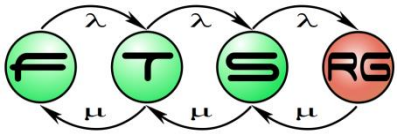
Modellek ellenőrzése és tesztelése

Rendszermodellezés

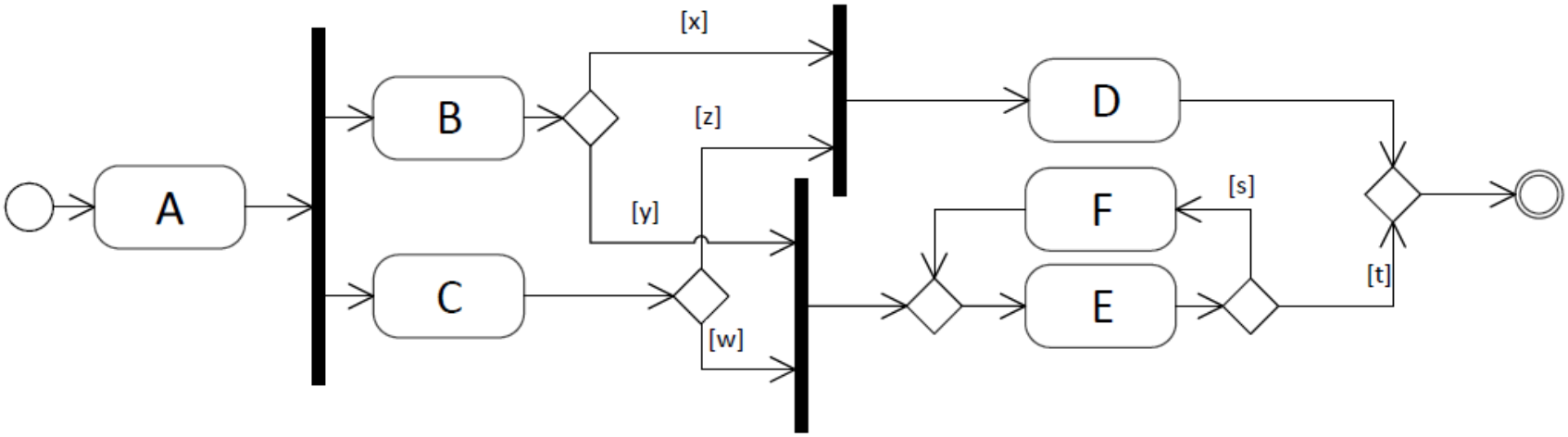
iMSc gyakorlat

Budapesti Műszaki és Gazdaságtudományi Egyetem
Hibatűrő Rendszerek Kutatócsoport



1. Folyamat statikus analízise

Ellenőrizzük az alábbi folyamatmodellt.



a) Milyen feltételek mellett **teljesen specifikált** a folyamat?

