Struktúra alapú teszttervezési módszerek

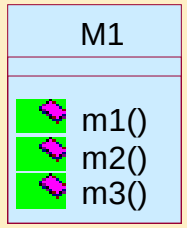
Majzik István, Micskei Zoltán

http://www.inf.mit.bme.hu/

Tesztttervezés módszerei

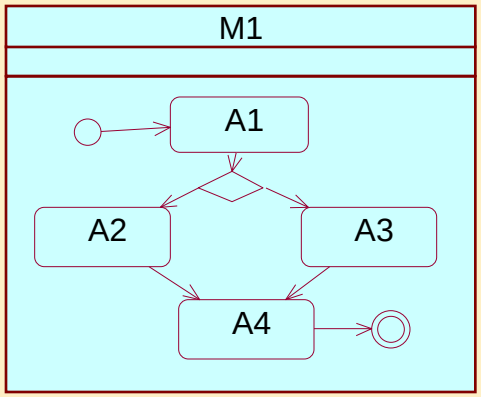
I. Specifikáció alapú

- A rendszer mint „fekete doboz” adott
- Csak a külső viselkedés (funkció) ismert, a belső felépítés (pl. forráskód) nem
- Tesztelés alapja: specifikált funkciók léte; extra funkciók hiánya



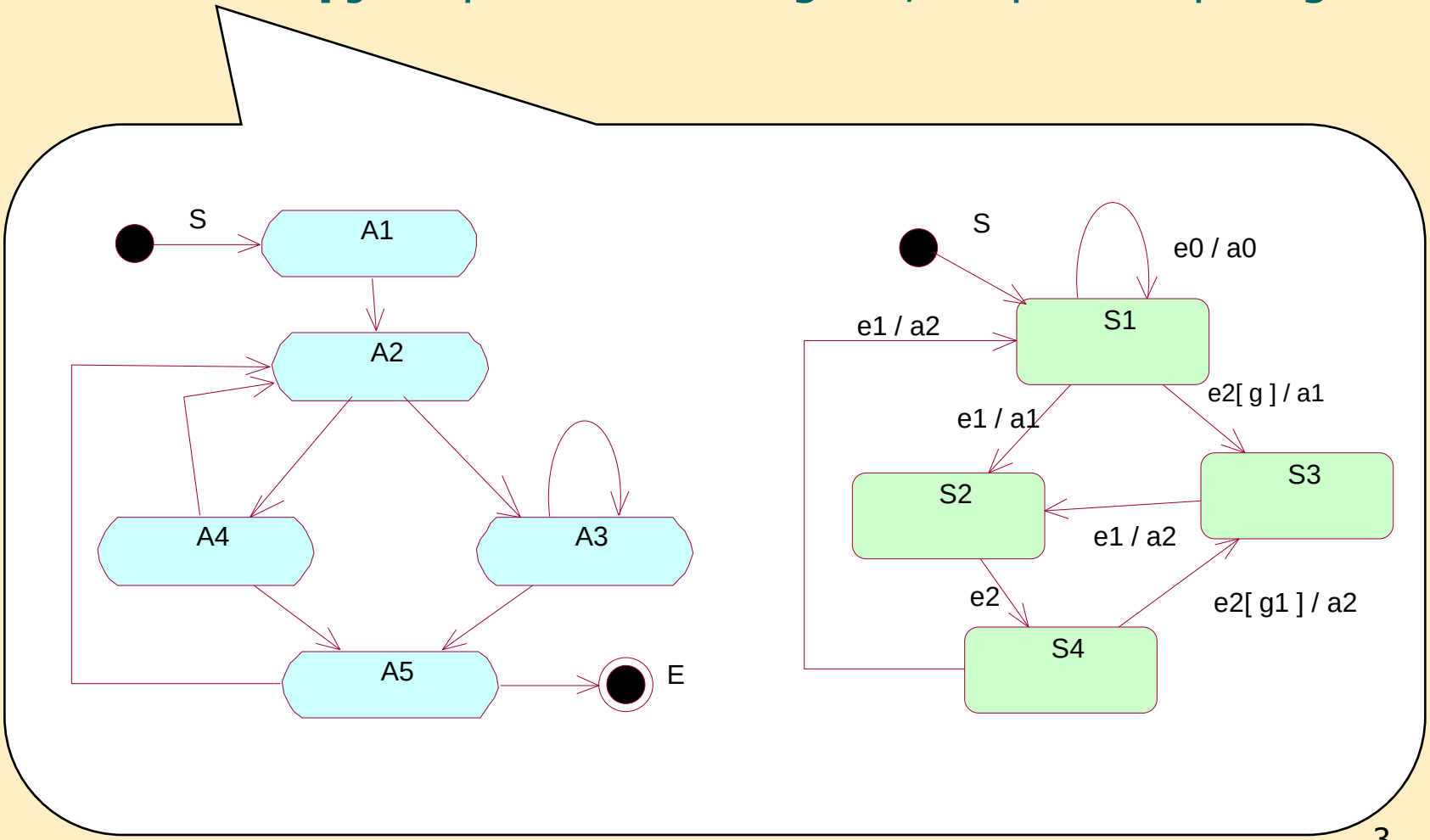
II. Struktúra alapú

- A rendszer mint „üvegdoboz” adott
- A belső struktúra is ismert
- Tesztelés alapja a belső működés: programgráf bejárása



A belső struktúra

- Jól kezelhető struktúra:
  - **Modell alapján**: pl. aktivitás diagram, állapottérkép diagram



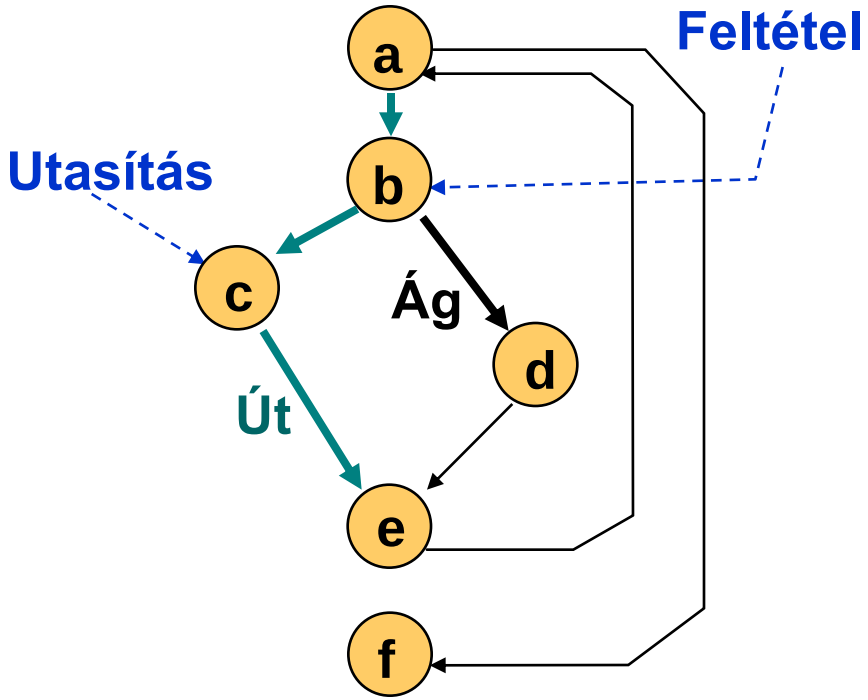
A belső struktúra

- Jól kezelhető struktúra:
  - Modell alapján: pl. aktivitás diagram, állapottérkép diagram
  - **Forráskód alapján**: vezérlési gráf (programgráf)

Forráskód:

```
a:  for (i=0; i<MAX; i++) {
b:      if (i==a) {
c:          n=n-i;
        } else {
d:          m=n-i;
        }
e:      printf("%d\n",n);
    }
f:  printf("Ready.")
```

Vezérlési gráf:



Tesztminőségi mértékszámok

A tesztelés hatékonyságának számszerű jellemzése:

A tesztelhető elemek mekkora részét teszteltük, pl.

