha, spontán átmenetkéni (TODO: átrajzolni).
b. Ehhez végezzünk **tokenfinomítást**: a s helyett vegyünk fel egy s_{OK} és egy s_{DEF} tokent.

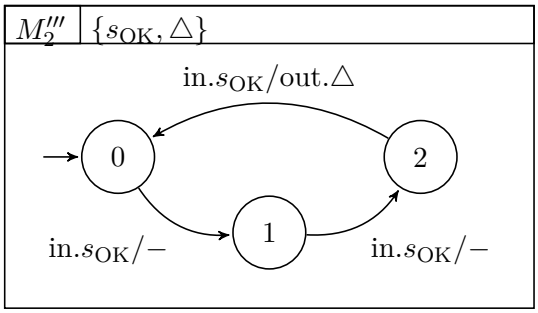


c. Ehhez végezzünk **struktúrafinomítást**: az M_2 állapotgép helyett vegyünk fel egy M_3 és egy M_2' állapotgépet.



A feladat megoldható tokenfinomítással is: ebben az esetben az M_2 állapotgépet egészítsük ki egy in.s_{DEF} önhurokkal.

d. Ehhez végezzünk **állapotfinomítást**.



Jó megoldás az is, ha felveszünk plusz egy olyan állapotot, ahol 3 oldallap van a rendszerben. Ennek a bemenő tranzíciója a 2 állapotból $\text{in}.s_{\text{OK}}/-$, a kimenő tranzíciója a 0 állapotba $-/\text{out}.\triangle$.

2. feladat

Modellezzük egy üzleti folyamat tipikus végrehajtásának aspektusait adatfolyamhálóval:

a. Írjuk le egy XOR gateway pár (decision-merge) működését.
b. Modellezzük egy AND gateway (fork-join) működését.
c. Mutassuk be két folyamatlépés közti adat- és vezérlési függőséget.
d. Modellezzük egy folyamatlépés erőforrásfoglalását, ha
   i. egy folyamatlépés egy erőforrást használ,
   ii. egy lépés két erőforrást használ.

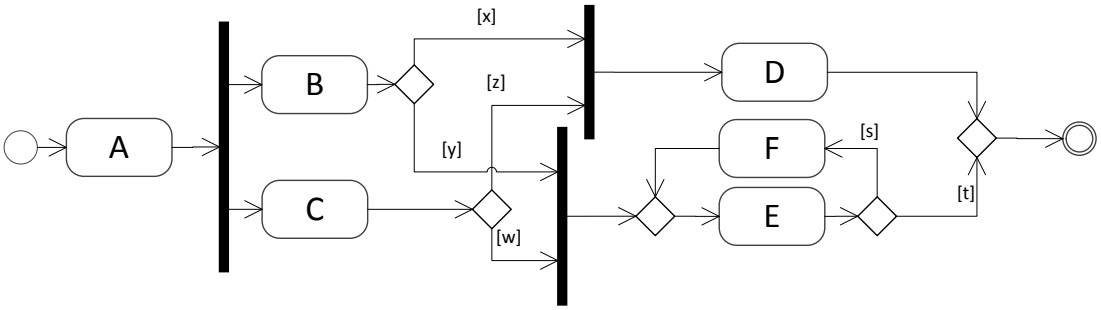
Milyen lehetőségeink vannak annak modellezésére, hogy egy erőforrásból több példány áll rendelkezésre? Mi ezeknek az előnye/hátránya?

Megoldás

TODO

3. feladat

Ellenőrizzük az alábbi folyamatmodellt.



1. ábra.

a. Milyen feltételek mellett teljesen (ellentmondásmentesen) specifikált a folyamat?
b. Milyen feltételek mellett determinisztikus is a folyamat?
c. Milyen feltételek mellett holtpontmentes is a folyamat?
d. Milyen további feltételek mellett termináló a folyamat?
e. Jólstrukturált-e a folyamat? Ha nem, hogyan lehetne azzá tenni? Segít-e ez a problémákon?

Megoldás

a. Az állapotgépekkel analóg módon a folyamat akkor teljesen specifikált, ha minden elágazáshoz érkezéskor (decision) a kimenő élek őrfeltételei közül legalább az egyik igaz – magyarul mindig járható legalább az egyik kimenő él. Ehhez elégséges feltétel, hogy teljes feltételrendszert alkossanak az őrfeltételek, de igazából elég annyit megkövetelni, hogy feltételesen teljes rendszer legyen, tehát ha odakerülhet a vezérlés, akkor álljon fenn, hogy legalább az egyik kimenő igaz (a harmadik decisionnél ez számít).

   - Következésképp:
     - $x \vee y$
     - $z \vee w$
     - $w \wedge y \implies s \vee t$ (a jobb oldal a ciklus bárhány végrehajtása után igaz).

b. A folyamat akkor determinisztikus, ha minden elágazáshoz érkezéskor (decision) a kimenő élek őrfeltételei közül legfeljebb az egyik igaz – magyarul mindig csak az egyik kimenő él járható. Ehhez elégséges feltétel, hogy kizárólagos feltételrendszert alkossanak az őrfeltételek, de igazából elég annyit megkövetelni, hogy feltételesen kizárólagos rendszer legyen, tehát ha odakerülhet a vezérlés, akkor álljon fenn, hogy legfeljebb az egyik kimenő igaz (a harmadik decisionnél ez számít).

