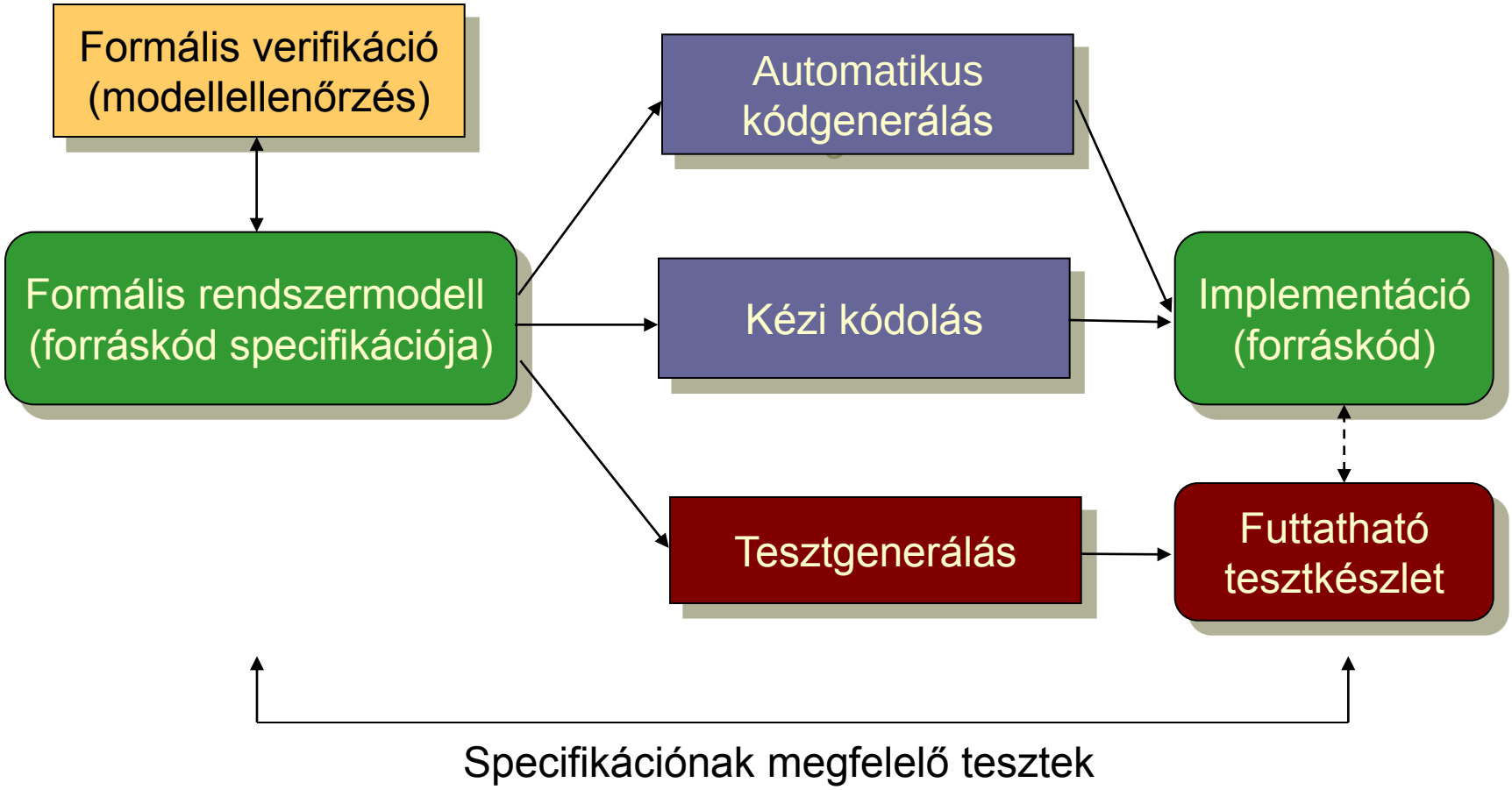
A modellellenőrzés érdekes alkalmazása: Tesztgenerálás modellellenőrzővel