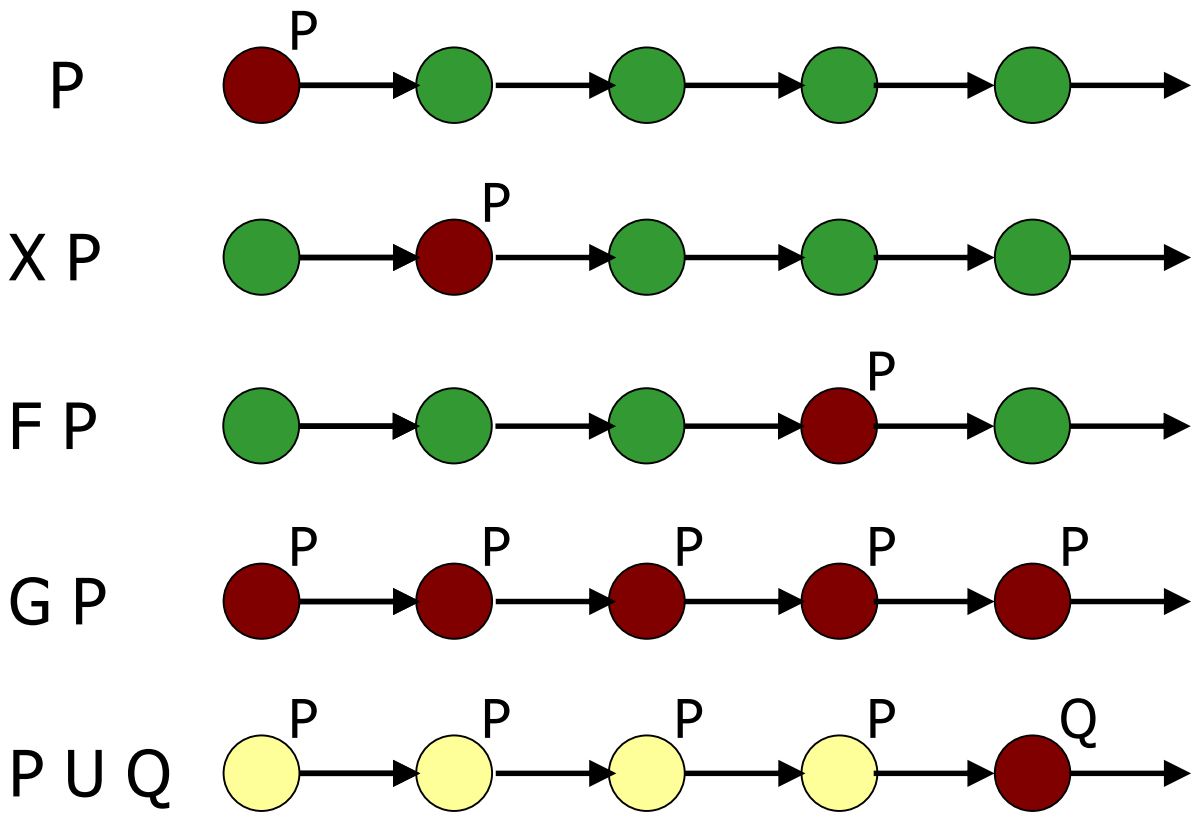
b) Milyen feltételek mellett **determinisztikus** is a folyamat?

c) Milyen feltételek mellett **holtpontmentes** is a folyamat?

d) Milyen további feltételek mellett **terminál**ó a folyamat?

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

Temporális logika



https://en.wikipedia.org/wiki/Linear_temporal_logic

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**.

2. Dinamikus analízis teszteléssel

```
 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 }
```

2. Dinamikus analízis teszteléssel

R1 Legalább egyszer outputot ad.

R2 Tetszőleges input-sorozat esetén terminál.

R3 Legutolsó output értéke kötelezően 0.

```
 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) Ábrázoljuk folyamatmodellként f() **vezérlési folyamát**!

b) Miért lehetünk biztosak az **R1 teljesülésében**?

2. Dinamikus analízis teszteléssel

R1 Legalább egyszer outputot ad.

R2 Tetszőleges input-sorozat esetén terminál.

R3 Legutolsó output értéke kötelezően 0.

```
 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 }
```

c) Miért lehetünk biztosak az **R2 teljesülésében**?

Terminálás bizonyítása

Általános séma:

- Keressünk egy függvényt
  - A ciklus őrfeltételében szereplő változók felett
- A következő feltételekkel
  - I. A ciklus minden végrehajtásában sz
  - II. Di
  - III. Al

2x+y > 0

```
4 void f() {
5   int x = readInput();
6   int y = readInput();
    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++;
      }
      writeOutput(z        y * y - x - y);
```

alapozot

2x+y

x’ = x - 1

y’ = y + 1

2x’+y’ = 2(x-1)+y+1 = 2x+y-1

2x+y

x’ = x

y’ = y - 1

2x’+y’ = 2x+(y-1)

Megállási probléma

Adott program esetén az a kérdés, hogy tetszőleges bemenet hatására garantáltan terminálódik-e, eldönthetetlen.

https://en.wikipedia.org/wiki/Halting_problem

- Praktikusan:
  - Léteznek programok, amikre bizonyítható, hogy leáll/nem áll le
  - Léteznek olyanok is, amikre nem lehet bizonyítást adni

A fenti tétel bizonyítása vázlatosan:

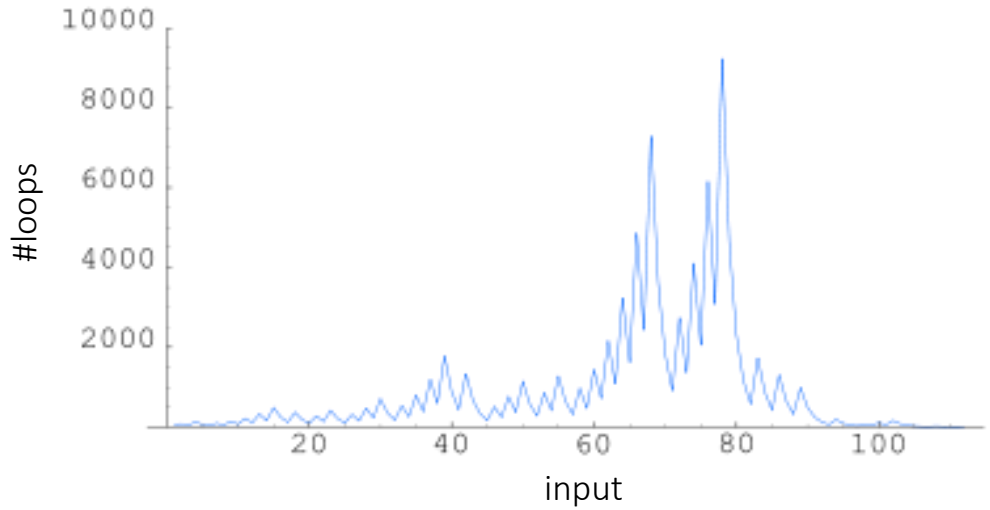
- Tfh. Létezik a *halts(p)* függvény, ami *minden programra* el tudja dönteni a megállási problémát
- Ekkor a következő program ellentmondásra vezet:

