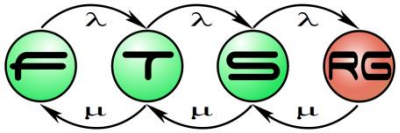
Futásidőbeli verifikáció

Szoftverellenőrzési technikák előadás

Majzik István
Budapesti Műszaki és Gazdaságtudományi Egyetem
Méréstechnika és Információs Rendszerek Tanszék



Tartalomjegyzék

- Definíció és motiváció
- Használati esetek
- Futásidőbeli verifikációs technikák
  - Verifikáció referencia automata alapján
  - Verifikáció temporális logikai követelmények alapján
  - Verifikáció LSC követelmények alapján
  - Verifikáció scenario követelmények alapján
- Implementációs tapasztalatok

Mit jelent a futásidejű verifikáció?

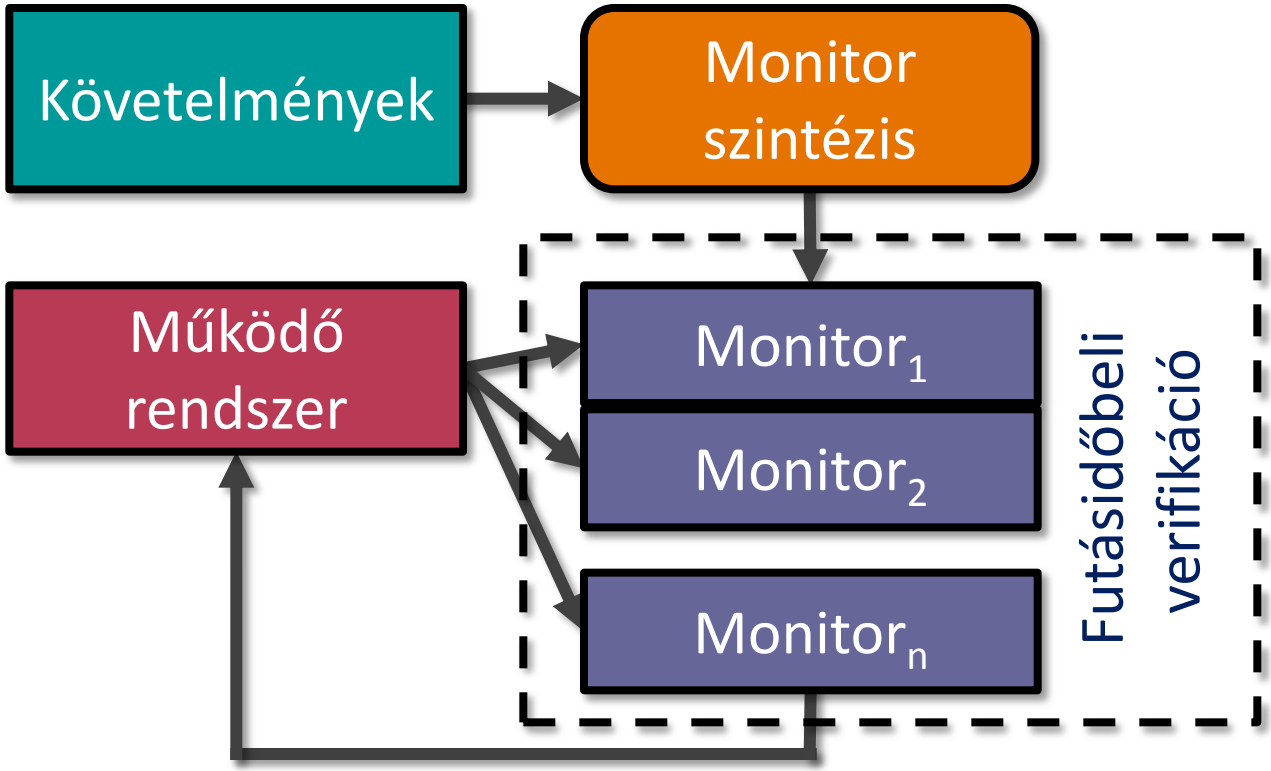
- Definíció:
  - Rendszerek viselkedésének ellenőrzése
  - működés közben (on-line),
  - formálisan specifikált követelmények alapján
- Motiváció
  - Megbízhatósági és biztonsági elvárások
    - IT infrastruktúra: Szolgáltatási szint szerződések (SLA)
    - Biztonságkritikus rendszer: Elviselhető hibagyakoriság (THR)
  - Futásidőbeli hibák elkerülhetetlenek
    - Hardver komponensek véletlen hibái
    - Szoftver tervezési, implementációs, konfigurálási hibák

Cél: Futásidőbeli hibadetektálás

- Futásidőbeli hibadetektálás a hibakezelés alapja
  - Klasszikus: Hibadetektálás a forráskód (referencia) alapján
    - Pl. CFG ellenőrzés (watchdog processzorok)
    - Csak működési hibákra, implementáció alapján
  - Ellenőrzés a követelmények alapján
    - Szisztematikus (tervezési, kódolási, konfigurációs) hibákra is
  - Futásidőbeli verifikáció formalizált követelmények alapján
    - Precíz követelmény megfogalmazás
    - Automatikus ellenőrző (monitor) szintézis lehetősége
- Példa: Reaktív hibakezelés
  - Hibadetektálás majd beavatkozás (pl. helyreállítás, biztonságos állapot beállítása, …)

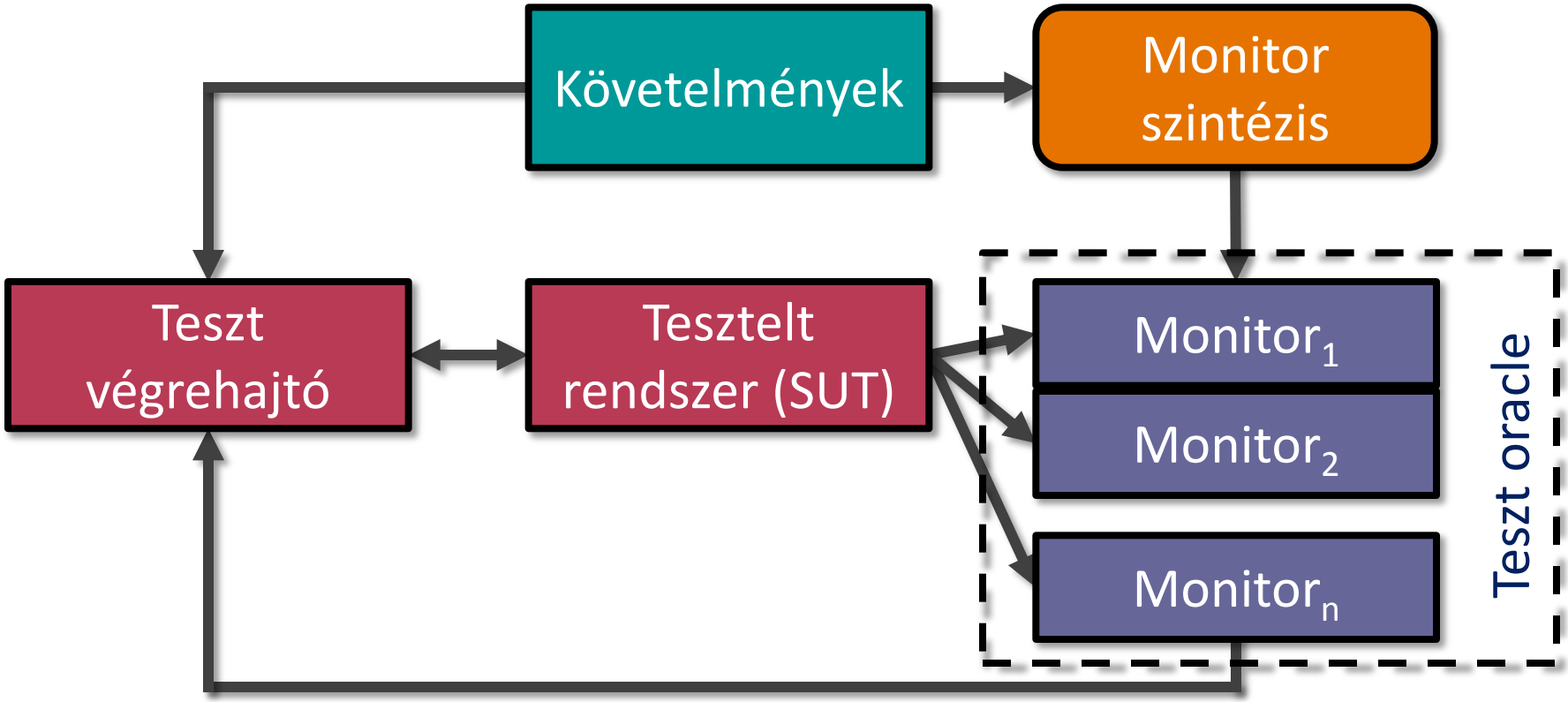
Használati eset 1: Futásidőbeli verifikáció