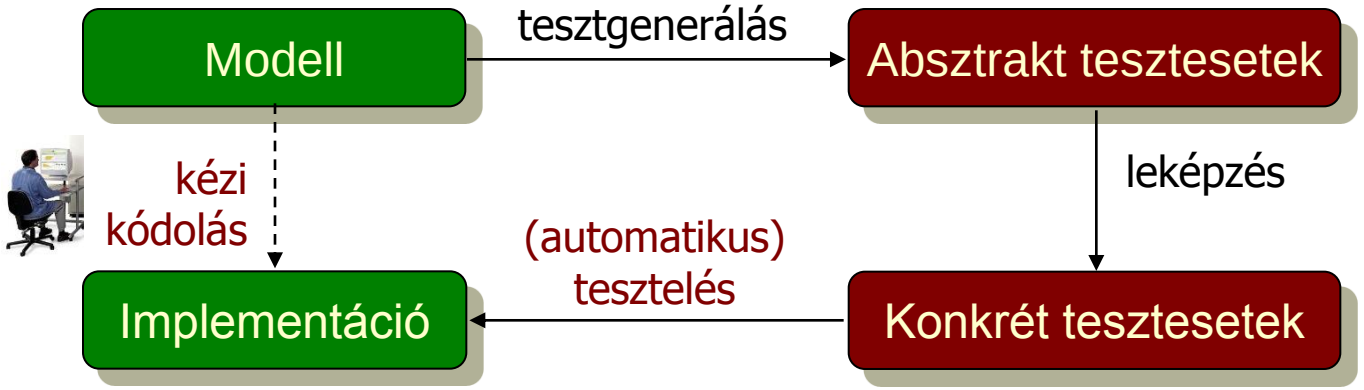
Majzik István
Micskei Zoltán
BME Méréstechnika és Információs Rendszerek Tanszék

Modell alapú fejlesztési folyamat (részlet)



Használati esetek

- Kézi kódolás esetén: Konformancia ellenőrzés

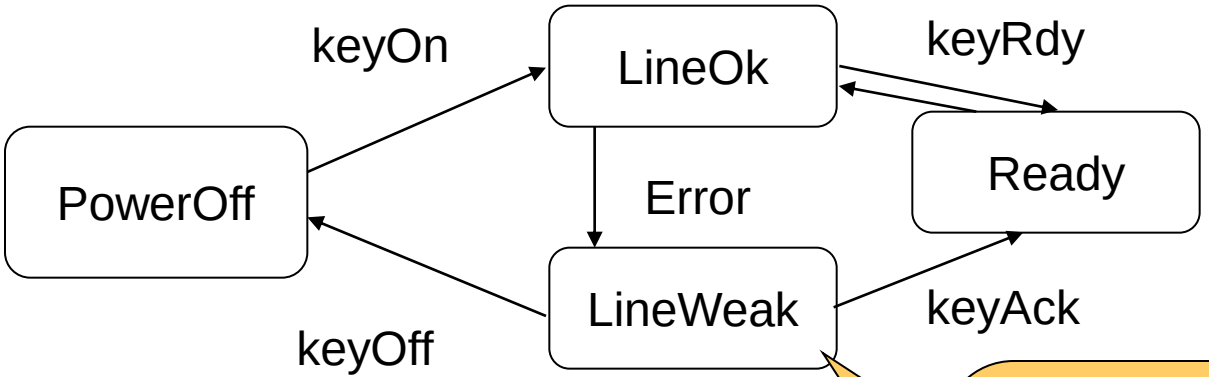


- Automatikus kódgenerálás esetén: Validáció

Előfeltételek

- Állapot alapú, eseményvezérelt működés
  - KS, LTS, KTS az alapszintű formalizmusok
- Fedési kritériumok szerinti tesztelés
  - Állapotfedés: A tesztekkel járjunk be minden állapotot
  - Átmenetfedés: A tesztekkel járjunk be minden átmenetet
- Alapötlet:
  - Egy teszt: Egy megfelelő állapottér bejárási szekvencia
  - Cél: A modellellenőrző járja be az állapotteret!
  - Irányítsuk úgy, hogy a modellellenőrző által generált ellenpélda legyen a teszteset
- Teszt elfogadhatósági kritérium
  - Modell mint referencia alapján származtatható