1. Utasítások → Utasítás fedettség
2. Döntési ágak → Döntési ág fedettség
3. Feltételek → Feltétel fedettség
4. Végrehajtási utak → Út fedettség

Ez nem a hibafedés!

Szabványok előírása lehet (DO-178B, MSZ EN 50128,...)

- 100% utasítás fedettség általában alapkövetelmény

Tartalom

- Vezérlési folyam alapú kritériumok
  - Utasítás lefedettség
  - Döntési ág lefedettség
  - Feltétel lefedettségek
  - Útvonal lefedettség
- Adatfolyam alapú kritériumok
  - Definiálás – használat fedettségek
  - Definíciómentes útvonalak fedettsége
- Összefoglalás
  - Módszerek kombinációja
  - Teszt fedettség mérése

Alapfogalmak

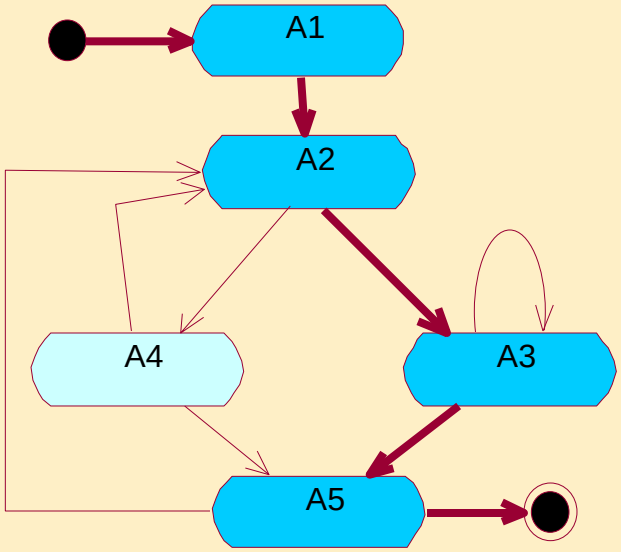
- Utasítás (statement)
- Blokk (block)
  - Utasítások egybefüggő sorozata, amik között nincs elágazás vagy függvényhívás
- Feltétel (condition)
  - Egyszerű vizsgálat, amiben nincs logikai (Boole) operátor
- Döntés (decision)
  - Egy ág bejárásának eldöntéséhez tartozó, nulla vagy több logikai operátorral összekötött feltételből álló kifejezés
- Út (path)
  - Utasítások sorozata, tipikusan a modul be és kilépési pontja között

1. Utasítás lefedettség (Statement coverage)

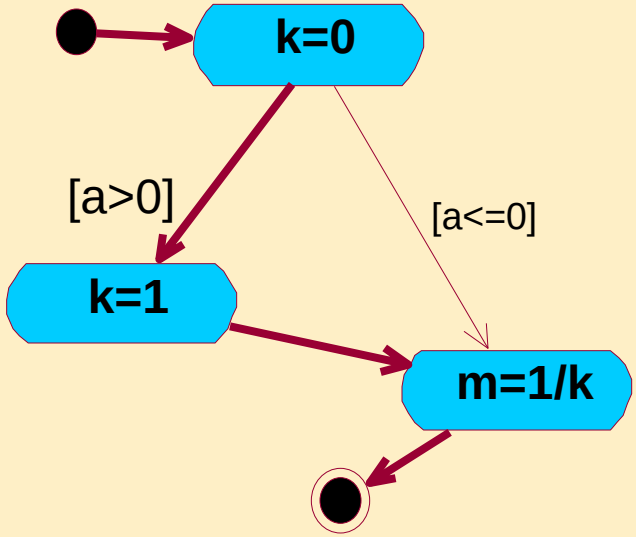
Definíció:

$$\frac{\text{Tesztelés során végrehajtott utasítások száma}}{\text{Összes utasítás száma}}$$

Utasításokat kihagyási feltételeit nem veszi figyelembe!



Utasítás fedettség: 80%



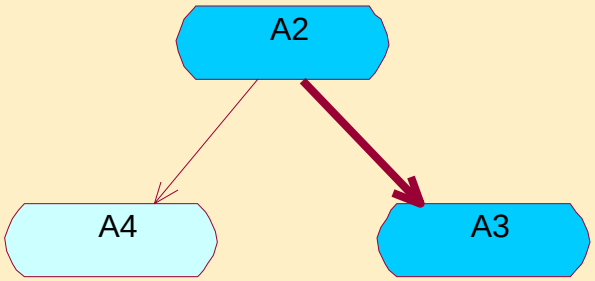
Utasítás fedettség: 100%

2. Döntés lefedettség (Decision coverage)

Definíció:

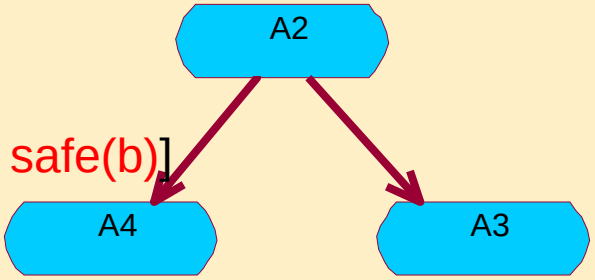
$$\frac{\text{Tesztelés során végrehajtott döntési ágak száma}}{\text{Összes lehetséges döntési ág száma}}$$

Nem vesz figyelembe minden feltétel-kombinációt!



Döntési ág fedettség: 50%

[safe(c) || safe(b)]



Döntési ág fedettség: 100%

3. Feltétel lefedettség (Condition coverage)

Generikus fedettségi mérték feltétel fedettségekhez:

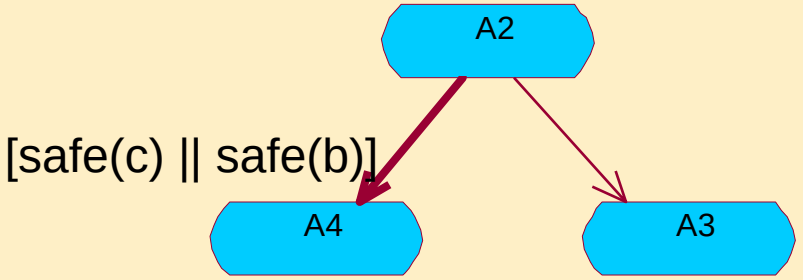
$$\frac{\text{Feltételek tesztelt kombinációinak száma}}{\text{Feltételek megcélzott kombinációinak száma}}$$

Definíció (milyen kombinációkat célzunk meg):

- Minden feltétel legyen igaznak és hamisnak is beállítva a tesztelés során
  - Nem feltétlenül eredményez 100% döntés lefedettséget!