- Monitorok használata futásidőbeli hibadetektálásra
  - Formalizált követelmények ellenőrzése
    - Pl. reaktív hibakezelés támogatása
  - A működési, konfigurációs és környezeti hibák detektálása



Használati eset 2: Teszt kimenet értékelése

- Monitorok használata mint teszt oracle
  - Teszt követelmények teljesítésének kiértékelése
  - Tervezési és megvalósítási hibák detektálása



Kihívások

- Ellenőrzési módszerek
  - Követelmények formalizálása
  - Verifikációs algoritmusok kidolgozása
- Felműszerezés
  - Ellenőrzéshez szükséges információ megfigyelhetővé tétele
  - Erőforrásigény minimalizálása

- Elméleti eredmények gyakorlati aspektusai
  - Monitor szintézis
  - Kis erőforrásigényű, skálázható megvalósítás

  $\rightarrow$ Alkalmazás beágyazott biztonságkritikus rendszerekben

Kihívások

- Ellenőrzési módszerek

- Végrehajtási szekvencia (trace) alapján specifikált temporális tulajdonságok ellenőrzése
  - Temporális logikák
  - Referencia automaták
  - Reguláris kifejezések
- Szerződés (design-by-contract) alapú monitorozás
  - Kiértékelhető állítások (executable assertions)
- Általános (specification-less) elvárások monitorozása
  - Konkurens végrehajtás követelményei (pl. holtpont, versenyhelyzet, sorosíthatósági konfliktus detektálása)

Kihívások

- Ellenőrzési módszerek
  - Követelmények formalizálása
  - Ellenőrzési algoritmusok és ezek megvalósítása
- Felműszerezés

- Aktív és passzív felműszerezés
  - Aktív: kódrészletek beillesztése
  - Passzív: beavatkozás nélküli megfigyelés
- Aktív felműszerezés megvalósítási technológiák
  - Aspektus-orientált programozás (AOP)
  - Tracematch: AspectJ kiterjesztés esemény-mintákhoz
- Szinkron és aszinkron monitorozás

Példa: Keretrendszer monitor szintézishez

- MOP: Monitoring-Oriented Programming

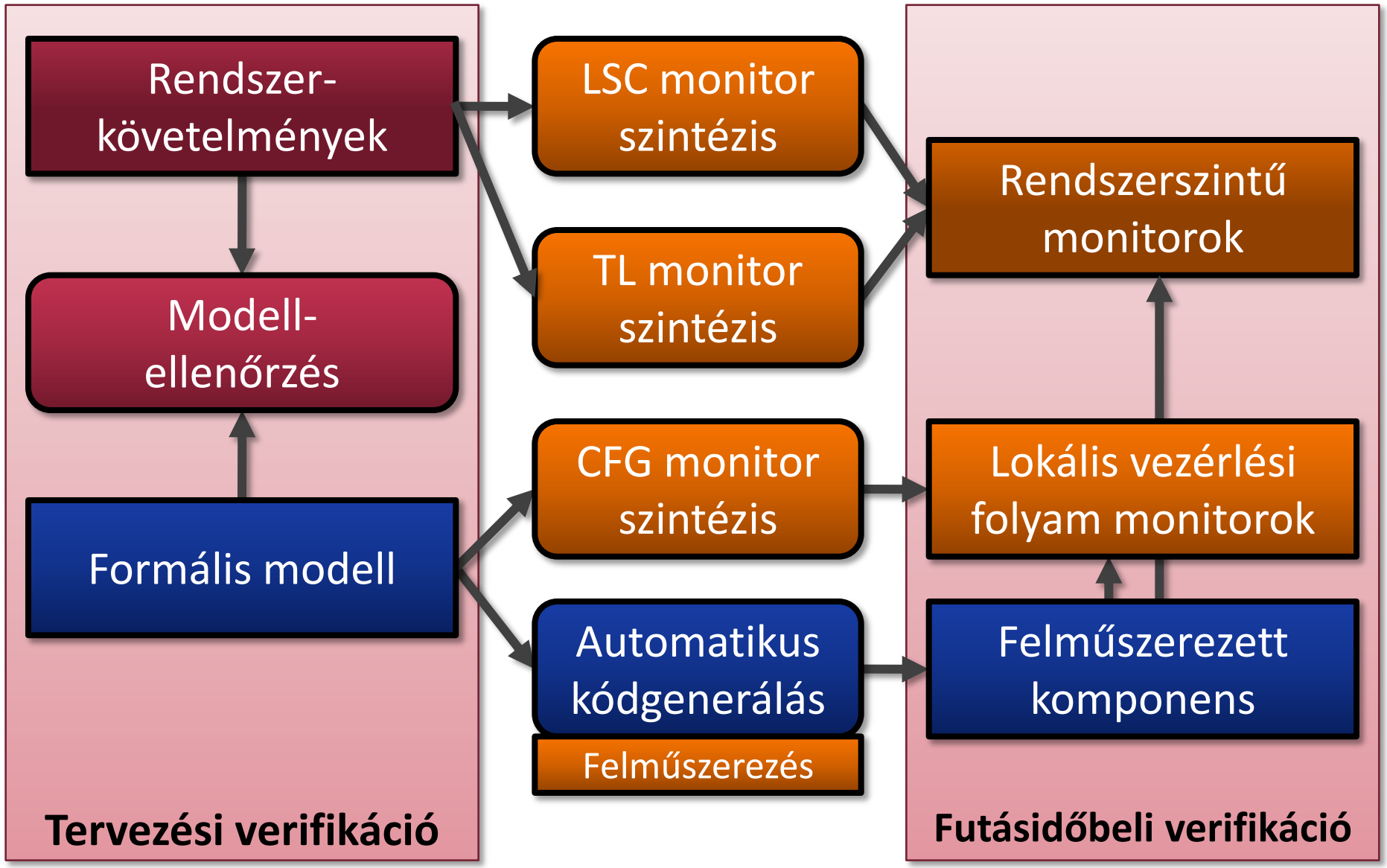
MOP / Languages \ Logic Plugins	FSM	ERE	CFG	PTLTL	LTL	PTCaRet	SRS	...
JavaMOP	JavaFSM	JavaERE	JavaCFG	JavaPTLTL	JavaLTL	JavaPTCaRet	JavaSRS	...
BusMOP	BusFSM	BusERE	...	BusPTLTL	...	...	...	...
ROSMOP	ROSFSM	...	ROSCFG	...	...	...	...	...
...	...	...	...	...	...	...	...	...

  - FSM -- Finite State Machines
  - ERE -- Extended Regular Expressions
  - CFG -- Context Free Grammars
  - PTLTL -- Past Time Linear Temporal Logic
  - LTL -- Linear Temporal Logic
  - PTCaRet -- Past Time LTL with Calls and Returns
  - SRS -- String Rewriting Systems

Az itt bemutatott megoldások

- Alkalmazás: Beágyazott rendszerekben
  - Állapot alapú, eseményvezérelt viselkedés
  - Biztonsági funkciókhoz (biztonsági protokollokhoz)
- Hierarchikus (skálázható) futásidőbeli verifikáció
  - Lokális: Egy-egy komponens (vezérlő, ECU) viselkedése
    - Referencia automata: vezérlési és egyszerű adathibák
    - Lokális temporális logikai (TL) követelmények
  - (Al)rendszerszintű: Komponensek interakciója
    - Rendszerszintű temporális követelmények
    - Scenario (LSC) alapú követelmények
- Kapcsolódás a modellvezérelt fejlesztéshez
  - Modell alapú kódgenerálás, felműszerezéssel