   - Következésképp:
     - $\neg(x \wedge y)$
     - $\neg(z \wedge w)$
     - $w \wedge y \implies \neg(s \wedge t)$ (a jobb oldal a ciklus bárhány végrehajtása után igaz)
   - Az előzővel összesítve némileg egyszerűsíthetjük az első két kritériumot:
     - $y = \neg x$
     - $w = \neg z$

c. Holtpont (deadlock) = örök várakozás; itt úgy fordulhat elő, hogy a fork után az egyik ág a fenti, a másik ág a lenti joint választja, és örökké várnak egymásra. Baj van, ha az egyik joinba csak az egyik ág fut be.

   - Következésképp:

  - $x = z$
  - $y = w$

d. Először is nyilván feltesszük, hogy maguk az elemi tevékenységek terminálnak – ha nem így lenne, akkor sem a folyamatmodell a nemterminálás forrása. A folyamatmodellben probléma lehet, ha a join örökké vár – de a holtpontot már kizártuk. Utolsó lehetőségként marad a livelock, vagyis a végtelen ciklus. Ebben akkor ragadunk bele, ha ráfutunk, és a kilépési feltétel sose válik igazzá.

   - Következésképp:
     - Ha $\neg x$ (ilyenkor y és w igaz), akkor előbb-utóbb t-nek igazzá kell válnia. Érdekesség: ahogy a fenti állításokat, úgy ezt is le lehet írni logikai formulaként (van "előbb-utóbb" szimbólum stb.), de az ehhez szükséges ún. temporális logikákat nem tanultuk.
     - $\neg x \implies \mathrm{F}t$

e. A tanult módszerrel megvizsgálva kiderül, hogy nem jólstrukturált. Azzá lehet tenni, ha B és C után van egy join, és utána egyetlen decision. Ez a deadlockot automatikusan kiküszöböli, a többi hibalehetőséget viszont nem – legfeljebb a modell átláthatóbbá tételével segít –, így pl. livelock megmaradhat, determinizmust nem garantál.

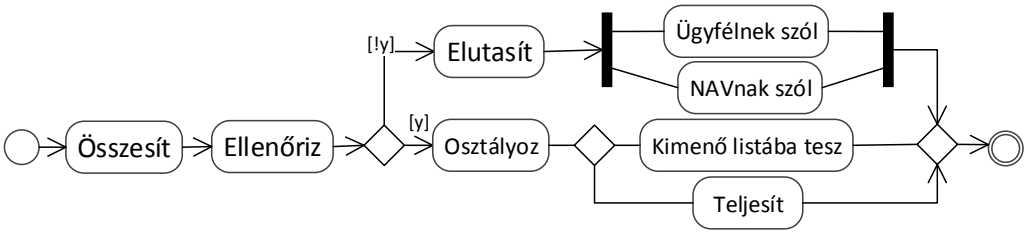
4. feladat

Egy bank minden éjszaka napi zárást végez. A zárás kezdetén összegyűjti az aznapra begyűjtött átutalási megbízásokat, majd egy ellenőrzési lépés következik, amikor az egyes utalásokat "átvilágítják" a pénzmosás elleni küzdelem jegyében. Amelyik tranzakciót a vizsgálat során gyanúsnak ítélik, azt elutasítja a bank. Ilyen esetekben egyrészt az ügyfél hibajelzést kap, másrészt automatikus értesítést küld a bank a NAV felé. Az ellenőrzés végeztével a bank osztályozza a rendben talált tranzakciókat aszerint, hogy az utalás kedvezményezett számláját melyik banknál vezetik. Az osztályozás után a bankon belüli tranzakciók teljesítése megkezdődhet; szintén az osztályozás után (de a belső tranzakciók teljesítésével átlapolva) a más bankokba irányuló utalásokat a bank az ún. GIRO szerveren keresztül eljuttatja az érintett idegen bankoknak. Szintén az osztályozás után, az előbbi két tevékenységgel átlapolva a bank fogadja a GIRO-n keresztül érkező, más bankokból kezdeményezett utalásokat. Egyszerűsített rendszerünkben a GIRO szerver egyszerre csak egy bank utaláslistáját tudja feldolgozni és továbbítani a többi banknak, a többi banknak addig türelmet kérő választ ad. Ha egy bank épp emiatt nem tudja a GIRO rendszerbe küldeni a saját utaláslistáját, t idő múlva megpróbálja azt újra elküldeni.