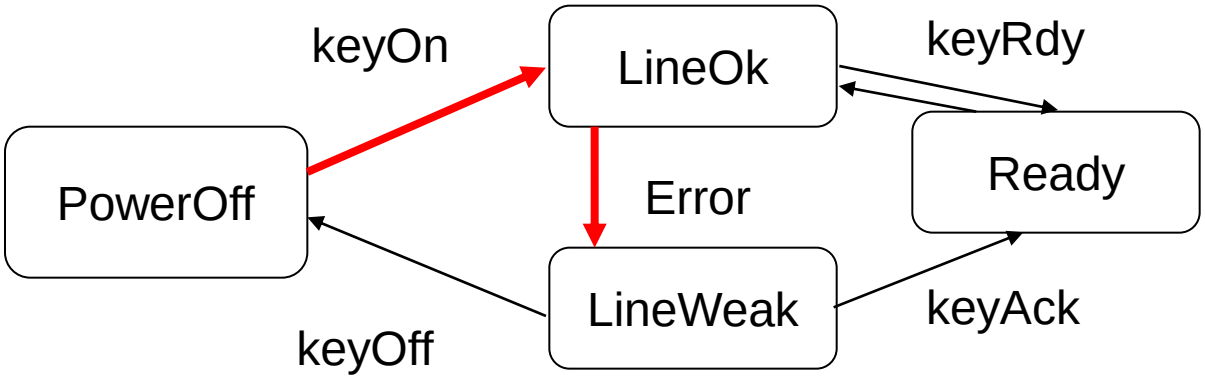
Hogyan használható a modell ellenőrző?



Kritérium megadása:
A LineWeak állapotot soha sem lehet elérni:
¬ EF LineWeak

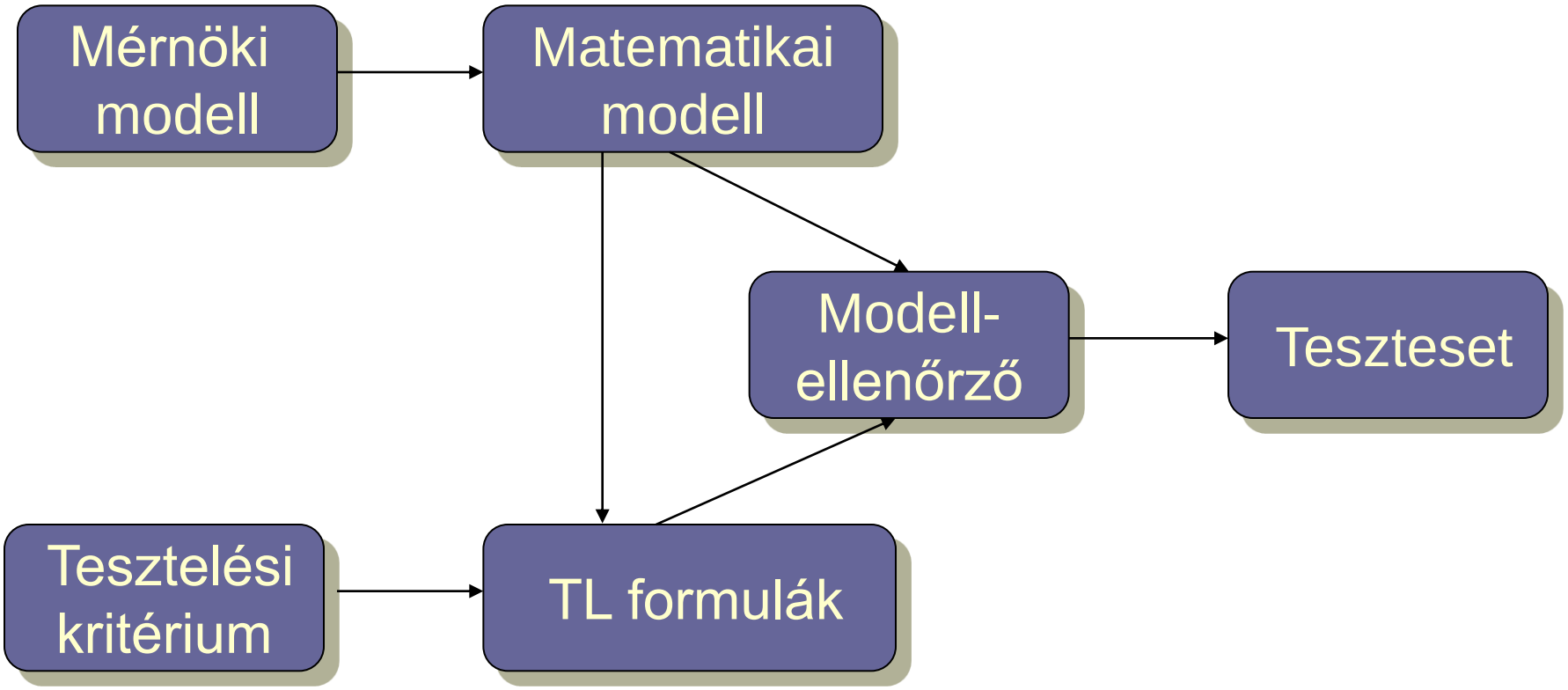
Hogyan használható a modell ellenőrző?