Hierarchikus monitorozás

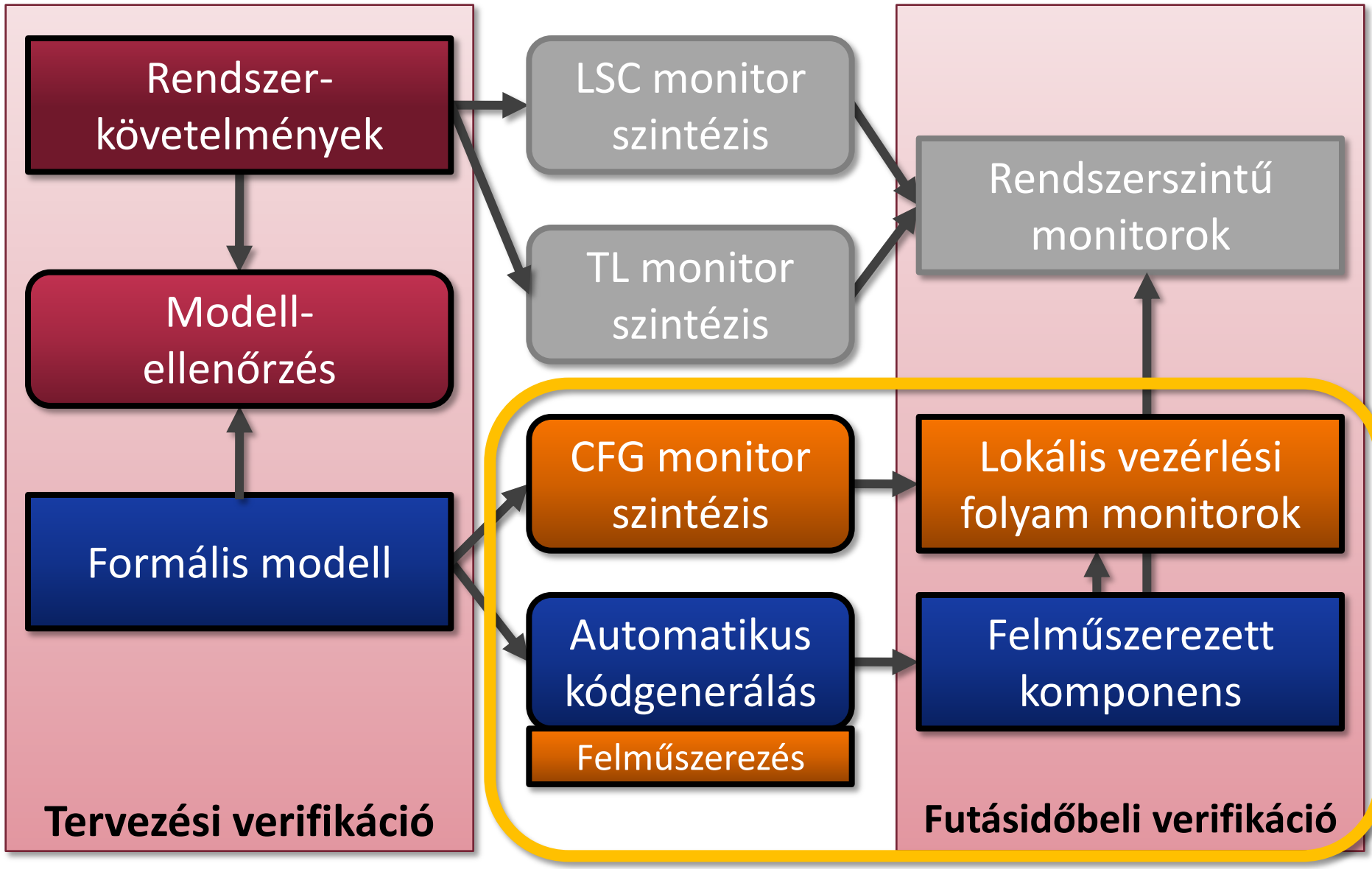


Futásidejű verifikáció referencia automaták alapján

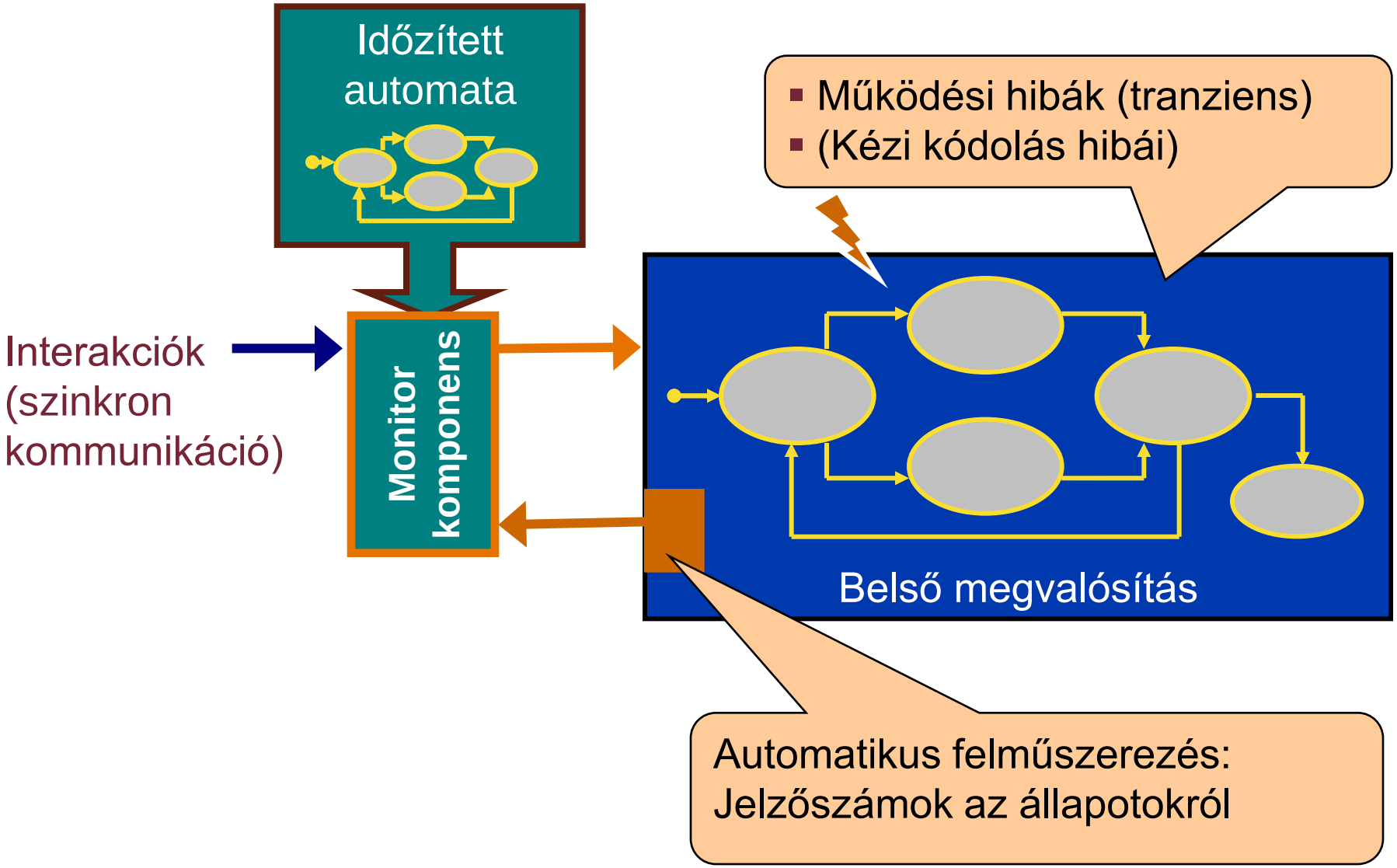
Időzített automata modell alapján

Állapottérkép modell alapján

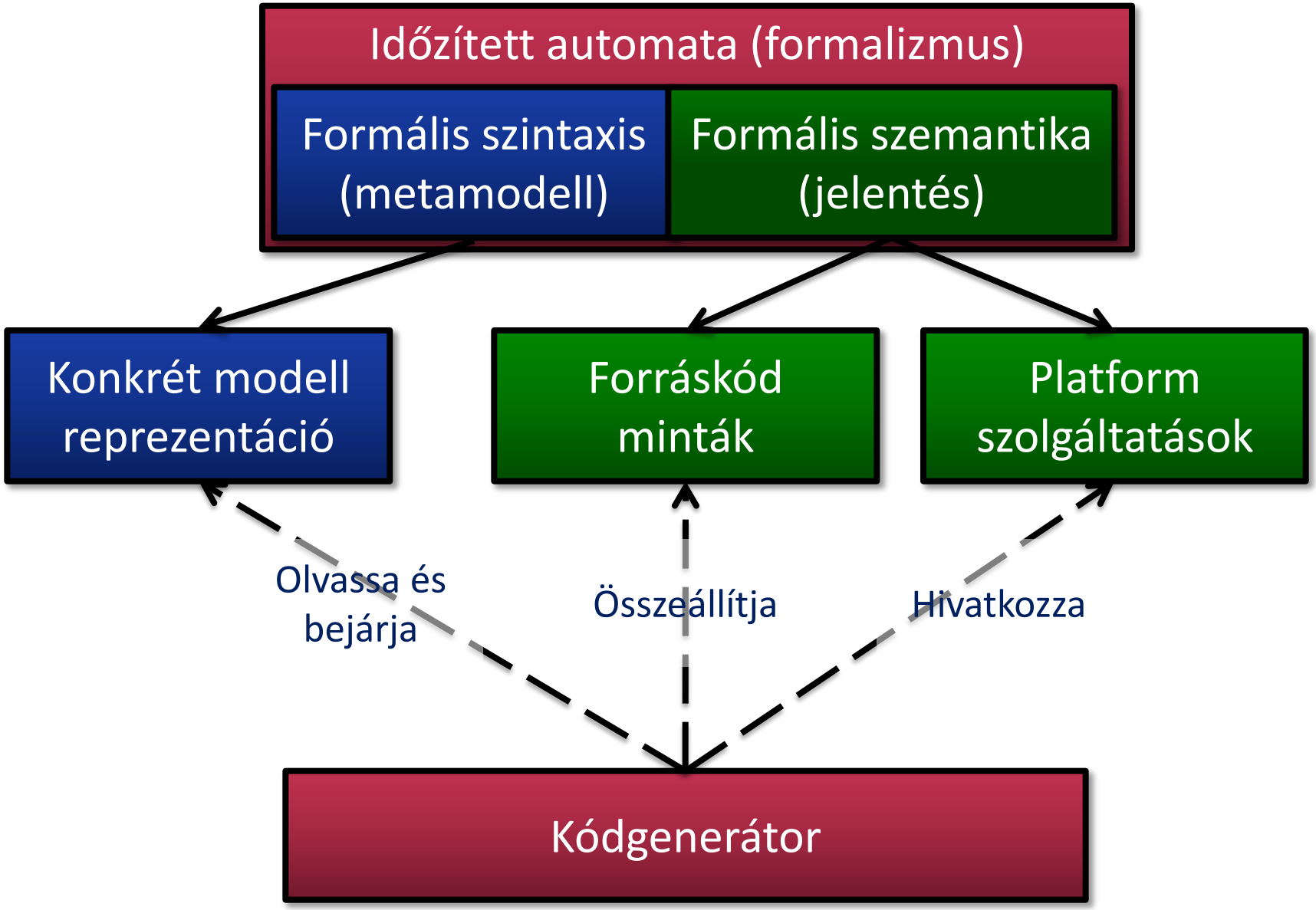
