5. gyakorlat – Modellek ellenőrzése és tesztelése

1. feladat

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:

```
int readInput();
void writeOutput(int out);

void f() {
  int x = readInput();
  int y = readInput();
  int z = x + y;
  writeOutput(x * y);
  while (x > 0 && y > 0) {
    if (1 == readInput() % 2) {
      y--;
      z--;
    } else {
      x--;
      y++;
    }
    writeOutput(z + x * y * y - x - y);
  }
}
```

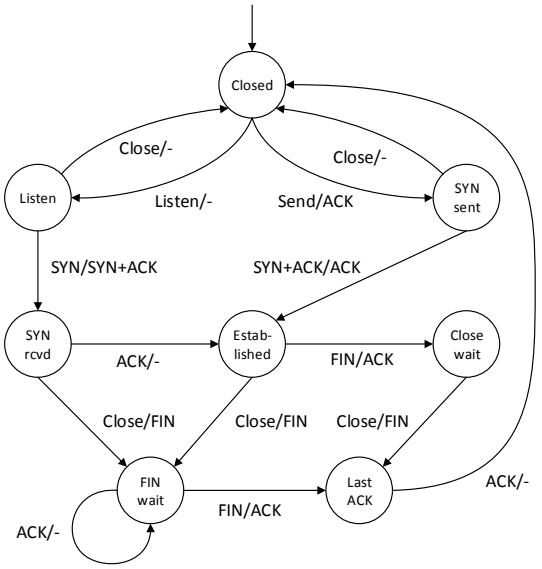
A következő lépések során ellenőrizzük a függvény működését!

a. Ábrázoljuk folyamatmodellként `f()` vezérlési folyamát!
b. Miért lehetünk biztosak az R1 teljesülésében?
c. Miért lehetünk biztosak az R2 teljesülésében?
d. 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 átmegy egy nyelő (kimenő átmenet nélküli) állapotba.
e. Az R3 követelményt teszteléssel ellenőrizzük. A $t_1 = \langle 2, 3, 5, 7, 11, 13, ... \rangle$ input szekvencia a tesztesetünk. Detektálunk-e hibát?
f. Számítsunk *utasításszintű tesztfedést*, vagyis hogy az utasítások mekkora hányadát járja be a tesztelt függvény a teszteset végrehajtása során! Hogy jelenik meg ez a mérőszám a vezérlési folyamon?
g. Az R3 követelményt teszteléssel ellenőrizzük. A $t_2 = \langle 1, 2, 4, 1, 2, 4, ... \rangle$ input szekvencia a tesztesetük. Detektál-e hibát ez a teszteset? Mekkora a két tesztből álló tesztkészlet együttes utasításfedése?
h. 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. Kimaradt-e a fedésből a vezérlési folyam, ill. az automatamodell bármelyik része? Milyen fedési metrika mutathatja ezt ki?

k. Adjunk meg egy tesztesetet, amely kimutat egy hibát a programban! Milyen elv alapján sejthettük volna meg, hogy a korábban összeállított tesztkészletünk kiegészítésre szorul?
l. *Otthoni munka: vegyük hozzá a tesztkészlethez a $t_3 = \langle 0, 1, 2, 3, 4, 5, ... \rangle$ és $t_4 = \langle 1, 2, 3, 4, 5, 6, ... \rangle$ input szekvenciákat mint további teszteseteket! Detektálunk-e hibát? Hogyan változnak a tesztfedési számok?
m. *Kiegészítő feladat: határozzuk meg, hogy pontosan milyen input szekvenciák esetén sérül R3, és javasoljunk hibajavítást!

2. feladat

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 tesztesetet, ami 100%-os állapotfedést eredményez! Adja meg a tesztesetre az állapotgép által adott kimenetsorozatot 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, amelyikben 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 tesztesetet a finomított (c.) / absztrahált (d.) modellen lejátszva (az esetleges nemdeterminisztikus döntéseknél megfelelően választva) megkaphatjuk-e a korábbi kimeneteket? Általánosan is érvényes-e ez a megállapítás, ha tetszőleges (de szabályos) finomítást alkalmazunk egy állapotgép állapotain és újrafuttatunk egy korábbi tesztkészletet? A válaszokat indokolja!