A modellellenőrzés alkalmazásai: Szintézis és tesztgenerálás modellellenőrzővel

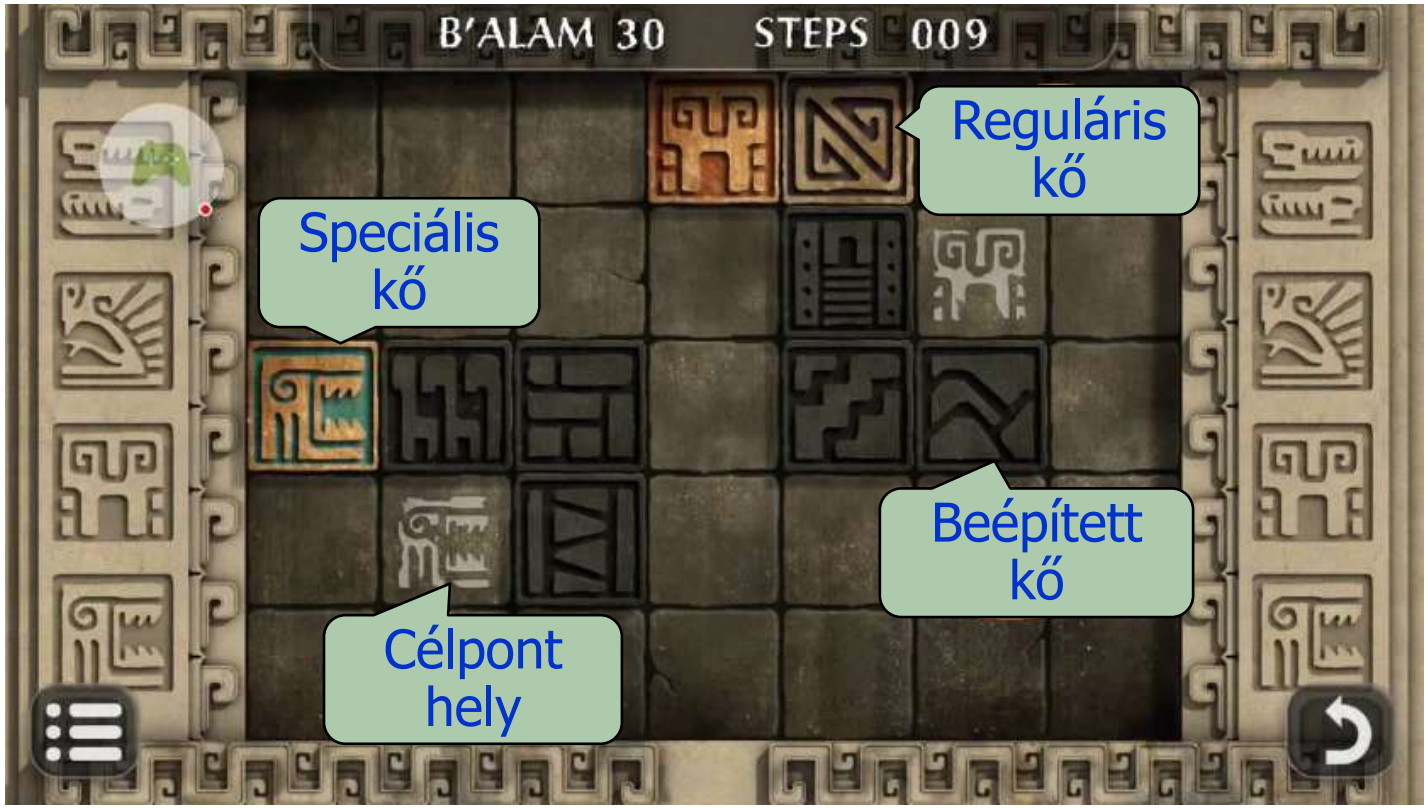
dr. Majzik István
dr. Micskei Zoltán
BME Méréstechnika és Információs Rendszerek Tanszék

Szintézis feladatok megoldása modellellenőrzővel

Bevezetés: Szintézis feladat

- Cél: Egy lépéssorozat meghatározása a probléma modelljében
  - Gyártási folyamat: milyen lépésekkel készül el a termék?
  - Protokoll, számítás: milyen lépésekkel terminál?
  - Játék, rejtvény: milyen lépésekkel van megoldás?
- Modellellenőrző: minden lehetséges lépéssorozatot vizsgál; ezek közül kell a számunkra érdekeset „lekérni”
  - Ha ellenpélda keresés a modellellenőrző célja:
    A keresett lépéssorozat legyen megfeleltethető egy várt ellenpéldának
  - Ha tanú (witness) keresés a modellellenőrző célja:
    A keresett lépéssorozat elérhetősége legyen vizsgálva
- Egyszerű esetek: Lépéssorozat célállapotának megadásával
  - Lépéssorozat generálás ellenpélda kereséssel: $\neg$EF <célállapot>
  - Lépéssorozat generálás tanú kereséssel: EF <célállapot>
- Legrövidebb, leggyorsabb lépéssorozat meghatározása
  - Ha az ellenpélda vagy tanú elvárt tulajdonsága beállítható
  - Pl. UPPAAL: legrövidebb vagy leggyorsabb trace

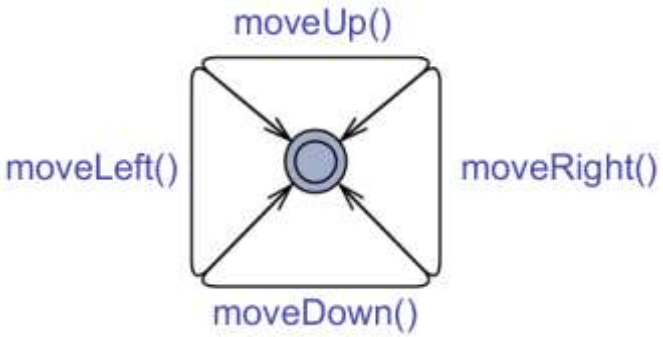
Mintapélda: A Cryptica játék megoldása



- Cél: Speciális kövek a célpontjaikba kerüljenek
- Egy lépés: Minden speciális és reguláris követ (amit lehet) balra vagy jobbra vagy felfelé vagy lefelé kell mozgatni
  - A pálya széle vagy beépített kő egy-egy követ megállít, de a többi kő mozgatását nem hiúsítja meg