Hierarchikus monitorozás



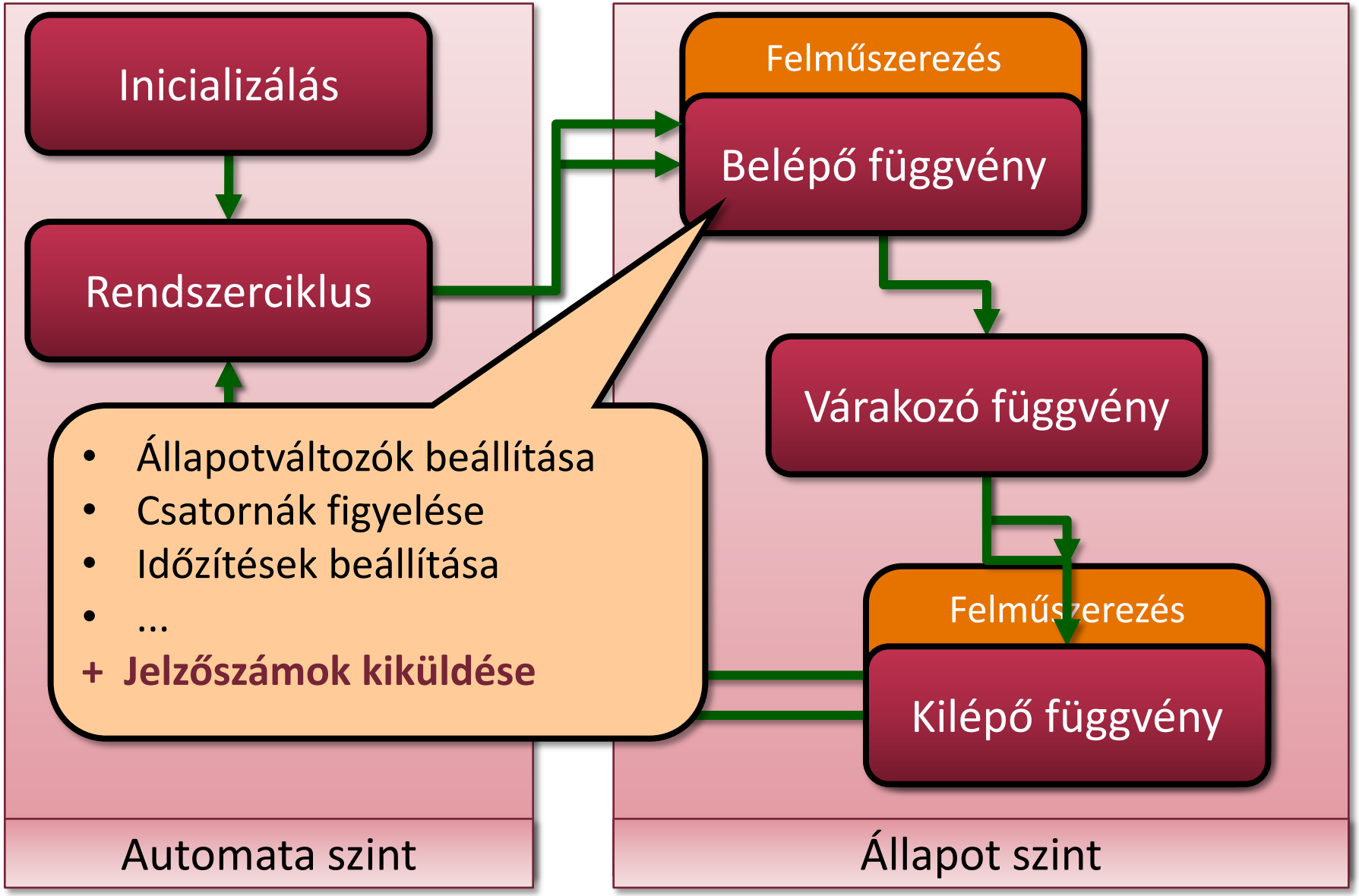
Monitorozás időzített automata alapján



Automatikus kódgenerálás



Felműszerezés a forráskódban



Jelzőszám alapú monitorozás

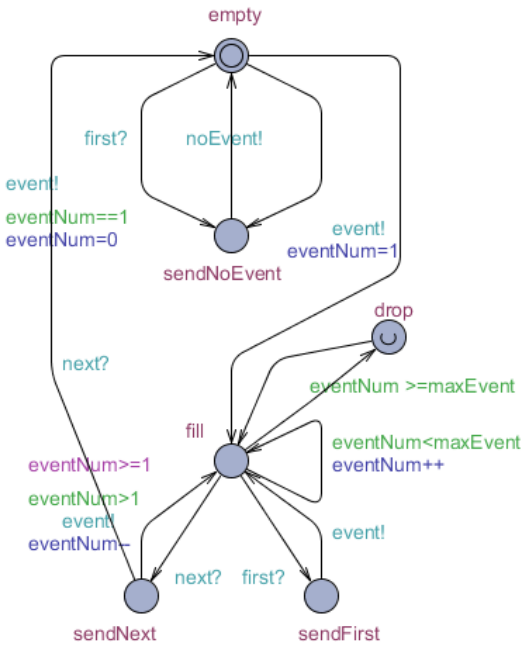
Komponens

Felműszerezés

Jelzőszámok: állapotok