Példa 100%-os feltétel lefedettséghez:

1. safe(c) = true, safe(b) = false
2. safe(c) = false, safe(b) = true



Más definíció:

- Minden feltételt igaznak és hamisnak is kiértékelünk
  - Ez nem ugyanaz mint a fenti, a lusta kiértékelés miatt

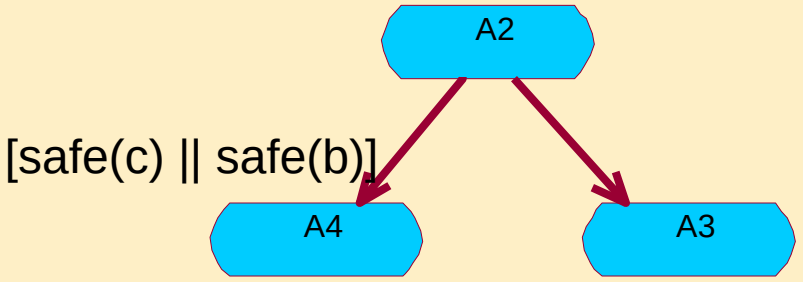
4. Feltétel/döntés lefedettség

Condition/Decision Coverage (C/DC)

Definíció:

- Minden feltétel felveszi az összes lehetséges kimenetét legalább egyszer, és
- minden döntés felveszi az összes lehetséges kimenetét egyszer

100%-os C/DC lefedettséghez:
1. safe(c) = true, safe(b) = true
2. safe(c) = false, safe(b) = false



Nem vizsgálja meg, hogy a feltételnek tényleg van-e hatása a döntés eredményére!

5. Módosított feltétel/döntés lefedettség

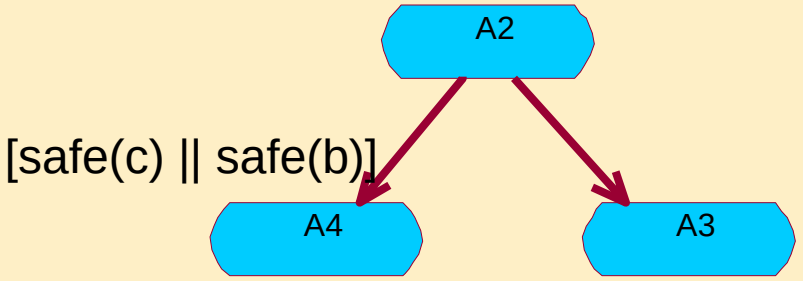
Modified Condition/Decision Coverage (MC/DC)

Definíció:

- Minden feltétel felveszi az összes lehetséges kimenetét legalább egyszer, és
- minden döntés felveszi az összes lehetséges kimenetét egyszer, és
- minden feltétel a többitől függetlenül befolyásolja a hozzá tartozó döntés kimenetelét

100%-os MC/DC lefedettséghez:

1. safe(c) = true, safe(b) = false
2. safe(c) = false, safe(b) = true
3. safe(c) = false, safe(b) = false



6. Minden feltétel kombináció lefedettsége

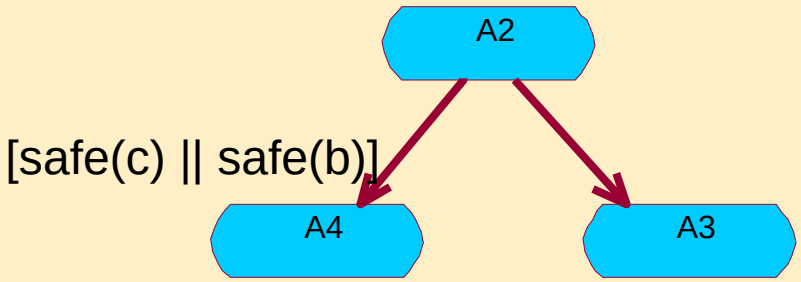
Multiple Condition Coverage

Definíció:

- A feltételek kimeneteinek minden lehetséges kombinációja bekövetkezett a tesztelés során
  - Általában a feltételek számával exponenciálisan növekedő számú teszt szükséges
  - Kevesebb, ha a lusta kiértékelést is figyelembe vesszük

100%-os feltétel kombináció lefedettséghez:

1. safe(c) = true, safe(b) = false
2. safe(c) = false, safe(b) = true
3. safe(c) = false, safe(b) = false
4. safe(c) = true, safe(b) = true



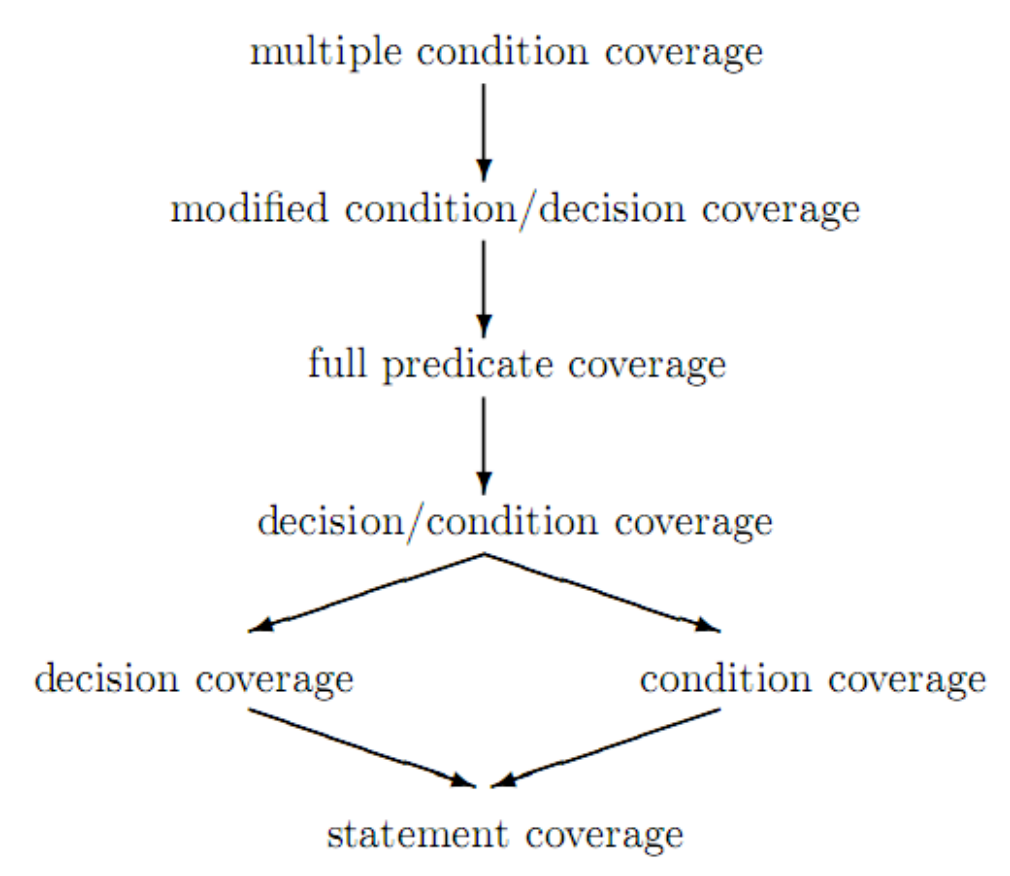
Feltétel és döntési lefedettségek összefoglalása

Table 1. Types of Structural Coverage

Coverage Criteria	Statement Coverage	Decision Coverage	Condition Coverage	Condition/ Decision Coverage	MC/DC	Multiple Condition Coverage
Every point of entry and exit in the program has been invoked at least once		•	•	•	•	•
Every statement in the program has been invoked at least once	•					
Every decision in the program has taken all possible outcomes at least once		•		•	•	•
Every condition in a decision in the program has taken all possible outcomes at least once			•	•	•	•
Every condition in a decision has been shown to independently affect that decision's outcome					•	•[8]
Every combination of condition outcomes within a decision has been invoked at least once						•

Forrás: Kelly J. Hayhurst et al. „A Practical Tutorial on Modified Condition/Decision Coverage", NASA/TM-2001-210876, 2001

Feltétel és döntési lefedettségek összefoglalása



Vezérlési folyam alapú kritériumok viszonya

Forrás: S. A. Vilkomir and J. P. Bowen, “From MC/DC to RC/DC: formalization and analysis of control-flow testing criteria,” *Formal Aspects of Computing*, vol. 18, no. 1, pp. 42-62, 2006.