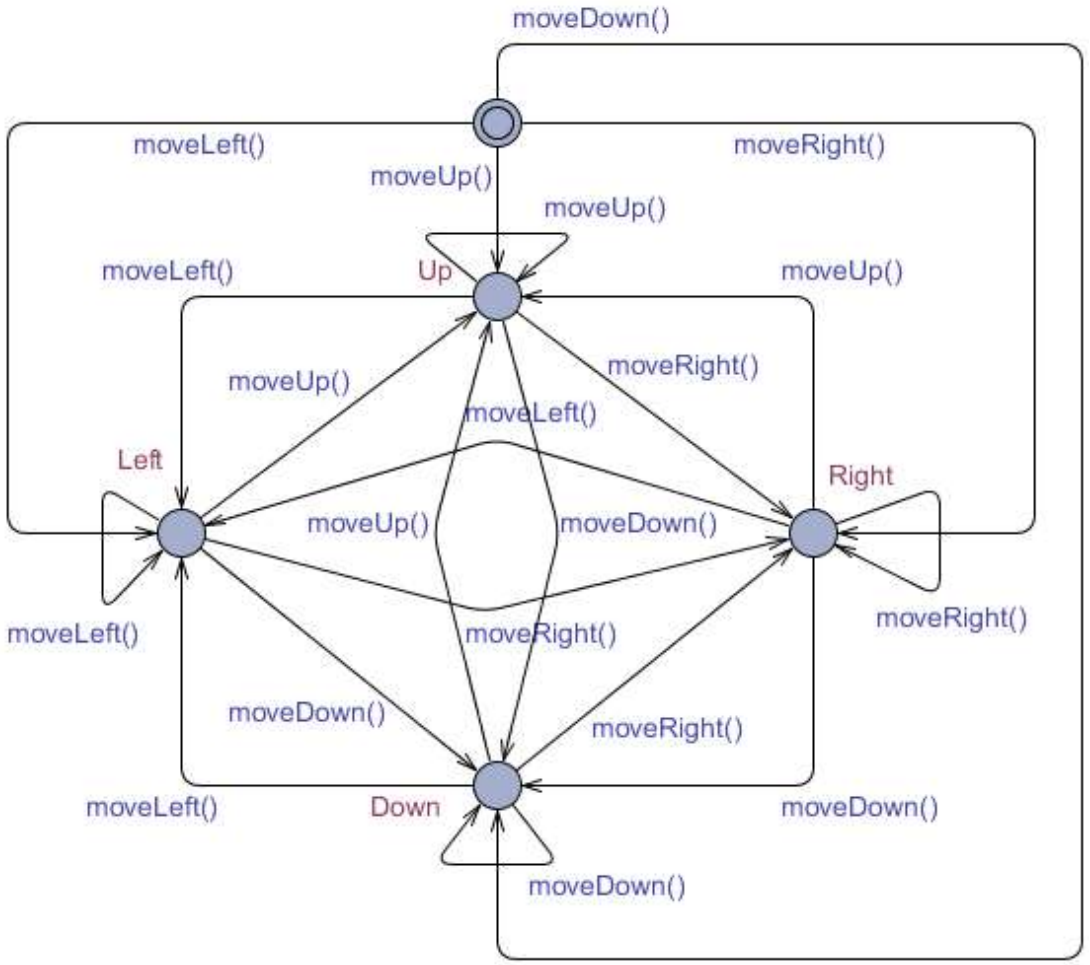
Mintapélda: A Cryptica játék modellje

- Adatstruktúrák
  - Pálya modellje: Beépített reguláris kövek és célpontok
  - Játékállás modellje: Speciális és reguláris kövek elhelyezkedése
- Dinamikus működés: Lehetséges lépések felvétele
  - Lépés balra / jobbra / felfelé / lefelé: játékállást módosító függvények a pálya modellje alapján
  - A modellben megjelenik minden lehetséges lépés (a kívánt lépéssorozatot „keresi” a modellellenőrző)
  - Legegyszerűbb automata:



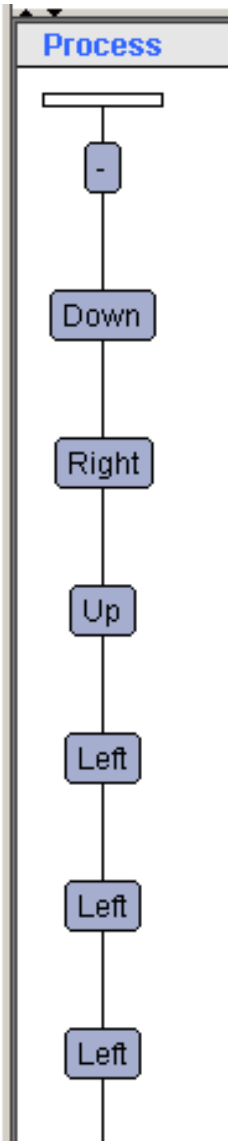
Mintapélda: A Cryptica játék modellje

- Célszerűbb modell: Lépések követhetővé tétele
  - A lépést jelzi a lépéssel elért állapot neve