Lokális vezérlési folyam monitor

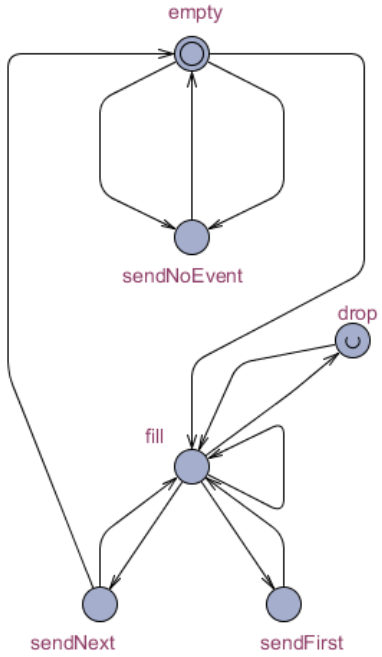
Referencia automata



Kódgenerálás alapja

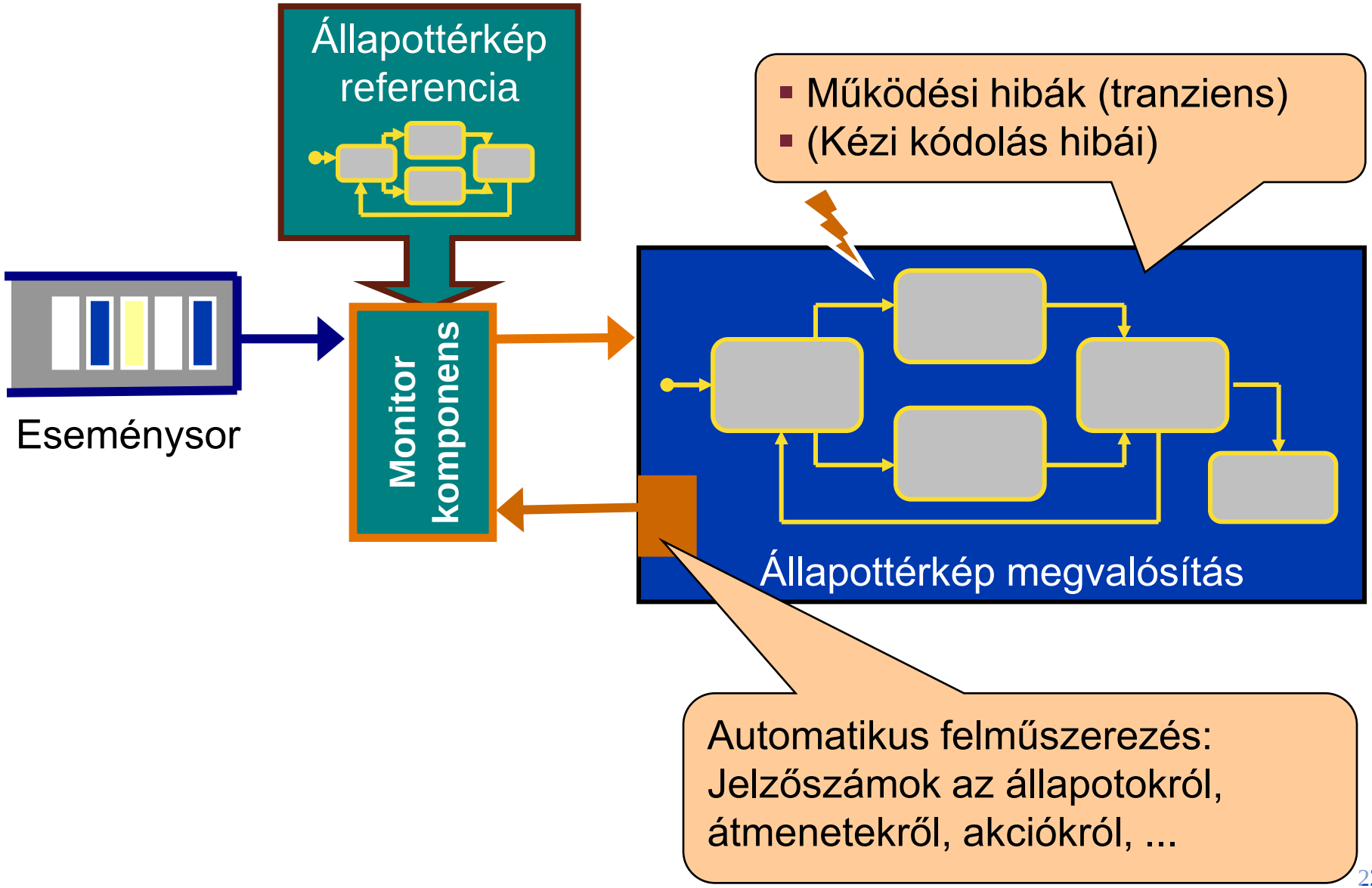
Detektálható hibák:

- Hibás állapot / átmenet szekvencia
- Leragadás (time-out)
- Időzítések megsértése