Példa: Vezérlési folyam alapú kritériumok

```
Product getProduct(String name, Category cat){
  if (name == null || ! cat.isValid)
     throw new IllegalArgumentException();

  Product p = ProductCache.getItem(name);

  if (p == null){
     p = DAL.getProduct(name, cat);
  }

  return p;
}
```

Cél: Utasítás fedettség, döntési ág fedettség, C/DC fedettség

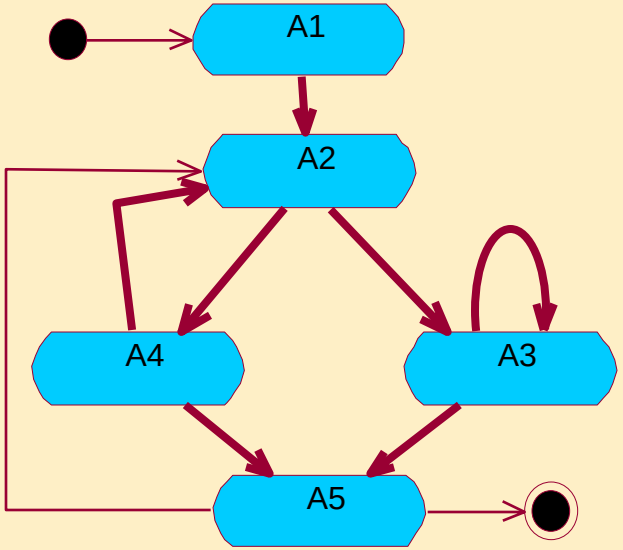
7. Alap út lefedettség (Basis path coverage)

Definíció:

$$\frac{\text{Tesztelés során bejárt független utak száma}}{\text{Összes független út száma}}$$

100% út lefedettség együtt jár:

- 100% utasítás lefedettség, 100% döntés lefedettség
- feltétel lefedettség nem garantált



Út fedettség: 80%

Utasítás fedettség: 100%

Tesztelés az út fedettség alapján 1/2

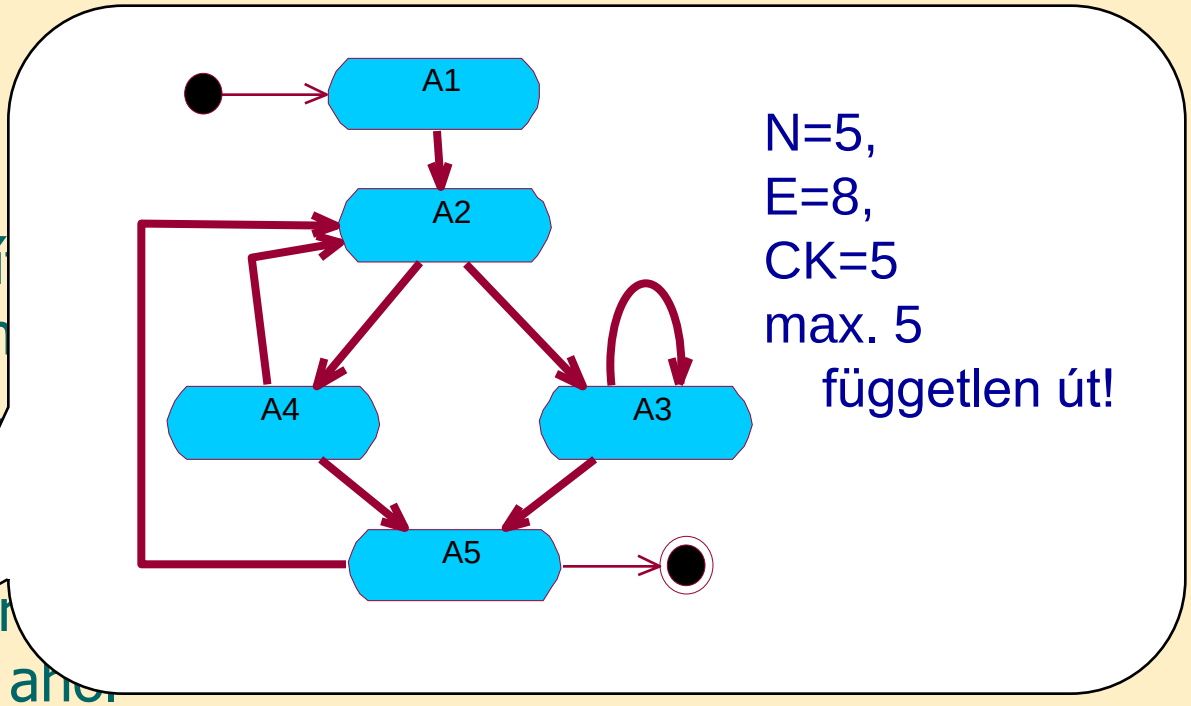
- Cél: Független utak bejárása
  - Független utak a tesztelés szempontjából:
    Van olyan utasítás vagy elágazás,
    ami a másikban nincs meg
- A független utak maximális száma:
  - CK, ciklomatikus komplexitás
  - Szabályos vezérlési gráf alapján meghatározható:
    CK(G)=E-N+2, ahol

    E: élek száma

    N: csomópontok száma a G vezérlési gráfban
    (vezérlési gráf összekötött, 1 kimenet és 1 bemenet)
- A független utak halmaza nem egyedi

Tesztelés az út fedettség alapján 1/2

- Cél: Független
  - Független utak
    Van olyan utasí
    ami a másikban
- A független ut
  - CK, ciklomatik
  - Szabályos v
    CK(G)=E-N+2, ah
    E: élek száma
    N: csomópontok száma a G vezérlési gráfban
    (vezérlési gráf összekötött, 1 kimenet és 1 bemenet)
- A független utak halmaza nem egyedi



Tesztelés az út fedettség alapján 2/2

- Algoritmus:
  - Max. CK számú független út kiválasztása
  - Bemenetek generálása egy-egy út bejárásához
- Problémák:
  - Nem minden út bejárható (ld. feltételek).
    - Generálható-e az úthoz bemeneti szekvencia?
    - Lehetséges-e a belső változók beállítása?
  - Ciklusok: Korlátozni (minimalizálni) kell a bejárást!
- Teljesen automatikus módszerek nem léteznek