Az eszköz ezzel az ellenpéldával demonstrálja, hogy a követelmény nem teljesül, az állapot elérhető.

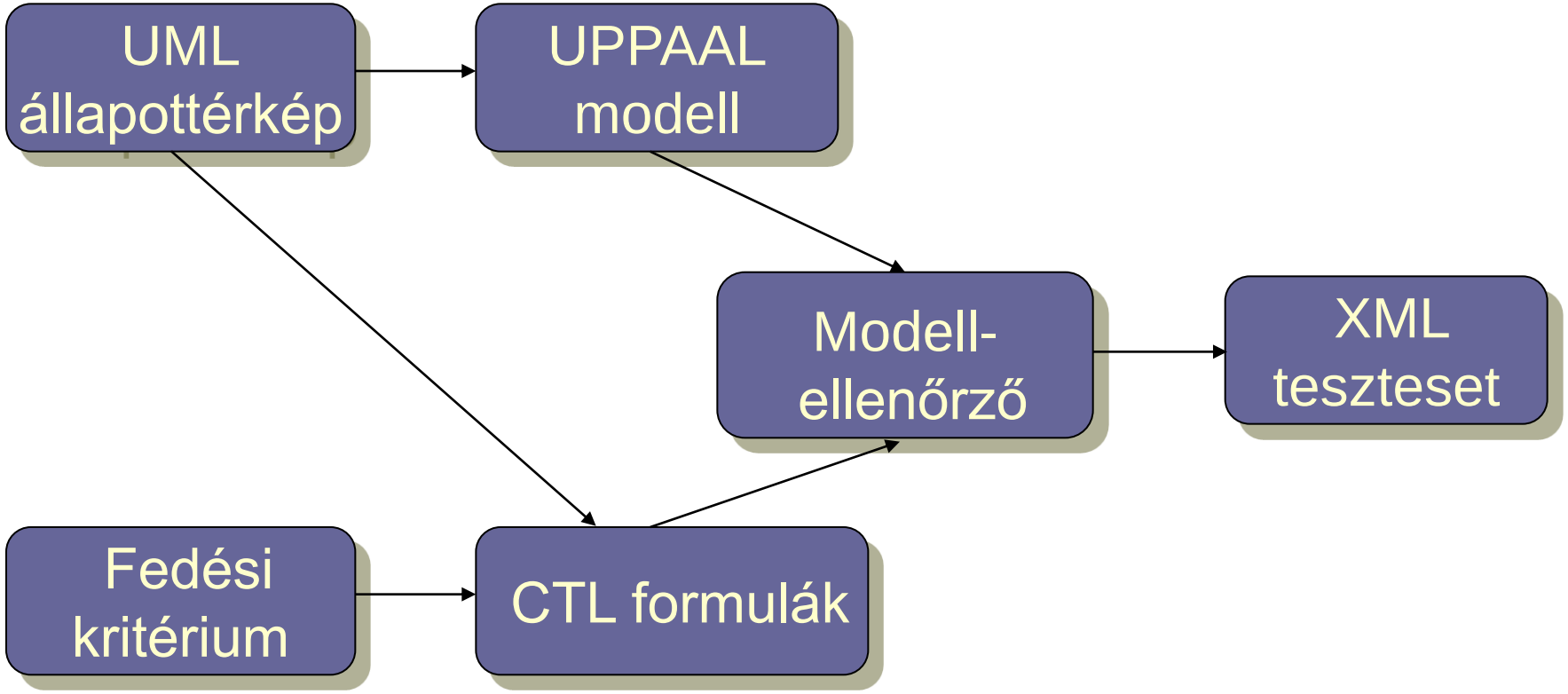


Ez viszont pontosan egy, a LineWeak állapotot lefedő teszteset!

