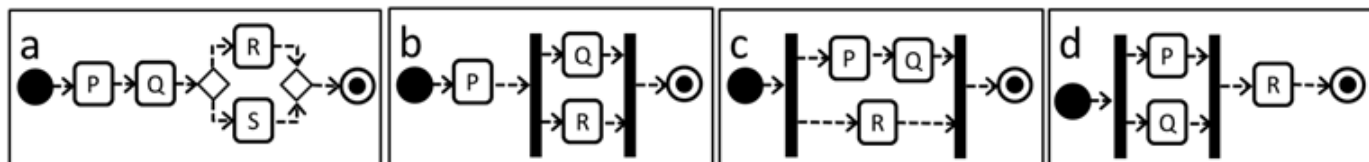


3. gyakorlat – Folyamatmodellek, kooperáló viselkedésmodellek

1. feladat

Egy folyamat végrehajtása során az összes lépést megfigyeltük. A következő eseménysor bekövetkeztét észleltük: *Folyamat indul, P elkezdődik, P befejeződik, Q elkezdődik, R elkezdődik, Q befejeződik, R befejeződik, Folyamat befejeződik.*



Az a, b, c, d folyamatmodellek közül melyek lehetnek helyes modelljei a rendszernek?

2. feladat

Felhő alapú adattárolást modellezünk (ld. Dropbox, Google Drive, Tresorit), egyetlen állományra szorítkozva. Az állománynak a szerveren és a kliensnél (pl. laptop) is elérhető egy-egy replikája, kezdetben azonos tartalommal. A fájl módosításai *szinkronizálás* során továbbítódnak a példányok között. Ha szinkronizáció előtt mindkét példányt módosítják, akkor *ütközés* lép fel, amelyet a felhasználónak kell feloldania a kliensen.

- Modellezzük először a laptop kliens (részleges) működését állapotgéppel! A kliens kezdetben *szinkron* állapotú (a lokális fájl másolat egyezik azzal, ami a szerveren a legutóbbi szinkronizációkor volt/lett), ám *írás* input hatására a *piszkos* állapotba kerül (és további *írás* hatására is ottmarad). Az *elvet* input hatására tetszőleges állapotból újra *szinkron* állapotba kerül.
- A szerver lehetséges állapotai (csupán az adott laptop klienssel való szinkronizációt vizsgálva) a *szinkron* és a *frissült*. Előfordulhat, hogy a megfelelő írási jog birtokában egy másik felhasználó (vagy ugyanazon felhasználó egy másik kliens, pl. a telefonja segítségével) frissíti a szerveren található állományt.
- Ha a szerver *szinkron* állapotban van, akkor a kliens a *szinkronizál* input hatására feltölti az esetleges lokális módosításokat a szerverre, és szintén *szinkron* állapotba kerül. A kliens és a szerver közben információt cserélnek – hol kooperál a két automata?
- Ha a szerver *frissült* állapotban van, akkor a kliensnek adott *szinkronizál* input hatására a szerver *szinkron* állapotba kerül; a kliens pedig *szinkron* állapotból nem mozdul, de *piszkos* állapotból *ütközés* állapotba megy. Mit jelent ez? Mi történjen az ütközés állapotban? Hol kooperál a két automata?
- A kliens időnként magától is szinkronizál a szerverrel, felhasználói input nélkül. Mit jelent ez? Hol kooperál a két automata?
- (Otthoni feladat) Teljes összetett állapottér kifejtése a vegyes szorzatban résztvevő két automata alapján.

3. feladat

Tekintsük az alábbi C nyelvű függvényt.

```
unsigned long long f(int n) {
    if (n <= 0) {
        return 0;
    } else if (n == 1) {
        return 1;
    } else {
        unsigned long long a = f(n - 1);
        unsigned long long b = f(n - 2);
        return a + b;
    }
}
```

- a. Milyen vezérlési folyamatot határoz meg a függvény?
- b. Mi biztosítja azt, hogy a függvény előbb-utóbb terminál?
- c. Azonosítsuk az adatfüggőségeket (adatáramlást) a tevékenységek között!
- d. (Kiegészítő feladat) Ha a programozási nyelv vagy a futtatókörnyezet megengedi, hol van lehetőség párhuzamosításra?

4. feladat

Egy nagy szoftveralapítvány kódtára (pl. Git) számos nyílt forráskódú szoftver fejlesztésének ad otthont. A megbízható belső fejlesztőkön kívül külsősök is gyakran küldenek be hibajavításokat vagy újonnan megvalósított képességeket. Oda kell figyelni arra, hogy a kiadott szoftverben csak jogszerűen (pl. munkaadó beleegyezésével) bekerült forráskód szerepeljen.

- a) Belső fejlesztők közvetlenül írhatnak a kódtár adott projekt részére fenntartott területére. Külsős fejlesztőknek először átvizsgálásra (*code review*) be kell nyújtaniuk a kódjukat; ezután egy belső fejlesztőnek ellenőriznie kell azt, és utána vagy elutasítania, vagy elfogadnia. Az elfogadást követően az alapítvány jogi osztálya egy külön adminisztratív eljárásban tisztázza a változtatások szellemi tulajdonának jogállását, és csak ennek sikeres lezárása után olvashatja be a belső fejlesztő a kódot. Frissen indított, első hivatalos kiadásuk előtt álló projekteknél itt tesznek egy kivételt: az elfogadott külső hozzájárulás kódtárba beolvasztásával nem kell megvárni ezt az adminisztratív eljárást. Készítsünk folyamatmodellt az itt leírt tevékenységekből!
- b) A szoftver fejlesztési projektje abból áll, hogy újabb és újabb módosításokat végeznek a forráskódon, amíg a projekt vezetése úgy nem látja, hogy a szoftver kellően stabil egy hivatalos kiadáshoz. Amikor eljött ez a pont, akkor közlést tesznek egy új stabil verziót a szoftverből, majd ismét a fejlesztésen a sor stb. Készítsünk folyamatmodellt az itt leírt tevékenységekből!
- c) Milyen viszonyban állnak egymással a fenti részfeladatokban elkészített folyamatmodellek?
- d) Ellenőrizzük, hogy jól strukturáltak-e a folyamataink!

5. feladat

Az első feladatban modellezett klijent és szervert alaposabb vizsgálatnak vetjük alá, figyelembe véve, hogy hálózaton keresztül, üzenetek segítségével tartják a kapcsolatot. Ami az előző, absztrakt modellben a két automata egyidejű (atomi) közös állapotátmenetként jelent meg, az valójában időbeli kiterjedéssel rendelkező, üzenetek küldésével és fogadásával megvalósuló kommunikációs tevékenység.

- a. Készítsünk adatfolyamhálót, amelynek a kliens és a szerver a csomópontjai! Az adatfolyamhálóban FIFO, kapacitáskorlát nélküli, pont-pont csatornákat használunk (az előadáson bevezetett módon). Határozzuk meg az adatfolyamháló bemenetén, illetve a két csomópont közötti interfészen érvényes tokeneket és a jelentésüket!
- b. Adjuk meg a csomópontok belső viselkedésének egy lehetséges modelljét állapotgráf formalizmussal! Az állapotgráfot írjuk fel táblázatos formában is.
- c. Hogyan modellezhető az üzenetek belső struktúrája?