Kiegészítő fedettségi mértékek

- Loop
  - Ciklusok 0 (ahol értelmezhető), 1, illetve többszöri végrehajtása
- Race
  - Több szál futása egyszerre egy-egy kódrészleten
- Relational operator
  - Határértékek használata összehasonlító operátorok esetén
- Weak mutation
  - Operátor vagy operandus hibákra tesztek készítése
- Table
  - Ugrási táblák (állapotgép megvalósítások) teljes tesztelése
- Linear code sequence and jump
  - Lineáris szekvenciák fedése a forráskódban (lehetnek benne vezérlési utasítások, de lineárisan bejárva)
- Object code branch
  - Gépi kódú feltételes ugrások fedése (fordító függő)

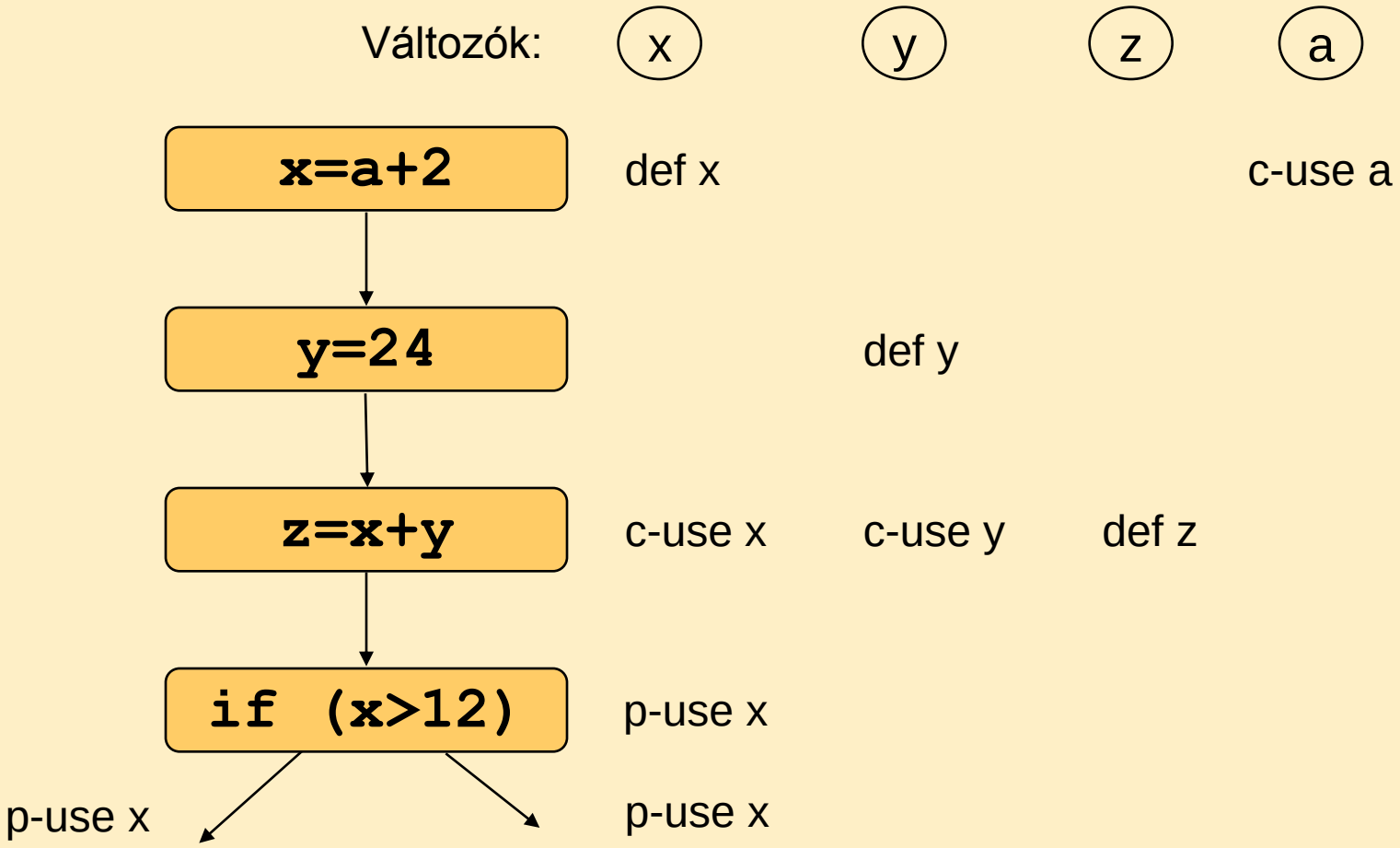
Tartalom

- Vezérlési folyam alapú kritériumok
  - Utasítás lefedettség
  - Döntési ág lefedettség
  - Feltétel lefedettségek
  - Útvonal lefedettség
- Adatfolyam alapú kritériumok
  - Definiálás – használat fedettségek
  - Definíciómentes útvonalak fedettsége
- Összefoglalás
  - Módszerek kombinációja
  - Teszt fedettség mérése

Alapfogalmak

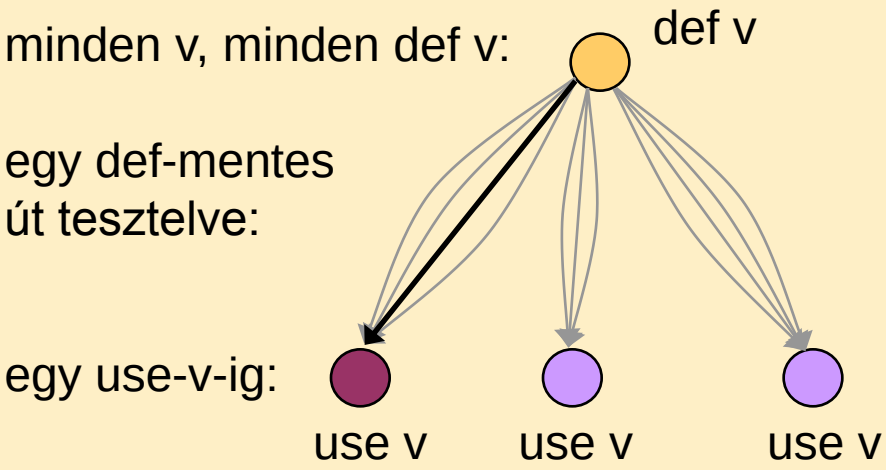
- Alapötlet: Változók értékadásának és felhasználásának viszonyát vizsgáljuk
  - Hibás értékadás, hibás felhasználás detektálható
- A programgráf címkézése
  - def(v): adott helyen a v változónak értéket adunk
  - use(v): adott helyen a v változót felhasználjuk
    - p-use(v): v értékét feltételben használjuk
    - c-use(v): v értékét számolásban használjuk
- A programgráf útjai
  - Definíciómentes út (definition clear path): v-nek nem adunk értéket az út csomópontjaiban