Automatikus tesztgenerálás



Egy megvalósítás



Vezérlés alapú fedettségi kritériumok

Minden állapot és átmenet egyértelműen azonosítva:
Egyedi L(s) címke illetve (a) akció

- Állapot fedés: KS vagy KTS modellen

  Minden s állapotra: $\neg$EF L(s) vagy

  $\neg$EF (L(s) $\wedge$ EF start)

  start egy kezdőállapot címkéje,
  a következő teszt indításához

- Átmenet fedés: LTS vagy KTS modellen

  Minden t átmenetre, ahol (s,a,s')$\in\rightarrow$: $\neg$EF (a)

Korlátozások

- Modellellenőrző:
  - Csak egy ellenpéldát generál
  - Célja a hatékony állapottér bejárás:
    Nem feltétlenül a legrövidebb tesztesetet kapjuk!
  - Sok modellellenőrző konfigurálható:
    - Szélességi bejárás kérhető
    - Mélységi bejáráshoz mélységkorlát megadható
    - Rövidebb ellenpélda iteratívan kereshető
  - Legrövidebb illetve legkisebb tesztkészlet kiválasztása:
    NP-teljes probléma!
- Absztrakt és konkrét tesztesetek közt leképezés kell
  - Absztrakt teszt eset: A modell bejárása (ellenpélda)
  - Konkrét teszt eset: Hívási szekvencia adott teszt környezetben
- Nemdeterminisztikus modellek esetén nehézségek

Példa: Tesztgenerálási eredmények (állapotfedés)

Options (compile or run-time)	Time required for test generation	Length of the test sequences	Longest test sequence
-i	22m 32.46s	17	3
-dBFS	11m 48.83s	17	3
-i -m1000	4m 47.23s	17	3
-I	2m 48.78s	25	6
default	2m 04.86s	385	94
-I -m1000	1m 46.64s	22	4
-m1000+	1m 25.48s	97	16
-m200 -w24	46.7s	17	3

Paraméterek a SPIN modellellenőrzőhöz:

