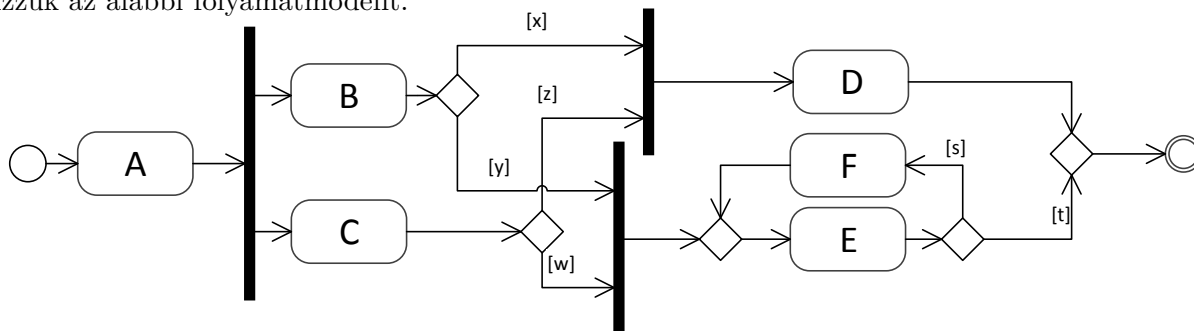


4. gyakorlat – Modellek ellenőrzése és tesztelése

1. Folyamat statikus analízise

Ellenőrizzük az alábbi folyamatmodellt.



- Milyen feltételek mellett teljesen (ellentmondásmentesen) specifikált a folyamat?
- Milyen feltételek mellett determinisztikus is a folyamat?
- Milyen feltételek mellett holtponmentes is a folyamat?
- Milyen további feltételek mellett terminál a folyamat?
- Jólstrukturált-e a folyamat? Ha nem, hogyan lehetne azzá tenni? Segít-e ez a problémákon?

2. Dinamikus analízis teszteléssel

Az $f()$ függvénnyel szemben a következő *követelményeink* vannak:

R1 Az $f()$ függvénynek minden végrehajtása során legalább egyszer outputot kell kiadnia.

R2 Az $f()$ függvénynek tetszőleges inputsorozat esetén terminálnia kell.

R3 Az $f()$ függvény végrehajtása során kiadott legutolsó output értéke kötelezően 0.

A függvény egy lehetséges megvalósítását adja meg az alábbi C nyelvű kódrészlet:

```

1 int readInput();
2 void writeOutput(int out);
3
4 void f() {
5     int x = readInput();
6     int y = readInput();
7     int z = x + y;
8     writeOutput(x * y);
9     while (x > 0 && y > 0) {
10        if (1 == readInput() % 2) {
11            y--;
12            z--;
13        } else {
14            x--;
15            y++;
16        }
17        writeOutput(z + x * y * y - x - y);
18    }
19 }
```

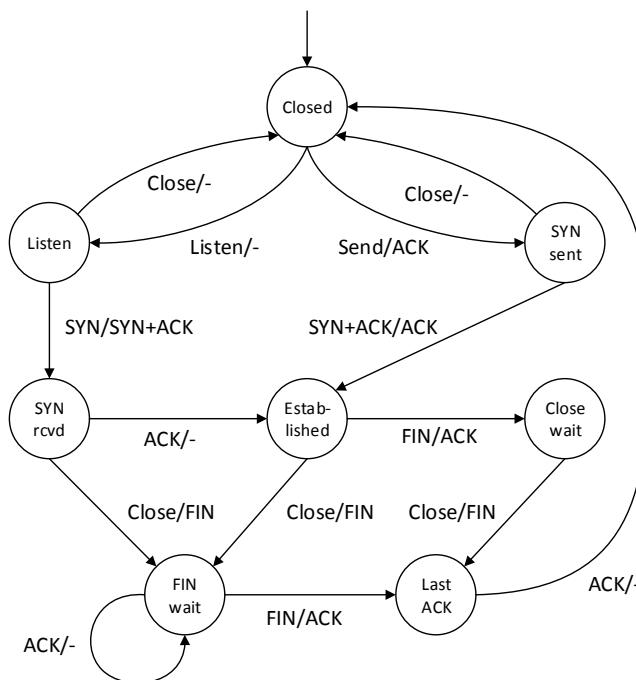
A következő lépések során ellenőrizzük a függvény működését!