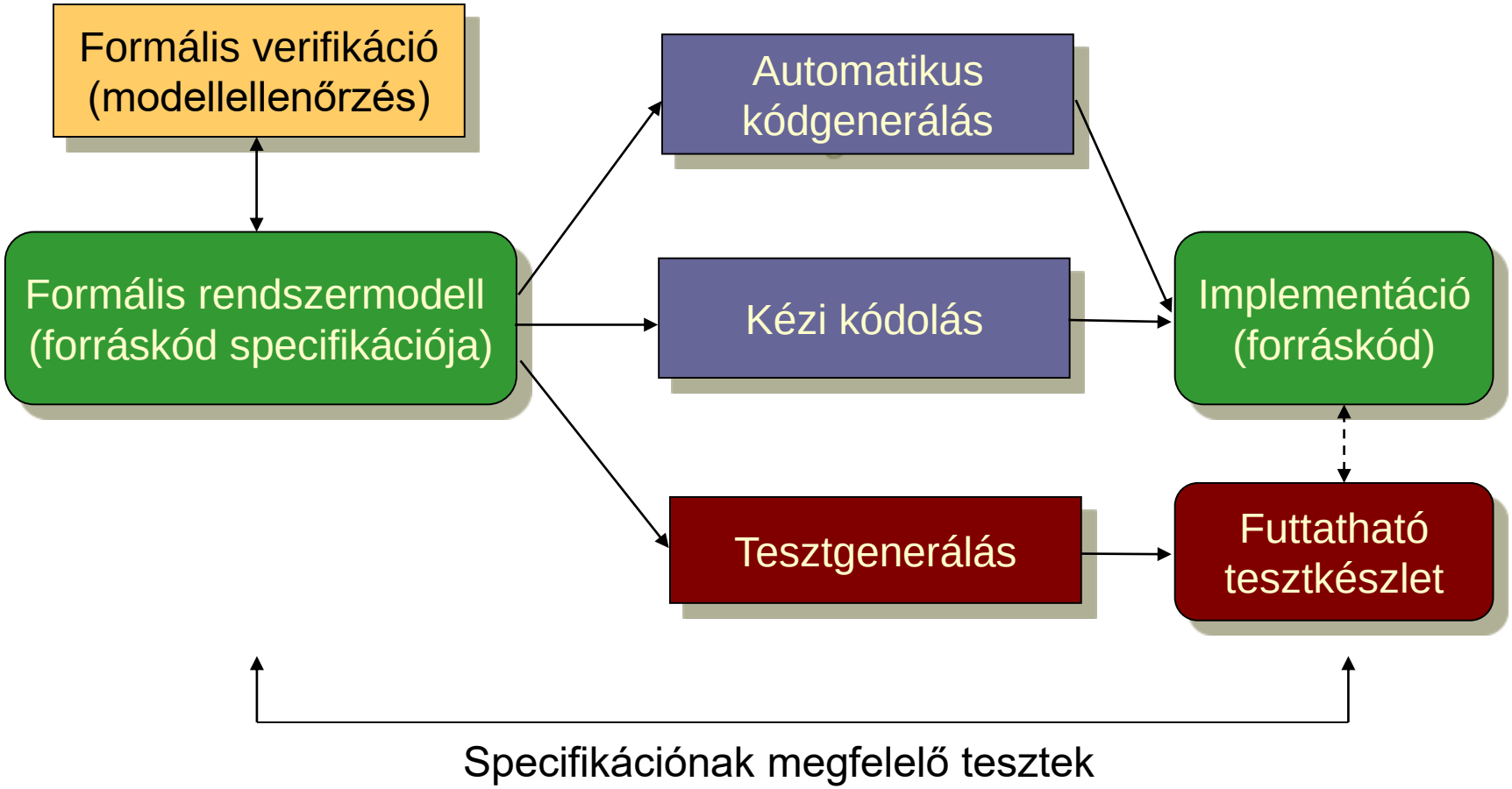
Mintapélda: Cryptica játék modellellenőrzése

- Modellellenőrzés: Kirakható-e a játék? Legkevesebb hány lépésben kerülhetnek helyükre a speciális kövek?
  - Célállapot meghatározás: isCorrectDistribution() logikai függvény (csak akkor igaz, ha a speciális kövek a helyükön vannak)
  - Temporális logikai követelmény: E<> isCorrectDistribution()
  - A legrövidebb tanú trace keresése
- A kirakás lépései a szimulátorban megjelenített állapotszekvencián követhetők



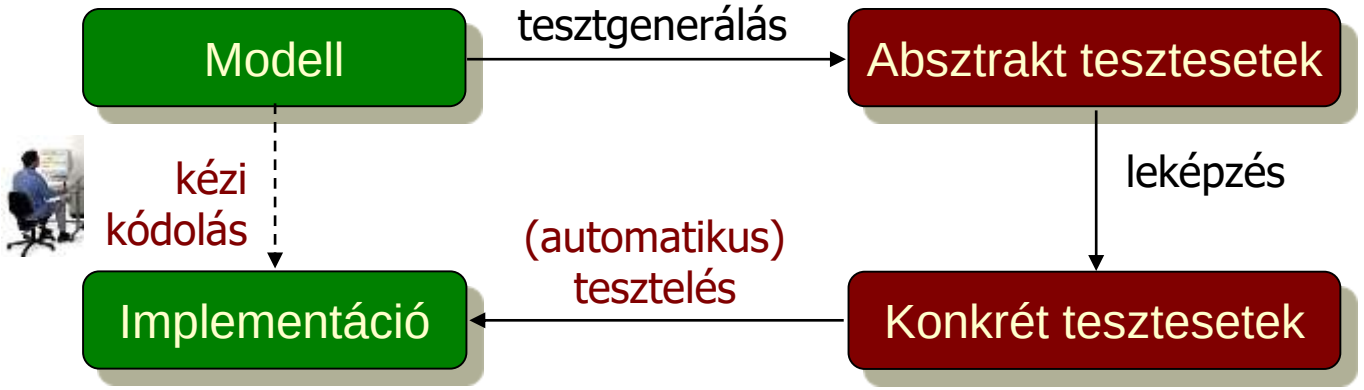
Tesztgenerálás modellellenőrzővel

Tesztgenerálás: Szerepe a modell alapú fejlesztésben



Tesztgenerálás: Használati esetei

- Kézi kódolás esetén: Konformancia ellenőrzés



- Automatikus kódgenerálás esetén: Validáció

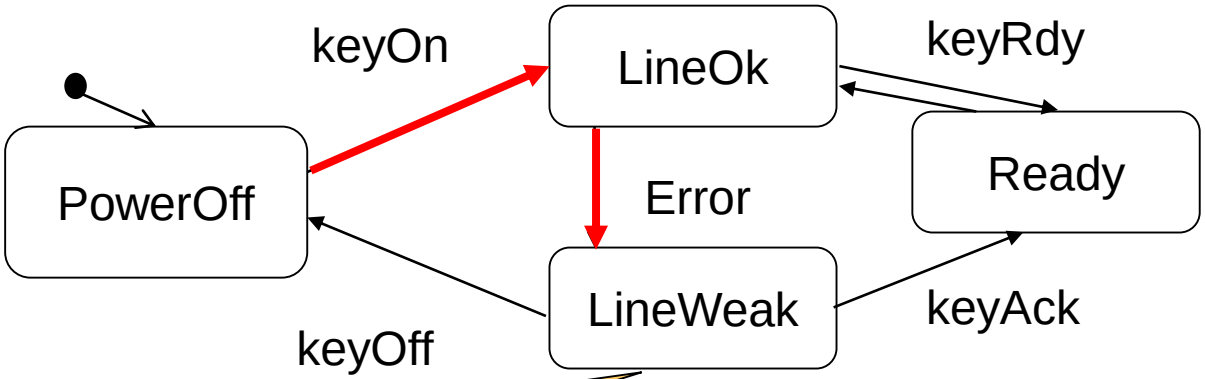
Előfeltételek és használat

- Állapot alapú, eseményvezérelt működés
  - KS, LTS, KTS, TA 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 lépéssorozat az állapottérben
  - Cél: A modellellenőrző keresse ezt az állapottérben
  - Irányítsuk úgy, hogy a modellellenőrző által generált ellenpélda legyen a teszteset
- Teszt elfogadhatósági kritérium
  - Modell mint referencia alapján származtatható

Hogyan használható a modellellenőrző?