-i iteratív, -I közelítő iteratív

-dBFS: szélességi keresés

-m mélységi keresés korlátja

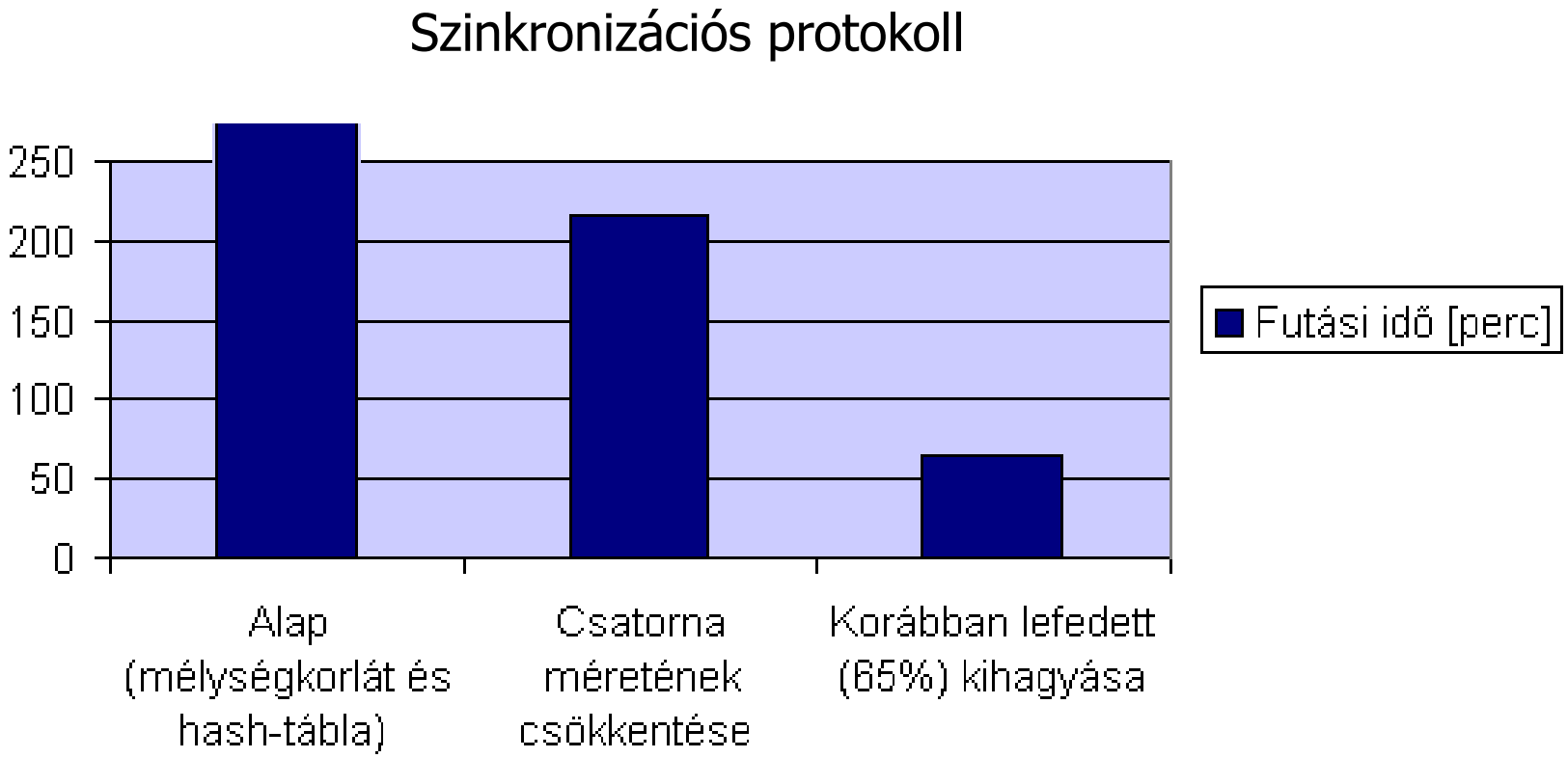
-w hash tábla korlátja

- Mobiltelefon vezérlését leíró modell
- 10 állapot, 21 átmenet

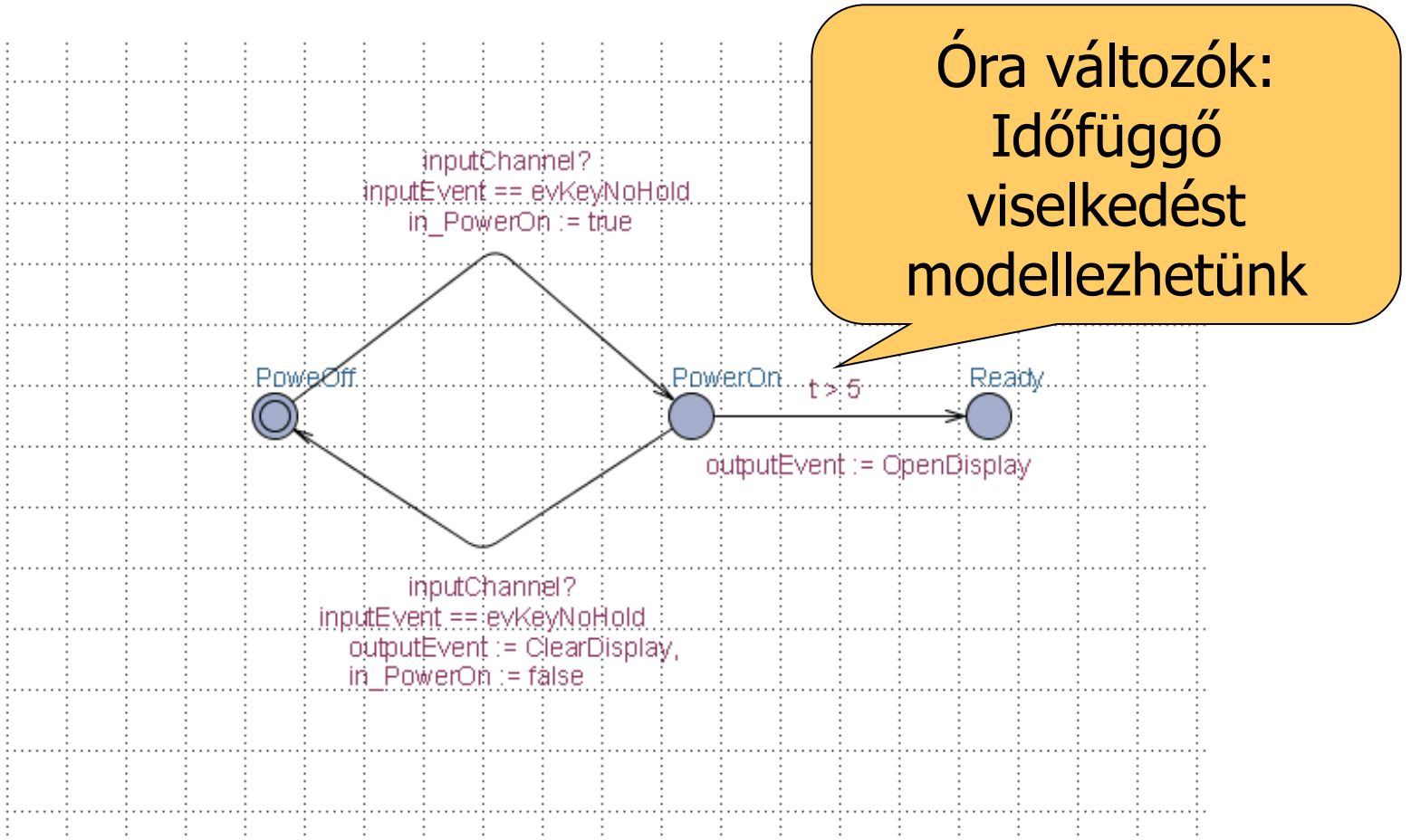
Példa: Szinkronizációs protokoll tesztelése