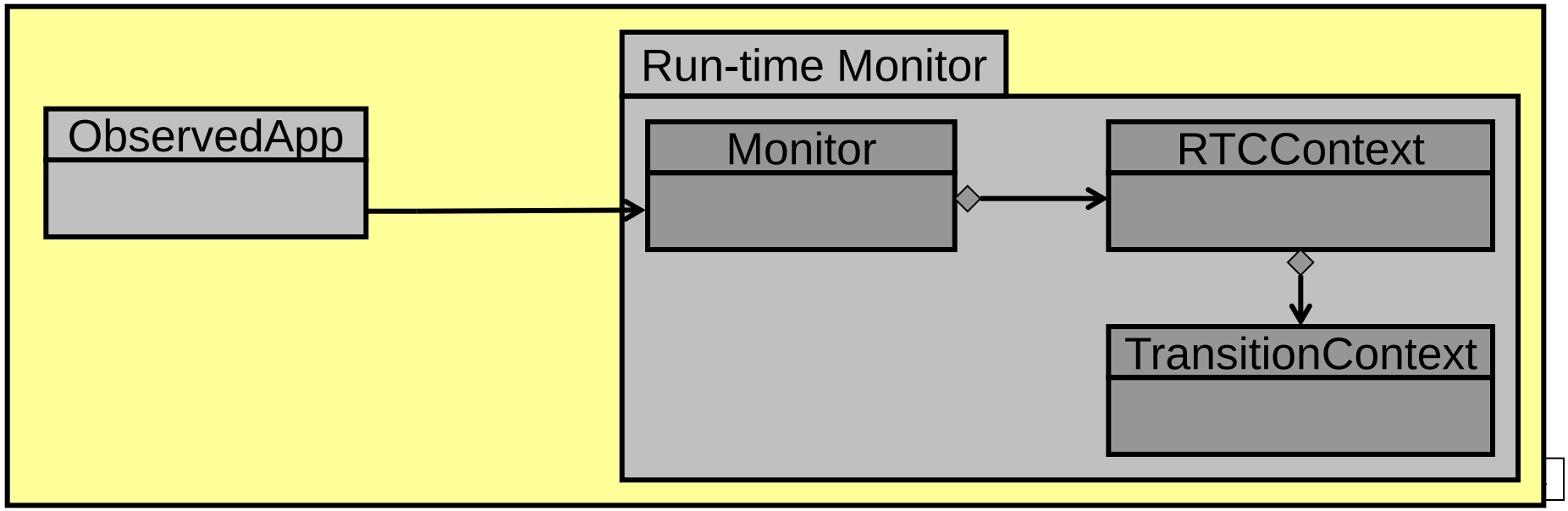
Referencia automata

Monitorozás állapottérkép alapján



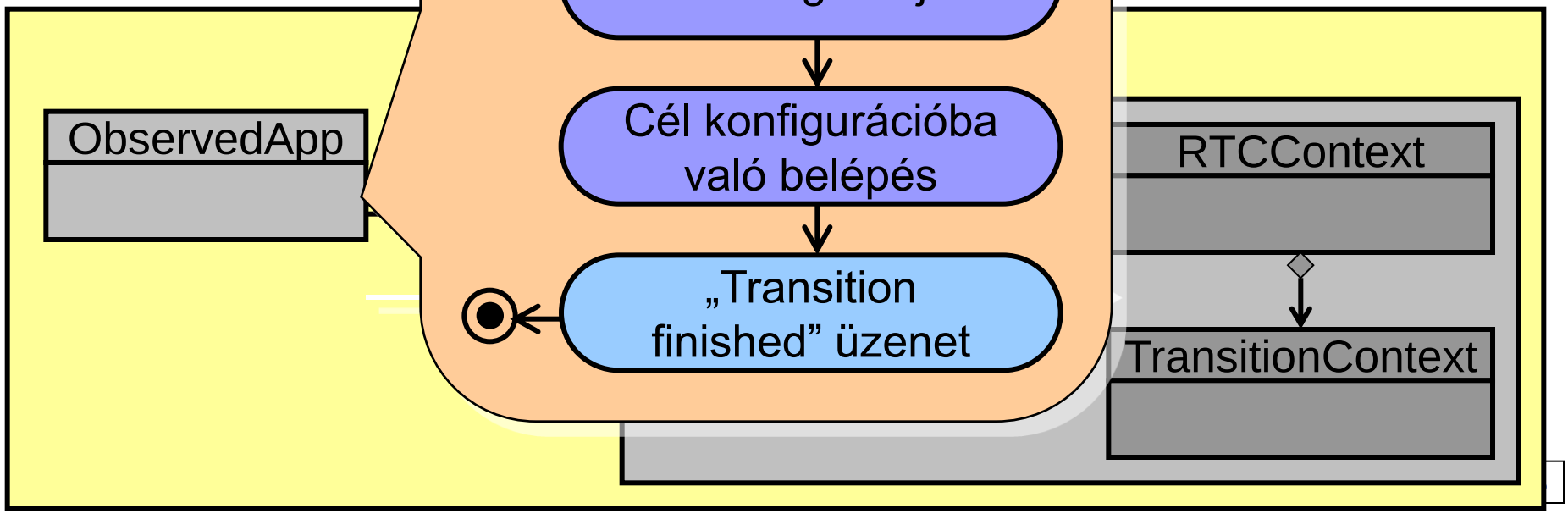
Ellenőrzés állapottérkép referencia alapján

- Szisztematikus transzparens felműszerezés:
  - Explicit jelzések a monitor számára
    - Elhagyott állapotok, belépett állapotok
    - Végrehajtott akciók
  - Megvalósítási példa: Aspektus-orientált programozás



Ellenőrzés áll rencia alapján

- Szisztematik erezés:
  - Explicit jelz
    - Elhagyott
    - Végrehajt
  - Megvalósít t programozás



Ellenőrzés állapottérkép referencia alapján

- Szisztemati
  - Explicit je
    - Elhagyo
    - Végreha
  - Megvalós

Extra kód átmenet tüzeléséhez (Java AOP)

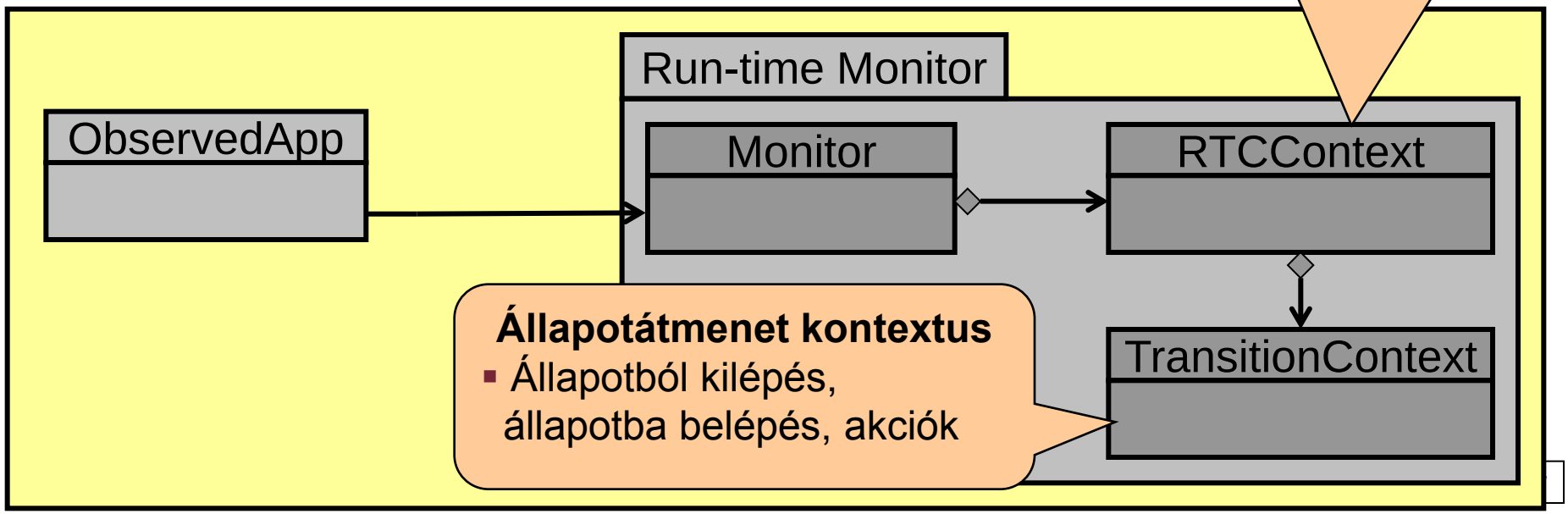
```
public aspect BehavioralMonitoring {
   // Define pattern matching
   pointcut firingTransitionPattern
      call (StatechartBase+.fireTransition(Transition t));

   // Define instrumentation to be applied
   around(): firingTransitionPattern() {
      msgq.sendTrStarting(...);
      proceed();
      msgq.sendTrFinishing(...);
   }
}
```

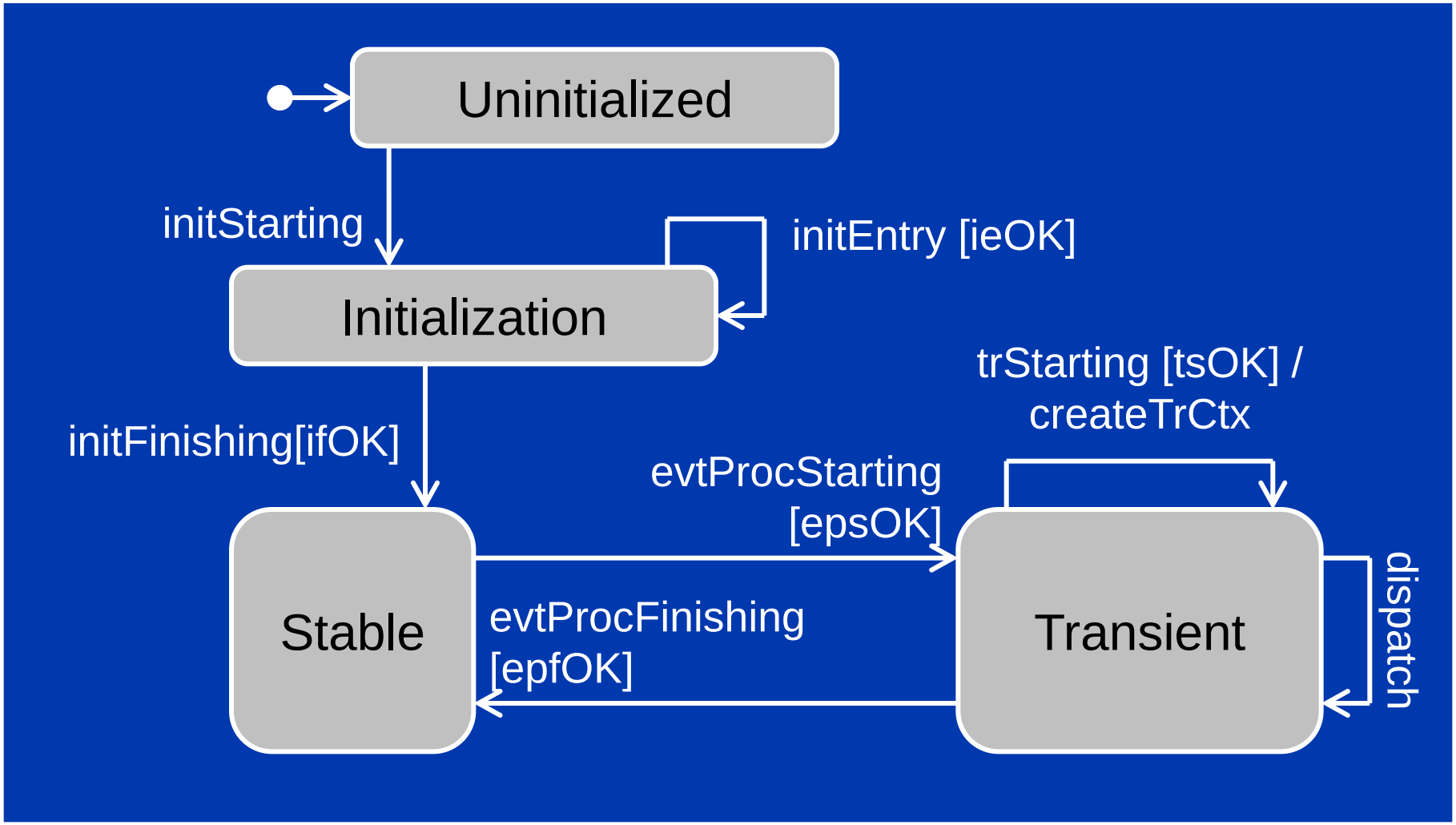
Ellenőrzés állapottérkép referencia alapján

