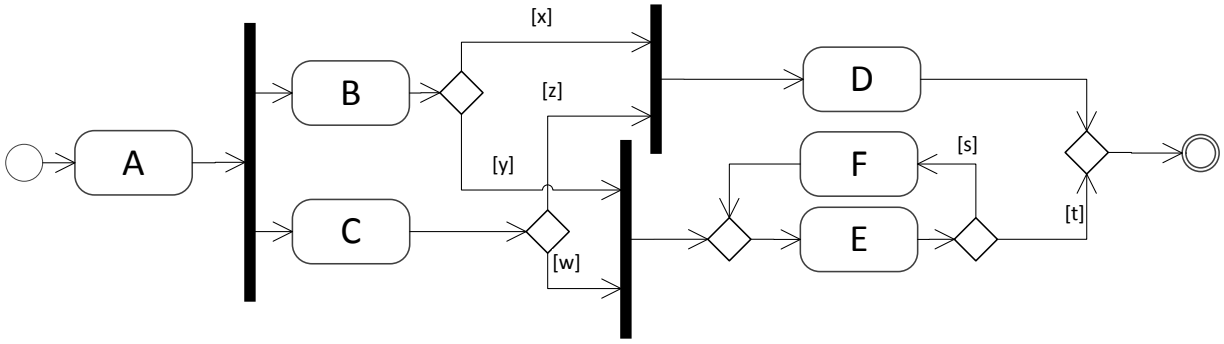
5. gyakorlat – Modellek ellenőrzése és tesztelése

1. feladat

Ellenőrizzük az alábbi folyamatmodellt.



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 terminál a folyamat?
e. Jólstrukturált-e a folyamat? Ha nem, hogyan lehetne azzá tenni? Segít-e ez a problémákon?

2. 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. *Kiegészítő feladat: miért lehetünk biztosak R2 teljesülésében?
d. Az R3 követelményt teszteléssel ellenőrizzük. A $t_1 = \langle 1, 2, 4, 1, 2, 4, ... \rangle$ input szekvencia szolgál első tesztinputként. Detektál-e hibát ez a teszteset?
e. Számítsunk *utasítás szintű 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?
f. 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. Milyen tesztfedettségi metrika számítható az állapotgép alapján?
g. 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?
h. Építsünk olyan adatfolyamhálót, amely összekapcsolja a tesztelés alatt álló (SUT) automatamodellt, a tesztorákulumot, és egy olyan komponenst, amely a fenti tesztszekvenciát generálja!
i. A $t_2 = \langle 2, 3, 5, 7, 11, 13, ... \rangle$ input szekvencia a második tesztesetünk. Detektálunk-e hibát? Mekkora a két tesztből álló tesztkészlet együttes utasításfedése?
j. Kimaradt-e a fedésből a vezérlési folyam ill. az automatamodell bármelyik része? Milyen fedési metrika mutathatja ezt ki?
k. *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?
l. Adjunk meg egy tesztesetet, amely kimutat egy hibát a programban! Milyen elv alapján sejthettük volna meg, hogy a korábban összeállított tesztkészletünk kiegészítésre szorul?
m. *Kiegészítő feladat: határozzuk meg, hogy pontosan milyen input szekvenciák esetén sérül R3, és javasoljunk hibajavítást!