```
void g() {
  if (halts(g)) loop_forever();
}
```

Megállási probléma - példa

Nem létezik bizonyítás a következő program terminálására (bár minden eddig vizsgált bemenetre terminált):

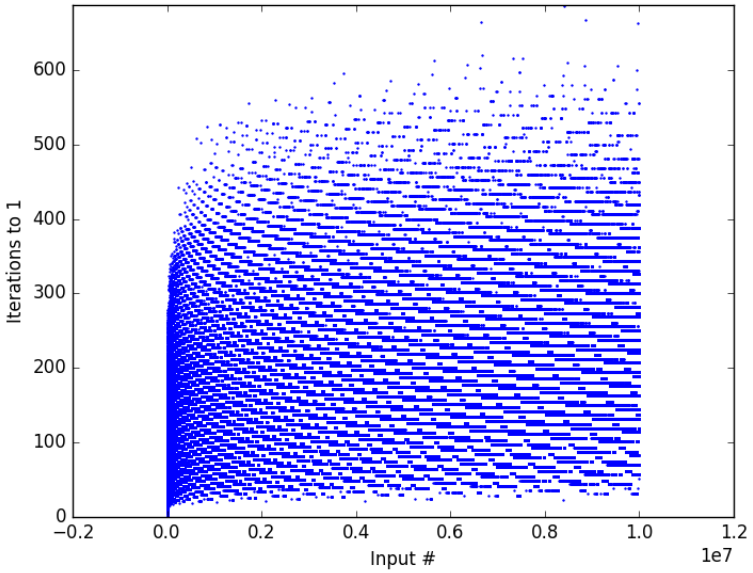
```
x = input();
while (x > 1) {
     if (x % 2 == 0)
          x = x/2;
     else
          x = 3*x + 1;
}
```

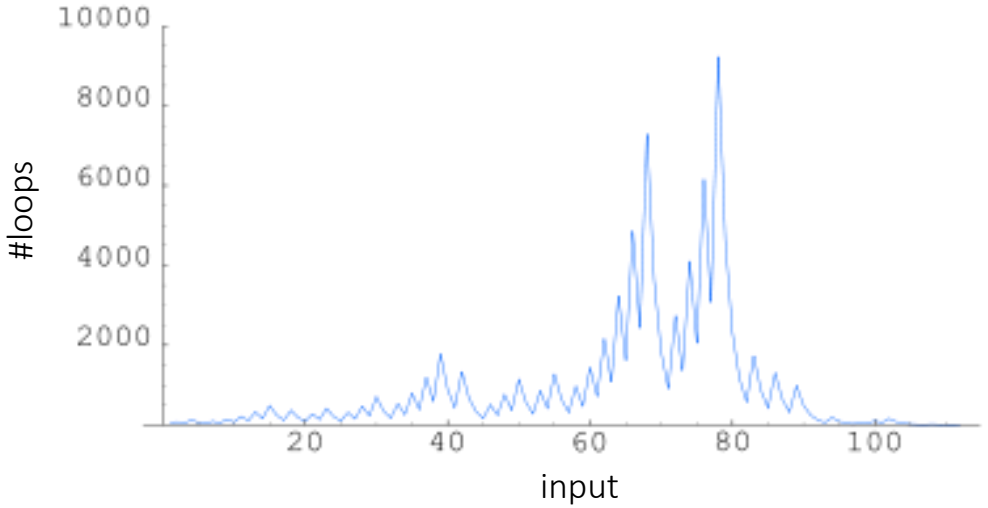


https://en.wikipedia.org/wiki/Collatz_conjecture

Megállási probléma - példa

Nem létezik bizonyítás a következő program terminálására (bár minden eddig vizsgált bemenetre terminált):





https://en.wikipedia.org/wiki/Collatz_conjecture

2. Dinamikus analízis teszteléssel

R1 Legalább egyszer outputot ad.

R2 Tetszőleges input-sorozat esetén terminál.

R3 Legutolsó output értéke kötelezően 0.