- Bitek szinkronizálása egy elosztott rendszerben
  - 5 objektum, 31 állapot, 174 átmenet
  - 2e+08 bejárandó állapot
- Más technikák is kellenek:
  - Mélységkorlát bevezetése
  - Szűkítések a modellben:
    - FIFO kommunikációs csatorna méretének korlátozása
  - Korábban lefedett kritériumok kihagyása
- További heurisztikák alkalmazása:
  - Mélyen fekvő állapotok lefedése előbb
  - Állapottér levágása

Példa: Teljes állapotfedésű tesztek



Kiterjesztés valósidejű rendszerekre



- Időzített automaták használata
- Modellellenőrző: UPPAAL

Példa: Generált tesztek

```
State:
( input.sending mobile.PowerOn mobile1.LineOK
  mobile2.CallWait )
t=0 inputEvent=28 outp                          #depth=5

Delay: 6

State:
( input.sending mobile.PowerOn mobile1.LineOK
  mobile2.CallWait )
t=6 inputEvent=28 outputEvent=14 in_PowerOn=1 #depth=5

Transitions:
input.sending->input.sendInput { 1, inputChannel!, 1 }
mobile2.CallWait->mobile2.VoiceMail { inputEvent ==
  evKeyYes && t > 5 && in_PowerOn, inputChannel?, 1 }
```

A teszt időzítési viszonyok is szerepelnek a generált tesztesetben

Tesztgenerálás SAT alapú modellellenőrzővel

- Logikai függvény konstruálása:
  - Állapottér kibontása k lépésben a kezdőállapotból
  - Teszt kritérium megadása: TG formula, pl.:
    - Adott állapot elérése
    - Adott állapotátmenet végrehajtása
    - Adott modellrészlet bejárása, …

FQL nyelv teszt célokhoz (FSHELL):
in /code.c/
cover @line(6),@call(f1)
passing @file(c1.c) \ @call(f2)

$$\mathrm{SAT}\left(I(s^0) \wedge \mathrm{path}(s^0, s^1, \ldots, s^k) \wedge TG \right)$$

Kezdő-állapot | Állapottér kibontása | Teszt cél

- Ha található behelyettesítés, akkor az egy tesztet ad:
  - A teszt teljesíti a TG kritériumot
  - A legrövidebb teszt 0-ról induló iteráció során található meg

Összefoglalás

- Tesztgenerálás fedési kritériumokhoz
  - Állapotok fedése
  - Átmenetek fedése
  - …
- Klasszikus modellellenőrzők használata
  - Fedési kritériumhoz temporális logikai kifejezéskészlet
  - Ellenpéldák adják a teszt eseteket
  - A modellellenőrző konfigurálása fontos
- SAT alapú (korlátos) modellellenőrző használata
  - Fedési kritérium (teszt cél) mint predikátum
  - SAT eredménye (behelyettesítés) adja a teszt esetet