Példa a címkézésre



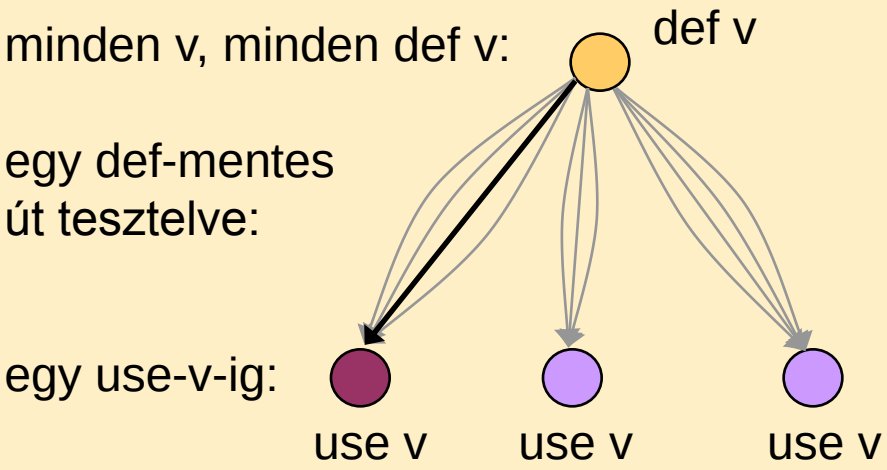
Adatfolyam alapú tesztelési kritériumok

- All-defs:
  - def v
  - use v

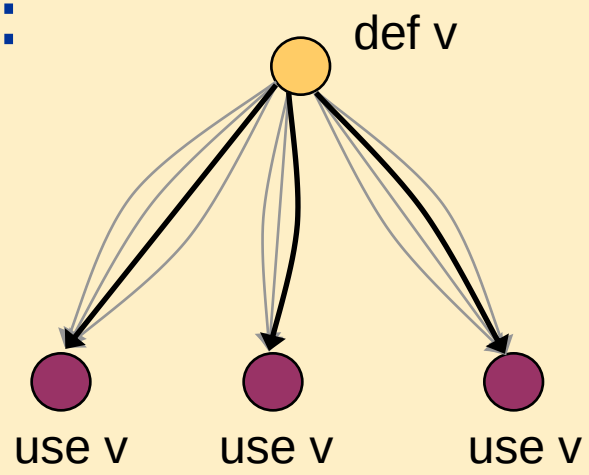


Adatfolyam alapú tesztelési kritériumok

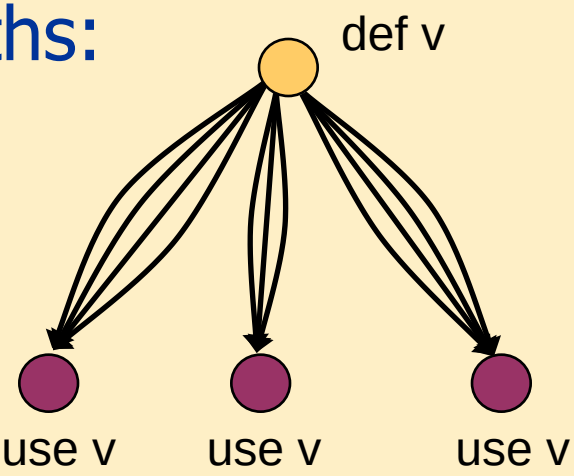
- All-defs:
  - def v
  - use v



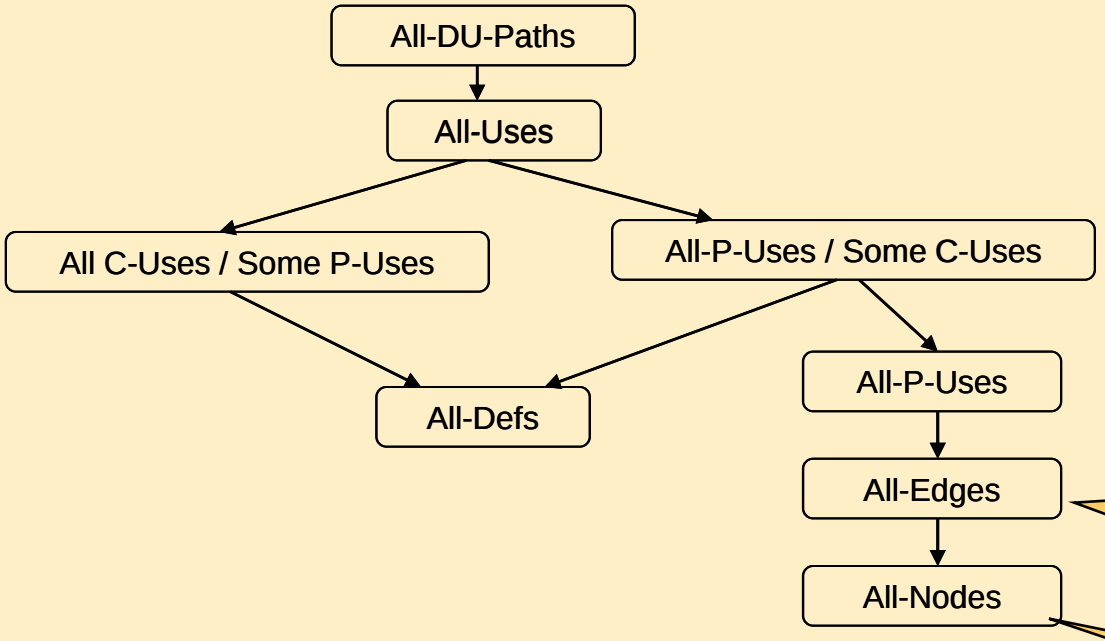
- All-uses:
  (p-uses,
  c-uses)



All-paths:



Strukturális fedettségi kritériumok viszonya



Bonyolultabb fedési kritériumok a biztonságkritikus szabványokban

Ezt már egy igényesebb tesztelési cél

A projektek nagy része ezt sem éri el, 90%-os lefedettség már egy jó cél

Tartalom

- Vezérlési folyam alapú kritériumok
  - Utasítás lefedettség
  - Döntési ág lefedettség
  - Feltétel lefedettségek
  - Útvonal lefedettség
- Adatfolyam alapú kritériumok
  - Definiálás – használat fedettségek
  - Definíciómentes útvonalak fedettsége
- Összefoglalás
  - Módszerek kombinációja
  - Teszt fedettség mérése

Teszttervezési módszerek összefoglalása

- Specifikáció és struktúra alapú módszerek
  - Sokféle módszer illetve technika
  - Mindegyik alkalmazásához gyakorlat kell
- A gyakorlatban általában csak a legegyszerűbb módszereket használják
  - Biztonságkritikus rendszerek: Vannak előírt módszerek (pl. DO178B szabvány: MC/DC szerinti tesztelés)
- Technikák kombinációja hatásos általában:
  - Példa (MS tanulmány):

    specifikáció alapú: 83%-os kódfedés

    + exploratory: 86%-os kódfedés

    + strukturális: 91%-os kódfedés

Mire jók a teszt fedettségi mértékek?