- Szisztematikus transzparens felműszerezés:
  - Explicit jelzések a monitor szám
    - Elhagyott állapotok, belépett álla
    - Végrehajtott akciók
  - Megvalósítási példa: Aspektus-

RTC kontextus
- Inicializálás
- Esemény-feldolgozás kezdete és vége (állapotok)
- Átmenet kontextus részére üzenetek



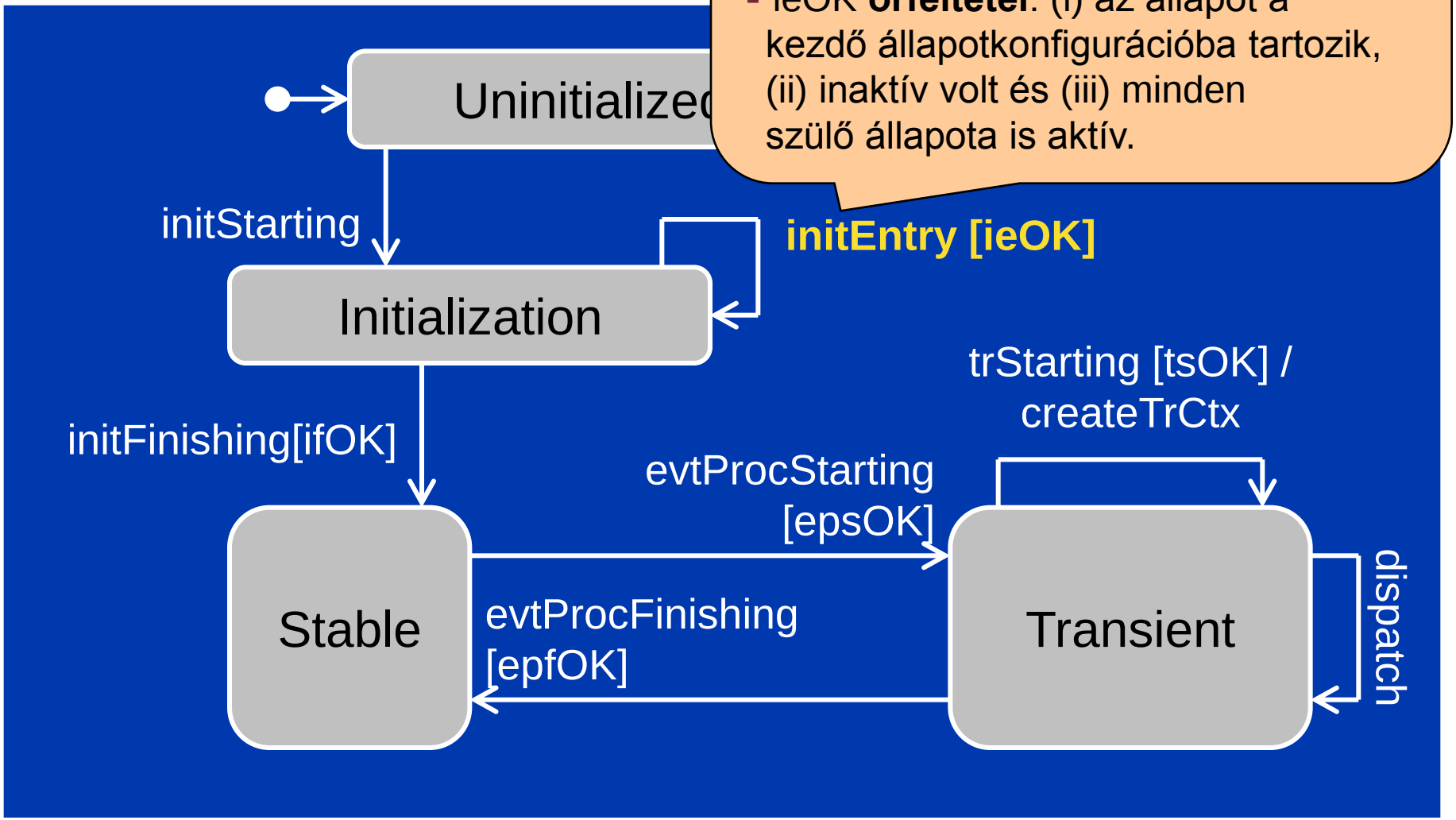
Run-to-completion context (RTCContext)



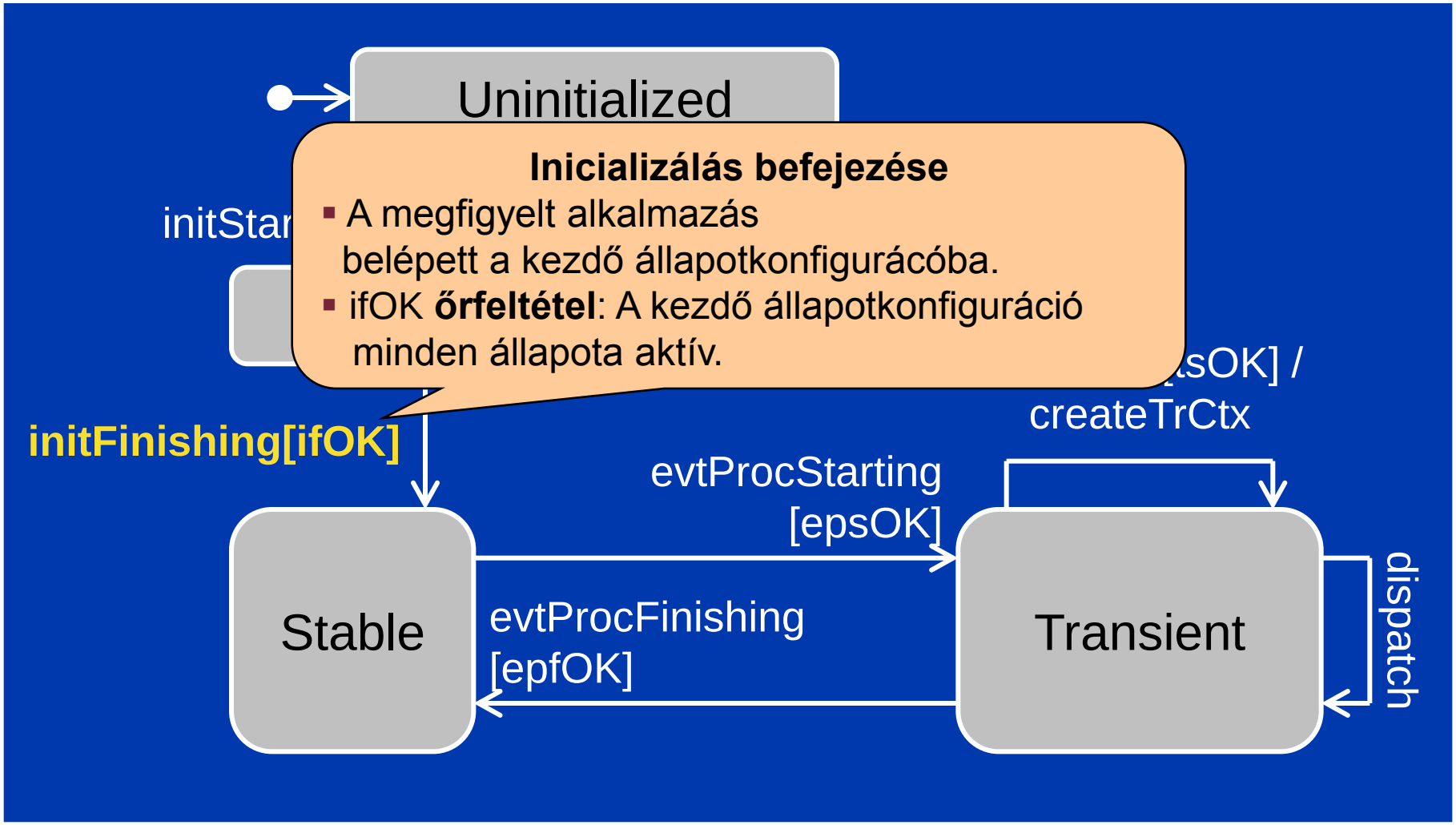
Run-to-completion context (RTCContext)