- Állapot fedettség: LineWeak állapotra

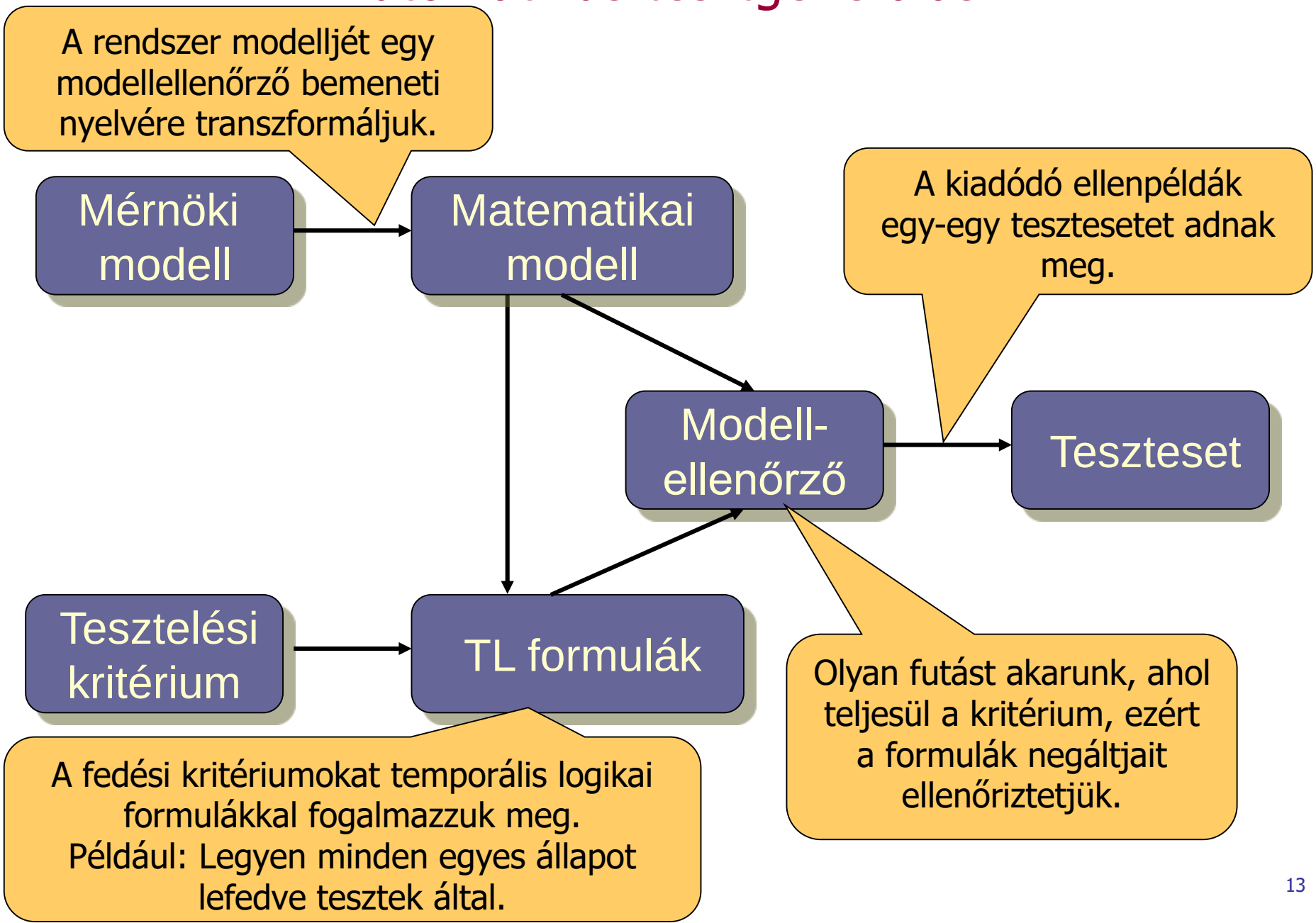
A modellellenőrző ellenpéldával demonstrálja, hogy a kritérium nem teljesül, az állapot elérhető.