- Mire jók?
  - Megtalálhatók azok a programrészek, ahol hiányos a tesztelés
    - Ez alapján bővíthető a teszt készlet
  - Azonosíthatók a redundáns tesztek (azonos részeket fednek le)
    - Adatfüggésre is figyelni kell (más adattal más hibát tesztel)
  - Indirekt mértéke a kód minőségének a sikeres tesztekhez tartozó fedettség
    - Inkább mértéke a tesztkészlet teljességének
  - A tesztelés befejezése a mértékekhez köthető
- Mire nem jók?
  - Az implementációból kihagyott (nem megvalósított) követelmények tesztelése
  - Kódrészletek kiragadásával történő tesztelés eredményessége kérdéses (a kódrészlet elveszíti környezetét)

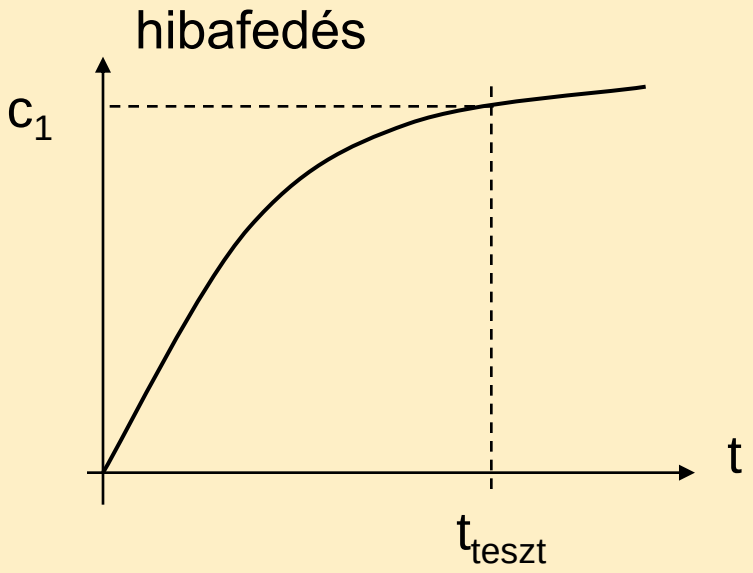
Tesztek végrehajtása

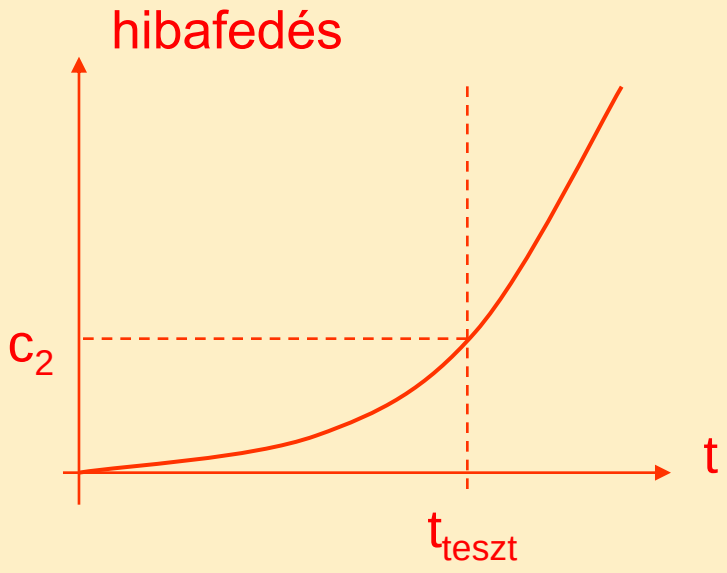
Milyen sorrendben hajtsuk végre a teszteket:

Ha várhatóan kevés a hiba:

Először a *hatékony* (nagyobb hibafedésű) teszteket!

- Hosszabb utak,
- összetett feltételű elágazások tesztje





Példa: Teszt fedettség mérése: gcov, ggcov

- gcov: gcc mellett használható
  - Korlátozott tudású (felülvizsgálat szükséges)
- Program felműszerezése: Speciális fordítás

  ```
  gcc -ftest-coverage -fprofile-arcs src.c
  ```

  - Naplózza az egyes programágak végrehajtását `-fprofile-arcs` esetén döntési ágakat is
  - Elkészülő log fájlok: `.bb, .bbg,.da`
- Off-line elemzés: gcov
  - ```
    gcov src.c
    88.89% of 9 source lines executed in file cov.c
    ```
  - Annotált forráskód
- Grafikus jelentés: ggcov
    ```

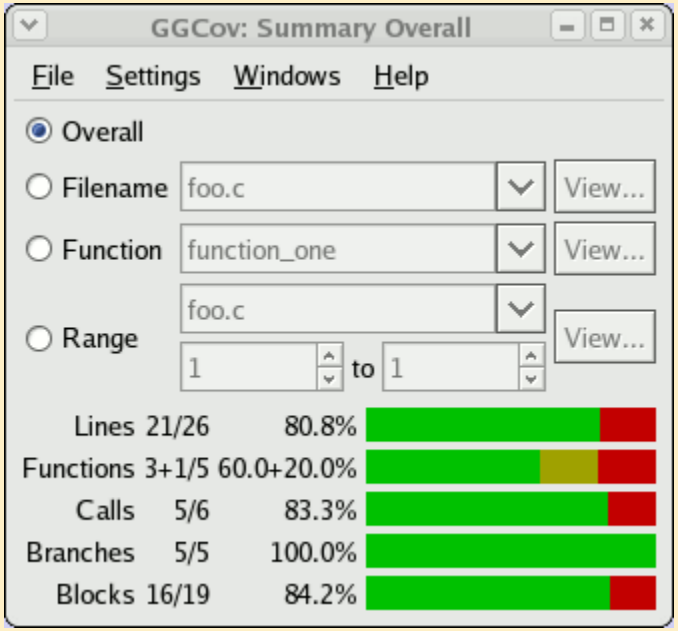
Példa: gcov annotált forráskód

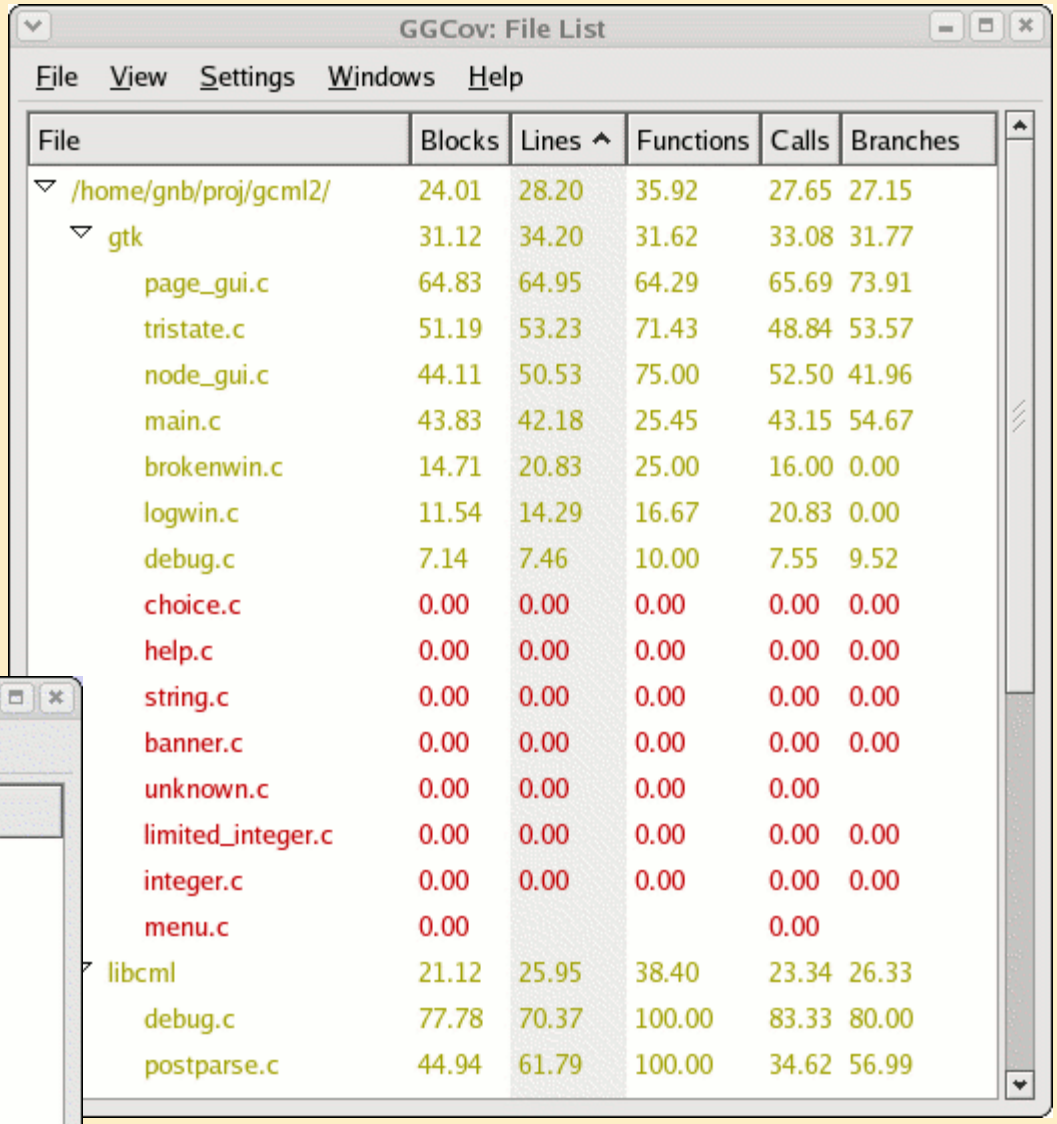
```
        #include <stdio.h>
        int main (void)
        {
     1      int i;