- Ábrázoljuk folyamatmodellként $f()$ vezérlési folyamatát!
- Miért lehetünk biztosak az R1 teljesülésében?
- Miért lehetünk biztosak az R2 teljesülésében?
- (Kiegészítő feladat.) Építsünk olyan állapotgépet, amely az $f()$ függvénnyel ekvivalens módon működik. Modellezzük a `readInput()` hívásokat input csatornaként, valamint a `writeOutput()` hívást output csatornaként. Az $f()$ függvény terminálását modellezzük úgy, hogy az automata ad egy speciális outputot, és átmegegy egy nyelő (kimenő átmenet nélküli) állapotba.
- Az R3 követelményt teszteléssel ellenőrizzük. A $t_1 = \langle 2, 3, 5, 7, 11, 13, \dots \rangle$ input szekvencia a tesztesetünk. Detektálunk-e hibát?
- Számítsunk *utasításszintű tesztfedést* a programkódon, vagyis hogy az utasítások mekkora hányadát járja be a tesztelt függvény a teszteset végrehajtása során! Hogy jelenik meg ez a mérőszám a vezérlési folyamaton?
- Az R3 követelményhez a $t_2 = \langle 1, 2, 4, 1, 2, 4, \dots \rangle$ input szekvencia a második tesztesetünk. Detektál-e hibát ez a teszteset? Mekkora a két tesztből álló tesztkészlet együttes utasításfedése?

- h) (Kiegészítő feladat.) Milyen tesztfedettségi metrika számítható a korábban megépített állapotgép alapján?
- i) Készítsünk olyan *tesztorákulum* automatát, amely $f()$ input és output szekvenciái és terminálása alapján el tudja dönteni, hogy az adott lefutás során az R3 követelmény sérült-e! Hogy viselkedik az orákulum a fenti tesztinputra?
- j) Adjunk meg egy tesztetet, amely kimutat egy hibát a programban! Milyen elv alapján sejthettük volna meg, hogy a korábban összeállított tesztkészletünk kiegészítésre szorul?
- k) *Otthoni gyakorlás: vegyük hozzá a tesztkészlethez a $t_3 = \langle 0, 1, 2, 3, 4, 5, \dots \rangle$ és $t_4 = \langle 1, 2, 3, 4, 5, 6, \dots \rangle$ input szekvenciákat mint további tesztteseteket! Detektálunk-e hibát? Hogyan változnak a tesztfedési számok?
- l) Határozzuk meg, hogy pontosan milyen input szekvenciák esetén sérül R3, és javasoljunk hibajavítást!

3. TCP protokoll

Az alábbi állapotgép a TCP protokoll egy részének működését írja le. (A Transmission Control Protocol (TCP) az internetes forgalomban fontos TCP/IP protokollcsalád egyik fő protokollja.) A TCP egy számítógépen futó program és egy másik számítógépen futó másik program között egy adatfolyam megbízható, sorrendhelyes átvitelét hivatott biztosítani. A protokoll egy jól elkülönülő része a kommunikációs csatorna felépítése és lebontása, ennek folyamatát mutatja be az állapotgép. A protokollnak adható parancsok a Listen, a Send és a Close. A küldött és fogadott üzenetek a SYN, az ACK és a FIN (az ábrán a SYN+ACK egy két parancsot tartalmazó üzenet). *Tipp:* a feladat megoldásához nem feltétlenül szükséges a protokoll működésének megértése.



- a) Ellenőrizze és indokolja, hogy az állapotgép teljesen specifikált és determinisztikus-e! Adjon meg két lehetséges módot, ahogyan a nem teljesen specifikált állapotgépek esetén a fel nem tüntetett bemeneteket kezelhetjük!
- b) Adjon egyetlen, legfeljebb 10 bemenetből álló tesztetet, ami 100%-os állapotfedést eredményez! Adja meg a tesztetre az állapotgép által adott kimenetsorozatát is! *Tipp:* a feladat 8 bemenettel is megoldható.
- c) Az eredeti állapotgépből kiindulva finomítással bontsa ketté a FIN wait állapotot. Rajzolja fel azt a variációt, amelyekben a finomított állapotgép pontosan egy ACK üzenetet fogad a Close parancs és a FIN üzenet fogadása között!
- d) Absztrakcióval vonja össze a SYN rcvd és Established állapotokat az Abs. state állapotba!
- e) A b) feladatban elkészített tesztetet a finomított (c) / absztrahált (d) modellen lejátszva (az esetleges nondeterminisztikus döntéseknél megfelelően választva) megkaphatjuk-e a korábbi kimeneteket? Általános is érvényes-e ez a megállapítás, ha tetszőleges (de szabályos) finomítást alkalmazunk egy állapotgép állapotain és újrafuttatunk egy korábbi tesztkészletet? A válaszokat indokolja!