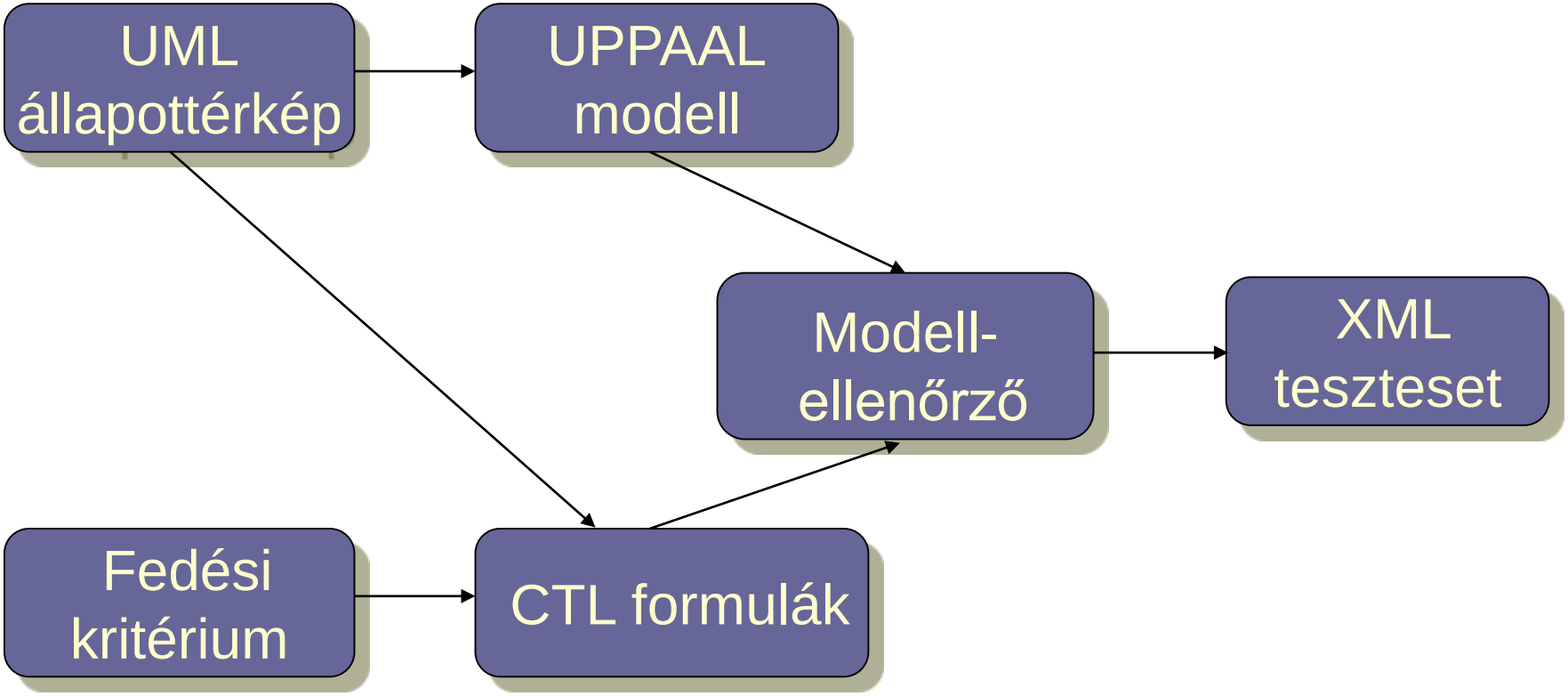
Kritérium megadása: A LineWeak állapotot soha sem lehet elérni:
¬ EF LineWeak

Ez viszont pontosan egy, a LineWeak állapotot lefedő teszteset!

Automatikus tesztgenerálás



Egy megvalósítás



Vezérlés alapú fedettségi kritériumok

Minden állapot illetve átmenet 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 megfelelő állapot 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
  - Általános célja a hatékony állapottér bejárás:
    Nem feltétlenül a legrövidebb tesztesetet kapjuk!
  - Sok modellellenőrző konfigurálható:
    - Legrövidebb ellenpélda kérhető
    - Szélességi bejárás kérhető (így rövid teszteset lesz)
    - Mélységi bejáráshoz mélységkorlát megadható (teszt hossz korlát)
    - Rövidebb ellenpélda iteratívan kereshető (hosszadalmas lehet)
  - Legrövidebb illetve minimális tesztkészlet kiválasztása: NP-teljes
- 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
-i	22m 32.46s	17
-dBFS	11m 48.83s	17
-i -m1000	4m 47.23s	17
-I	2m 48.78s	25
default	2m 04.86s	385
-I -m1000	1m 46.64s	22
-m1000+	1m 25.48s	97
-m200 -w24	46.7s	17

- Mobiltelefon vezérlését leíró modell
- 10 állapot, 21 átmenet

Paraméterek a SPIN modellellenőrzőhöz:

-i iteratív, -I közelítő iteratív

-dBFS: szélességi keresés

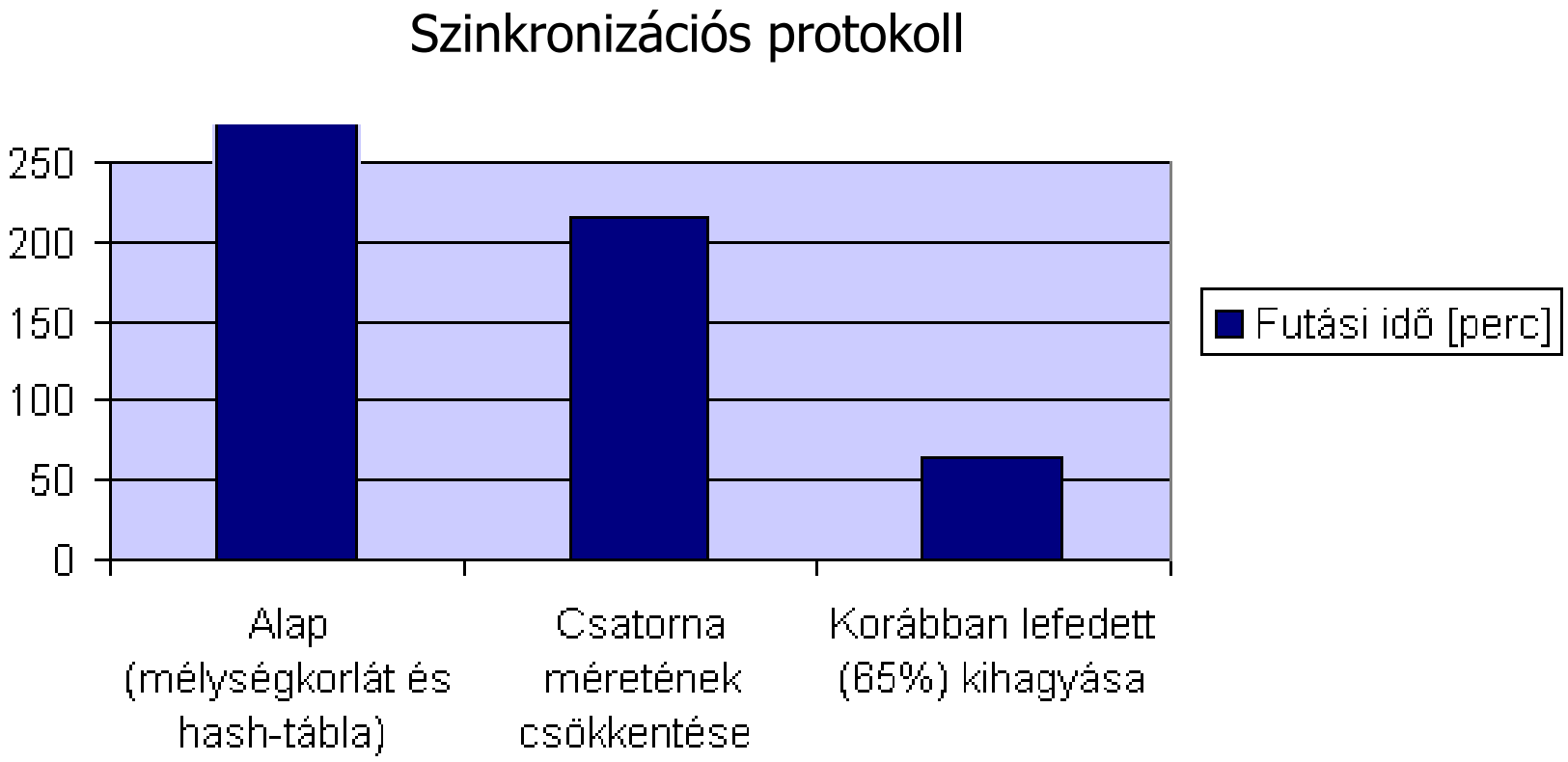
-m mélységi keresés korlátja

-w hash tábla korlátja

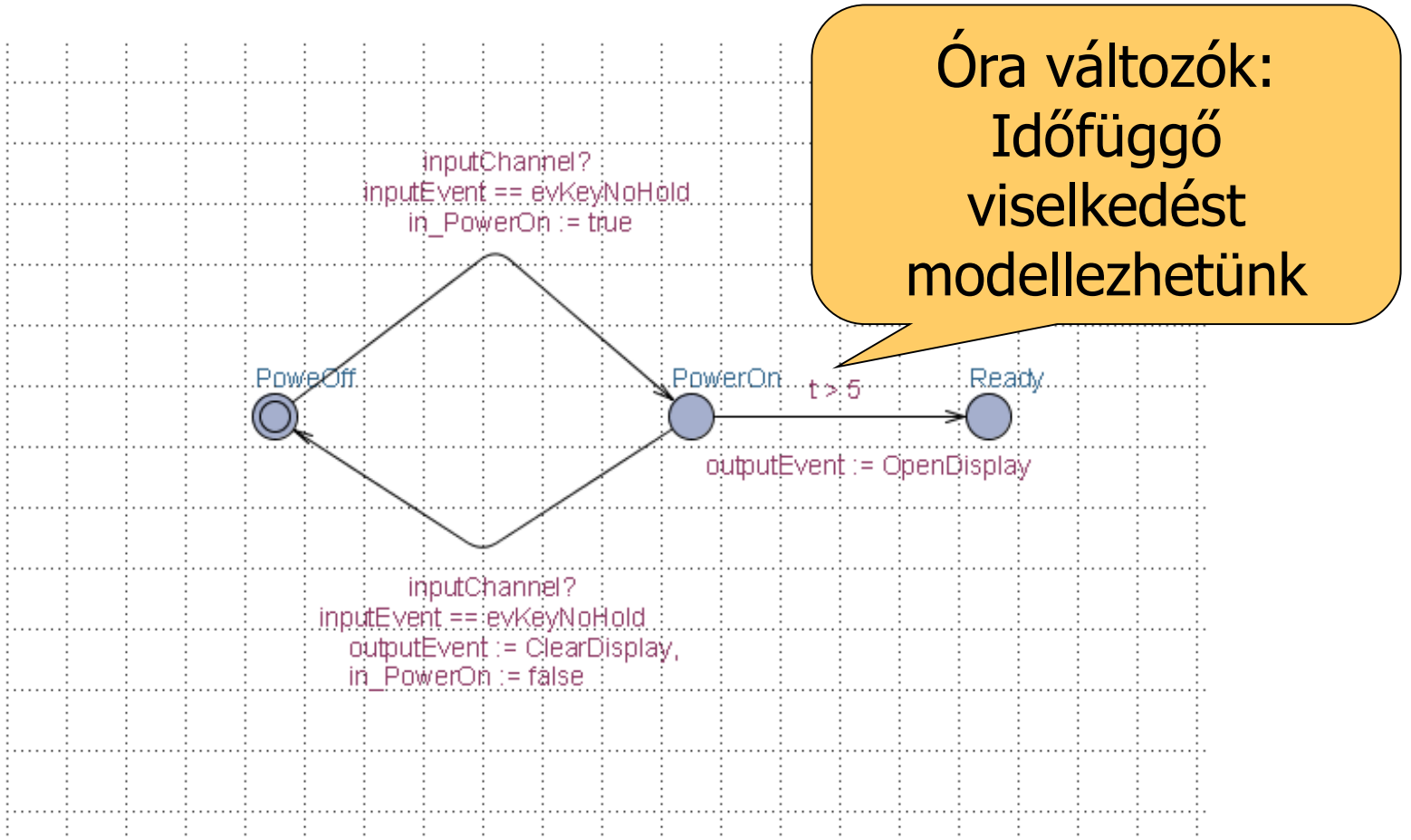
Példa: Szinkronizációs protokoll tesztelése

- Bitek szinkronizálása egy elosztott rendszerben
  - 5 objektum, 31 állapot, 174 átmenet
  - 2e+08 bejárandó állapot
- Más technikák is kellenek:
  - Mélységkorlát bevezetése a kereséshez
  - 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 korlátos 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, …

FSHELL FQL nyelv teszt célokhoz:
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