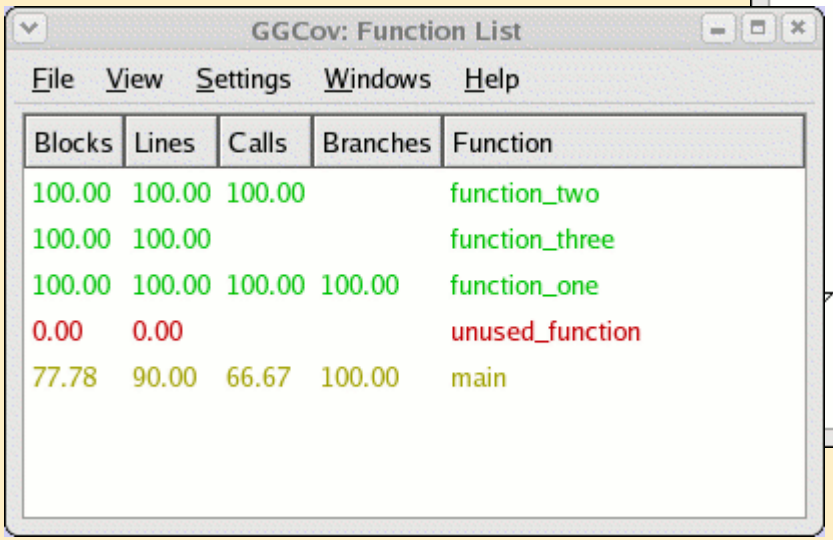
    10      for (i = 1; i < 10; i++)
             {
     9         if (i % 3 == 0)
     3          printf ("%d is divisible by 3\n",i);
     9         if (i % 11 == 0)
######          printf ("%d is divisible by 11\n",i);
     9        }

     1      return 0;
     1  }
```

Példa: ggcov elemzés

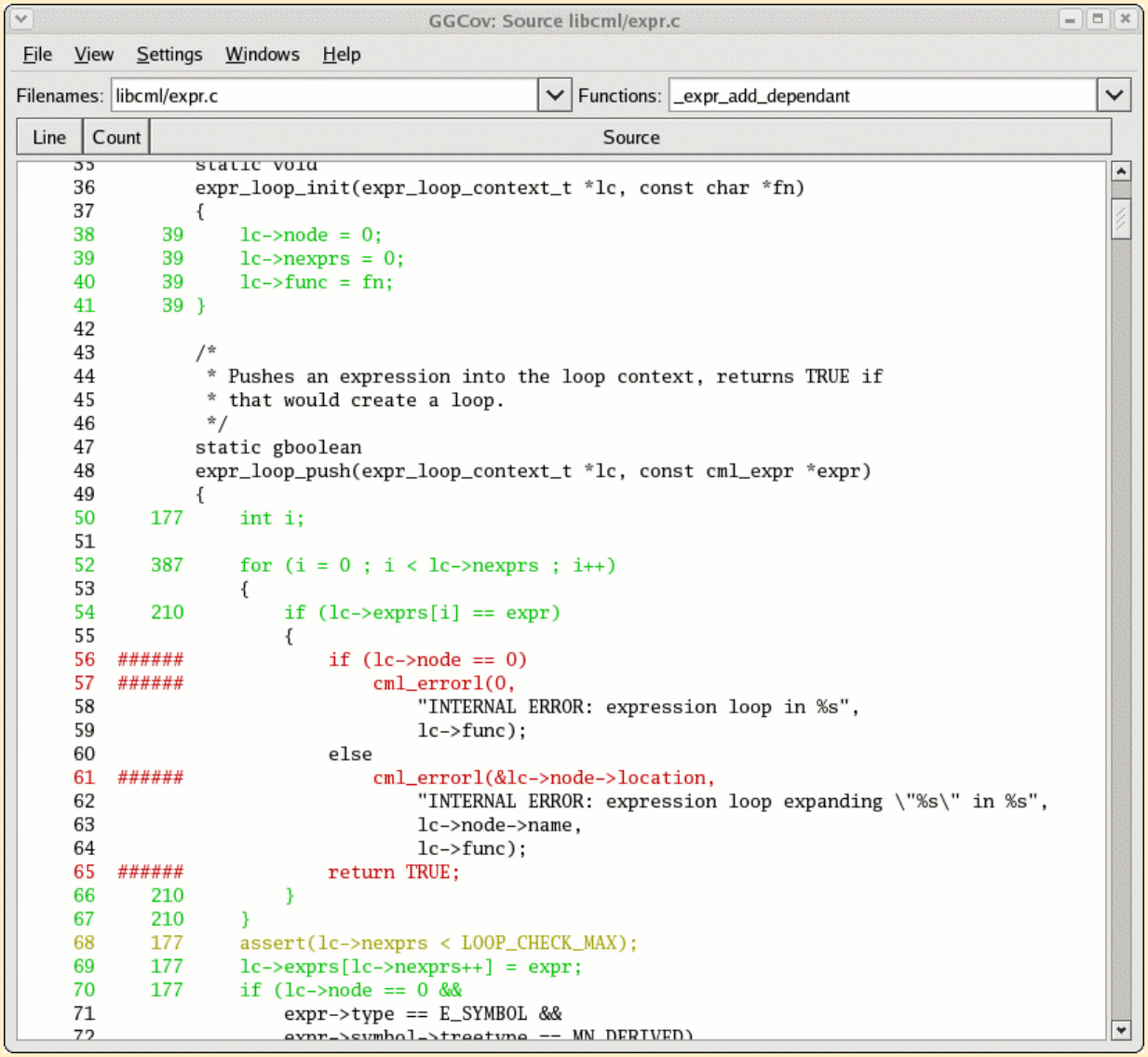






Példa: ggcov elemzés

- green - executed at least once
- yellow - part of the line executed
- red - never executed
- blue - suppressed
- black - no executable code



IPL Cantata++ architektúra