a. Készítsen folyamatmodellt (az előadáson megismert formalizmussal) a bankban kezdeményezett egyes tranzakciók életútjáról a napi zárás során!
b. Készítsen folyamatmodellt (az előadáson megismert formalizmussal) arról, hogy a bank a tranzakciói összességére elvégzi a napi zárást! Ügyeljen arra, hogy ebben a folyamatban az előzőtől részben eltérő tevékenységek és vezérlési elemek jelenhetnek meg.
c. Készítsen adatfolyamhálót (az előadáson bemutatottak szerint állapotgép csomópontokkal és pont-pont FIFO csatornákkal) a banki zárás működéséről. A modellben egyrészt jelenjen meg egyetlen bank és a GIRO egymás közötti kommunikációja (beleértve azt is, amikor a GIRO más bankoktól kapott utaláslistát továbbít a vizsgált banknak), másrészt minden említett külső szereplővel (pl. NAV) folytatott kommunikációjuk is. Az adatfolyamháló csomópontjainak belső működését elég annyira részletezni, hogy kiderüljön, hogy az elvégzendő tevékenységeik bizonyos ideig eltarthatnak, hogy bizonyos üzenetküldések sorrendje meghatározott, és hogy a GIRO egyszerre egy bank utaláslistájával foglalkozik (a többi addig vár).

Megoldás

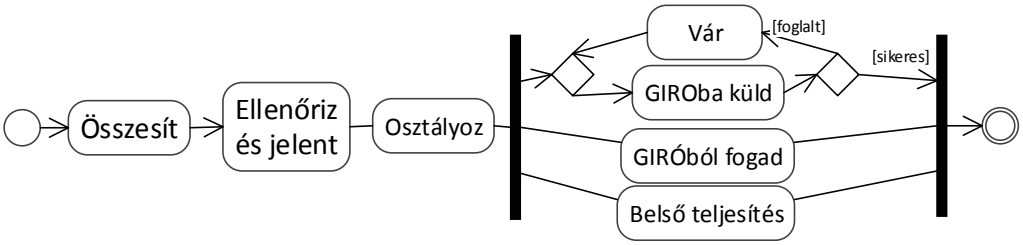
a. Egy lehetséges megoldás a lenti ábra, de van némi szabadság, pl. összesítés vagy kimenő listába tevés lehagyható, a fork-join lehet szekvencia, az ellenőrzés és osztályozás összeolvashat a decisionnel, lehet 2 merge egymás után a jólstrukturáltság miatt, stb.



2. ábra.

- Érvényes folyamatmodell, többé-kevésbé elfogadható szintaxissal (1 pont)
- Felfogta, hogy a tranzakció szemszögéből nézzük, nem az egész bankéból (1pont)
- A vezérlési struktúra jó taszkokkal van kitöltve (1 pont)
- Döntések logikája, sorrendje jó, őrfeltételek is (amik most nálam technikai okból lemaradtak) (1pont)

b. Egy lehetséges megoldás a lenti ábra, de most is van némi szabadság, pl. az ellenőrzés belseje ki van-e fejtve.



3. ábra.

- Érvényes folyamatmodell, többé-kevésbé elfogadható szintaxissal (0,5 pont) – erre azért nem adnék másodszor is egy teljes pontot
- Felfogta, hogy a bank szemszögéből nézzük, nem az egyes tranzakciókéból (1pont)
- A vezérlési struktúra jó taszkokkal van kitöltve (0,5 pont)
- Fork-join logika helyes (1 pont)
- Ciklus jó (1 pont)

c. Pontozási javaslatok:

- Érvényes adatfolyammodell, állapotgép csomópontokkal és köztük pont-pont csatornákkal, többé-kevésbé elfogadható szintaxissal (1 pont)
- Csomópontok azonosítása: GIRO, egy bank (0,5 pont)
- Belső kommunikáció csatornái és tokenjei: bank → GIRO irányban kimenő utaláslista, GIRO → bank irányban "foglalt" és bejövő utalás/lista, utóbbi lehet két külön vagy egyetlen csatorna (0,5 pont)
- Külső kommunikációs csatornák és tokenek: ügyfél és NAV számára pénzmosási jelentés a bankból (ha az egyik lemarad, nem tragédia), GIRO → külső bank és GIRO ← külső bank a vizsgált bankkal egyező interfésszel (0,5 pont)
- GIRO belső logika: alapból tétlen; ha listát kap (bármelyik banktól), elkezdi feldolgozni; eközben küldhet listát bármelyik banknak, majd visszatér a tétlen állapotba. Miközben tétlen, foglaltsági jelzéssel válaszol minden lista beérkezésekor. (1 pont)
- Bank belső logika: alapból küldhet spontán jelzést az ügyfélnek / NAV-nak, de utána állapotot vált. Innentől tud listákat fogadni a GIRO-tól (mindegyiket feldolgozza egy ideig, majd visszamegy), valamint egyszer spontán küld kimenő listát a GIRO-nak (ha foglaltsági

jelzés jön, akkor visszamegy, hogy később újra megpróbálhassa), amely után már csak fogad utaláslistát (ha nem jön foglaltságjelzés, nem küld újra). Nem érdekes, mi történik a végén, befejeződik-e. (1,5 pont)