```
 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 }
```

d) *(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 átmegy egy nyelő (kimenő átmenet nélküli) állapotba.

2. Dinamikus analízis teszteléssel

R1 Legalább egyszer outputot ad.

R2 Tetszőleges input-sorozat esetén terminál.

R3 Legutolsó output értéke kötelezően 0.

```
 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 }
```

e) Az R3 követelményt **teszteléssel** ellenőrizzük. A $t1 = \langle 2, 3, 5, 7, 11, 13, \ldots \rangle$ **input szekvencia** a tesztesetünk. Detektálunk-e hibát?

2. Dinamikus analízis teszteléssel

R1 Legalább egyszer outputot ad.

R2 Tetszőleges input-sorozat esetén terminál.

R3 Legutolsó output értéke kötelezően 0.

```
 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 }
```

f) 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 folyamon**?

2. Dinamikus analízis teszteléssel

R1 Legalább egyszer outputot ad.

R2 Tetszőleges input-sorozat esetén terminál.

R3 Legutolsó output értéke kötelezően 0.

```
 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 }
```

g) Az R3 követelményhez a *t2* = ⟨ h1, 2, 4, 1, 2, 4, ...⟩ input szekvencia a **második tesztesetünk**. Detektál-e hibát ez a teszteset? Mekkora a két tesztből álló tesztkészlet **együttes utasításfedése**?

2. Dinamikus analízis teszteléssel

R1 Legalább egyszer outputot ad.

R2 Tetszőleges input-sorozat esetén terminál.

R3 Legutolsó output értéke kötelezően 0.

```
 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 }
```

h) (Kiegészítő feladat.) Milyen **tesztfedettségi metrika** számítható a korábban megépített **állapotgép** alapján?

2. Dinamikus analízis teszteléssel

R1 Legalább egyszer outputot ad.

R2 Tetszőleges input-sorozat esetén terminál.

R3 Legutolsó output értéke kötelezően 0.

```
 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 }
```

i) Készítsünk olyan **tesztorákulum** állapotgépet, 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?

2. Dinamikus analízis teszteléssel

R1 Legalább egyszer outputot ad.

R2 Tetszőleges input-sorozat esetén terminál.

R3 Legutolsó output értéke kötelezően 0.

```
 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 }
```

j) Adjunk meg egy **tesztesetet**, amely **kimutat egy hibát** a programban! **Milyen elv alapján** sejthettük volna meg, hogy a korábban összeállított tesztkészletünk kiegészítésre szorul?

2. Dinamikus analízis teszteléssel

R1 Legalább egyszer outputot ad.

R2 Tetszőleges input-sorozat esetén terminál.

R3 Legutolsó output értéke kötelezően 0.

```
 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 }
```

k) *(Kiegészítő feladat.)* Vegyük hozzá a tesztkészlethez a *t3* = ⟨0, 1, 2, 3, 4, 5, ...⟩ és *t4* = ⟨1, 2, 3, 4, 5, 6, ...⟩ input szekvenciákat mint **további teszteseteket**! Detektálunk-e hibát? Hogyan változnak a **tesztfedési számok**?

2. Dinamikus analízis teszteléssel

R1 Legalább egyszer outputot ad.

R2 Tetszőleges input-sorozat esetén terminál.

R3 Legutolsó output értéke kötelezően 0.

```
 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 }
```

I) *(Kiegészíto feladat.)* Határozzuk meg, hogy **pontosan milyen input szekvenciák** esetén sérül R3, és javasoljunk **hibajavítást**!