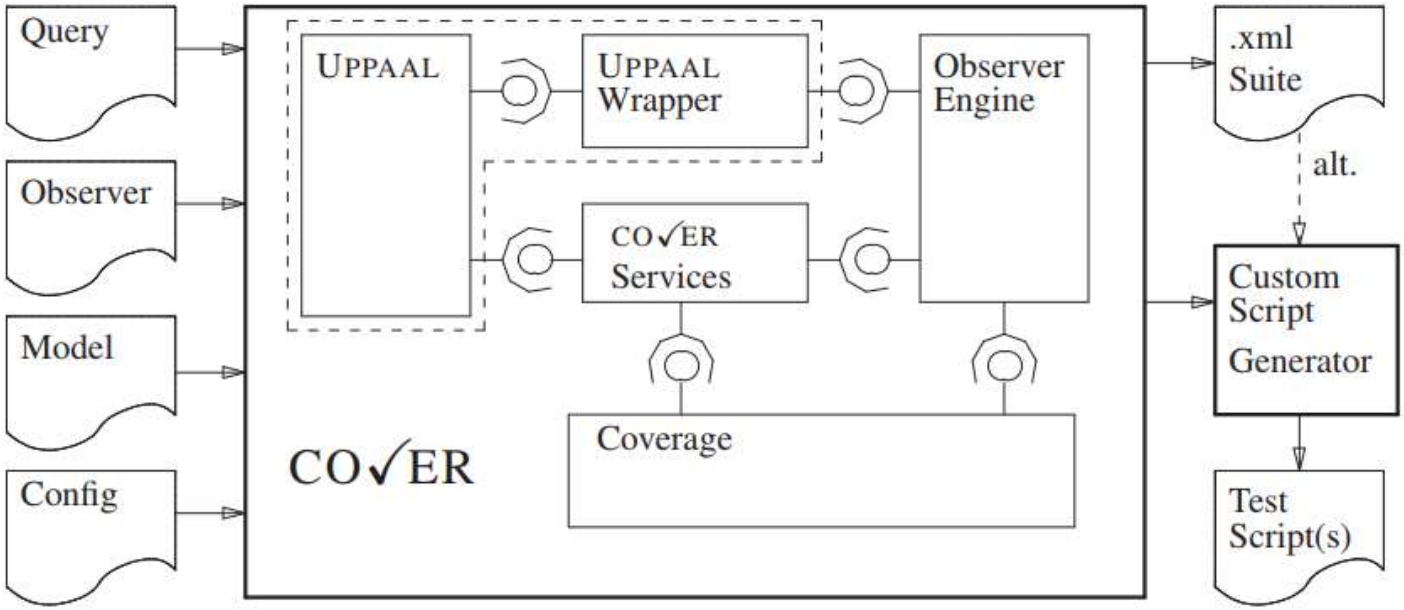
Állapottér kibontása

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

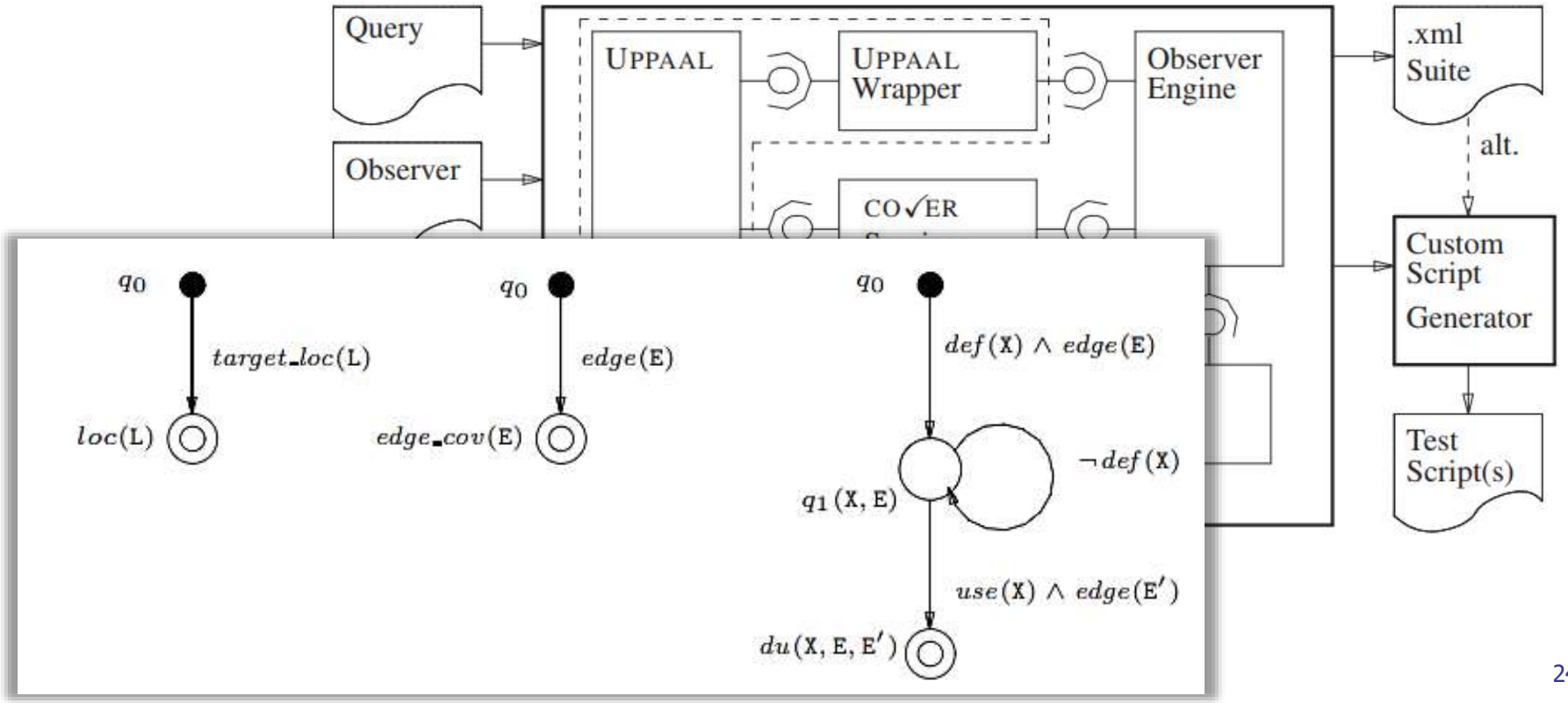
Példa eszközök: UPPAAL CoVer

- UPPAAL COVER
  - „UPPAAL CoVer is a tool for creating test suites from UPPAAL models with coverage specified by coverage observers a.k.a. observer automata.”



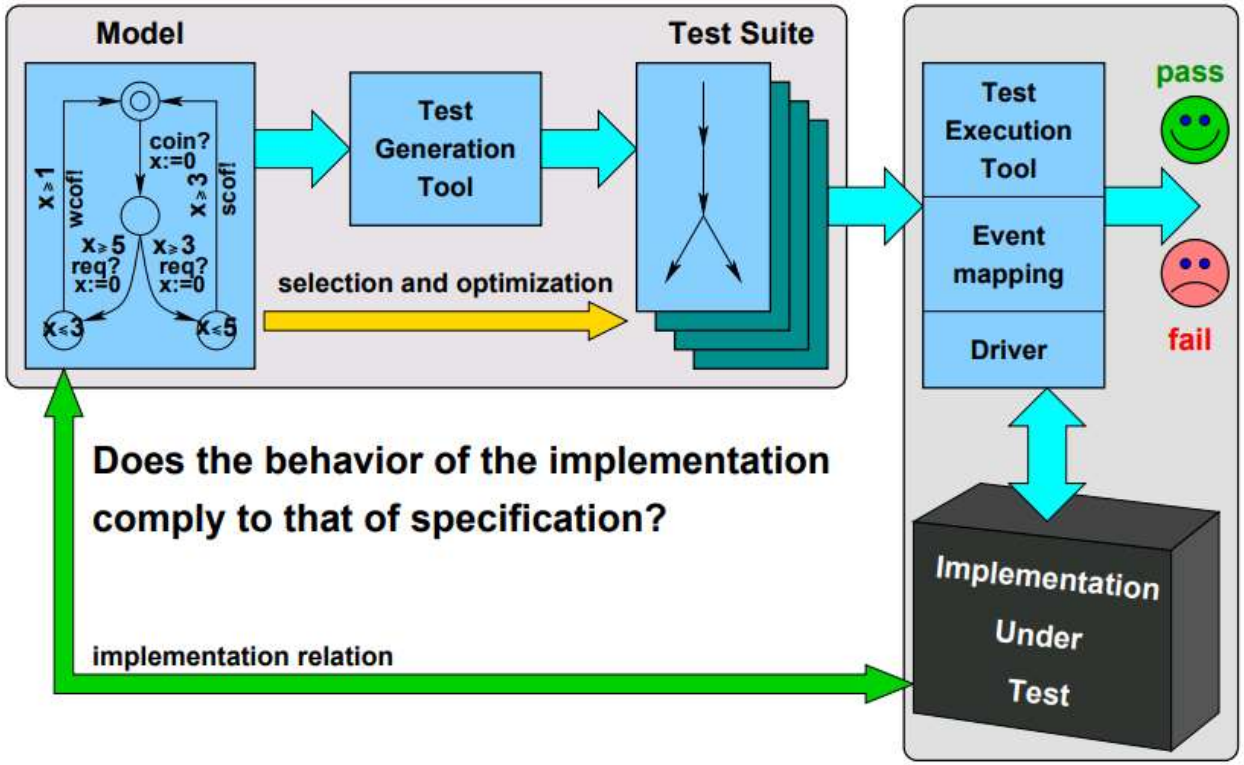
Példa eszközök: UPPAAL CoVer

- UPPAAL COVER
  - „UPPAAL CoVer is a tool for creating test suites from UPPAAL models with coverage specified by coverage observers a.k.a. observer automata.”



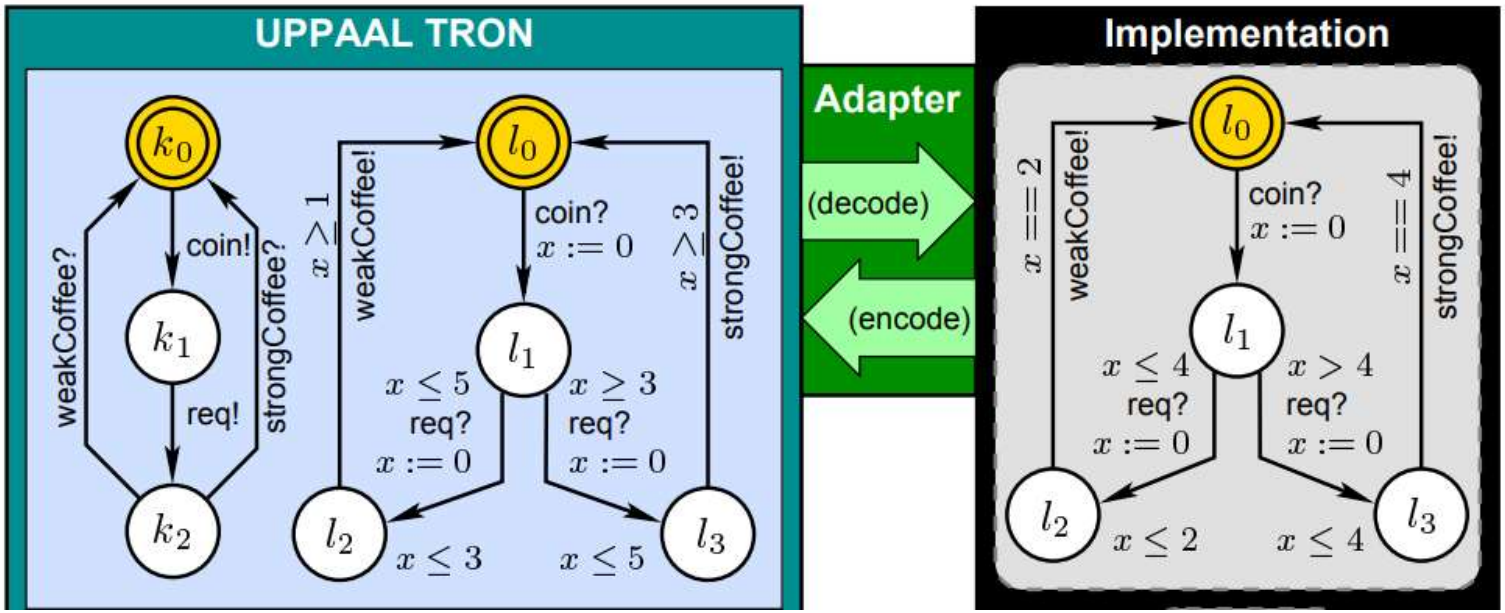
Példa eszközök: UPPAAL TRON

- UPPAAL TRON: Testing Real-time Systems Online
  - „UPPAAL TRON is a testing tool, based on UPPAAL engine, suited for black-box conformance testing of timed systems, mainly targeted for embedded software commonly found in various controllers.
  - By online we mean that tests are derived, executed and checked simultaneously while maintaining the connection to the system in real-time.”



Példa eszközök: UPPAAL TRON

- UPPAAL TRON: Testing Real-time Systems Online
  - „UPPAAL TRON is a testing tool, based on UPPAAL engine, suited for black-box conformance testing of timed systems, mainly targeted for embedded software commonly found in various controllers.
  - By online we mean that tests are derived, executed and checked simultaneously while maintaining the connection to the system in real-time.”



Relativized Timed I/O Conformance

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