Inicializálás közben

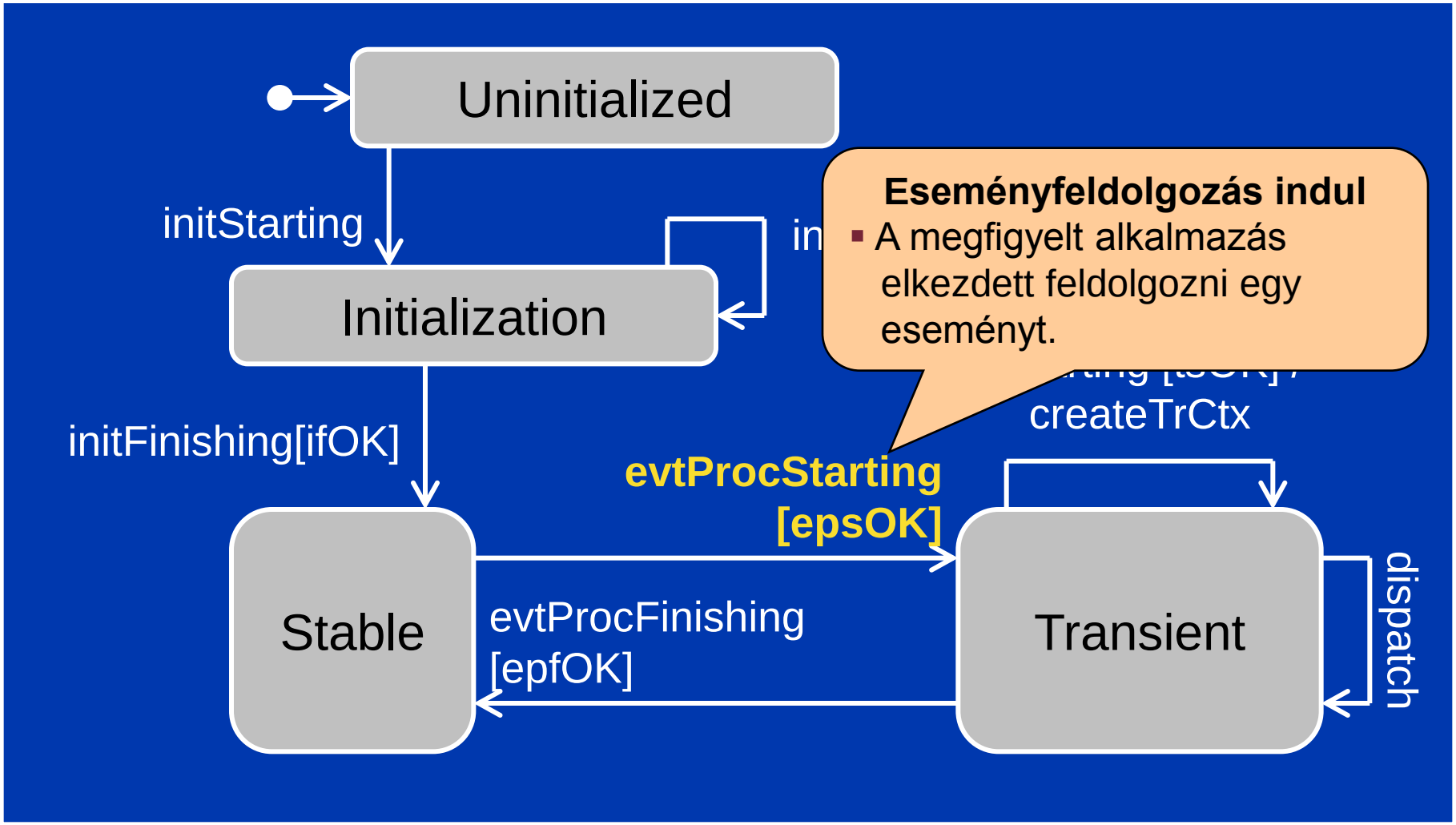
- initEntry: A megfigyelt alkalmazás belépett egy kezdőállapotba
- ieOK **őrfeltétel**: (i) az állapot a kezdő állapotkonfigurációba tartozik, (ii) inaktív volt és (iii) minden szülő állapota is aktív.



Run-to-completion context (RTCContext)



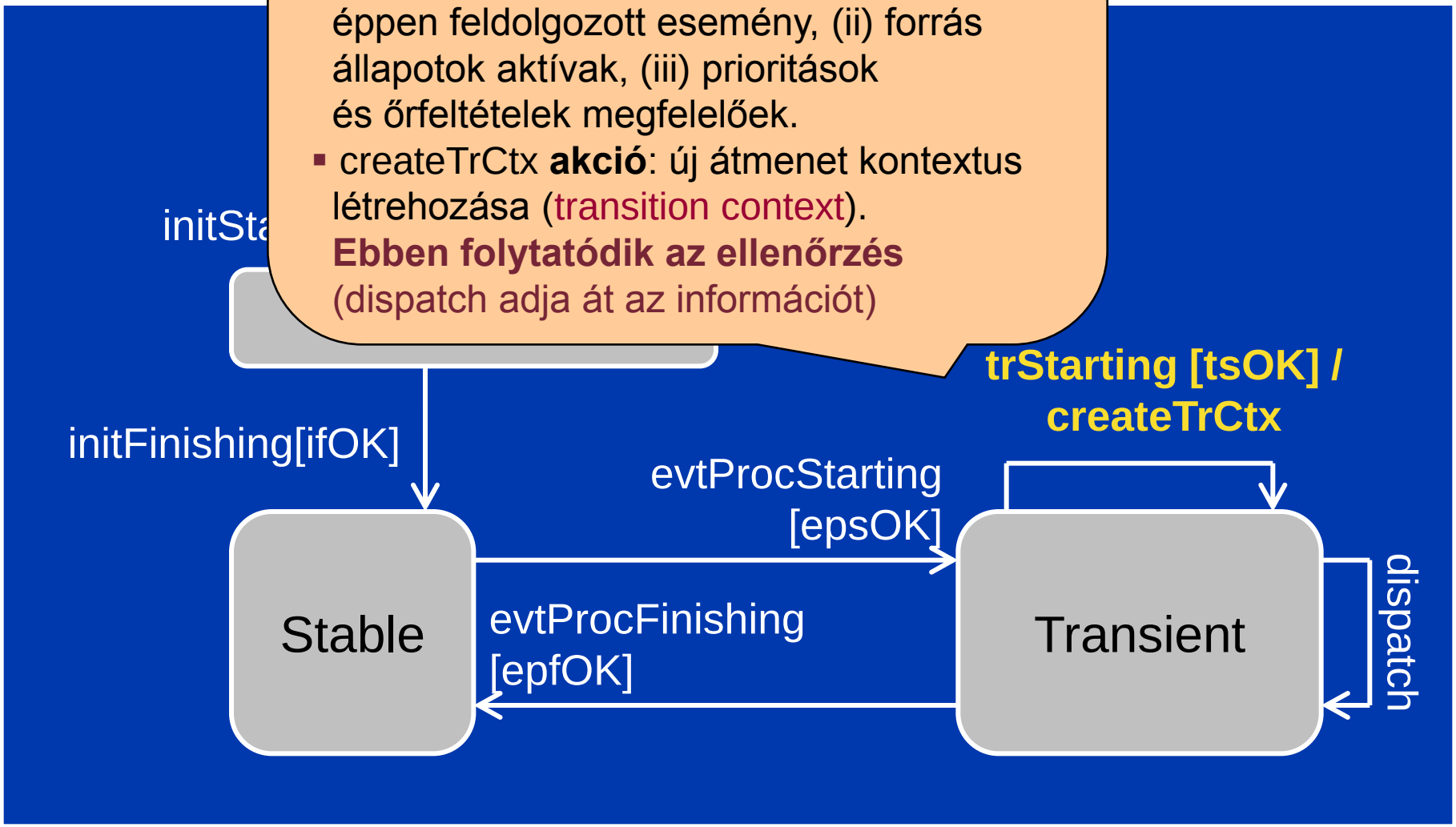
Run-to-completion context (RTCContext)



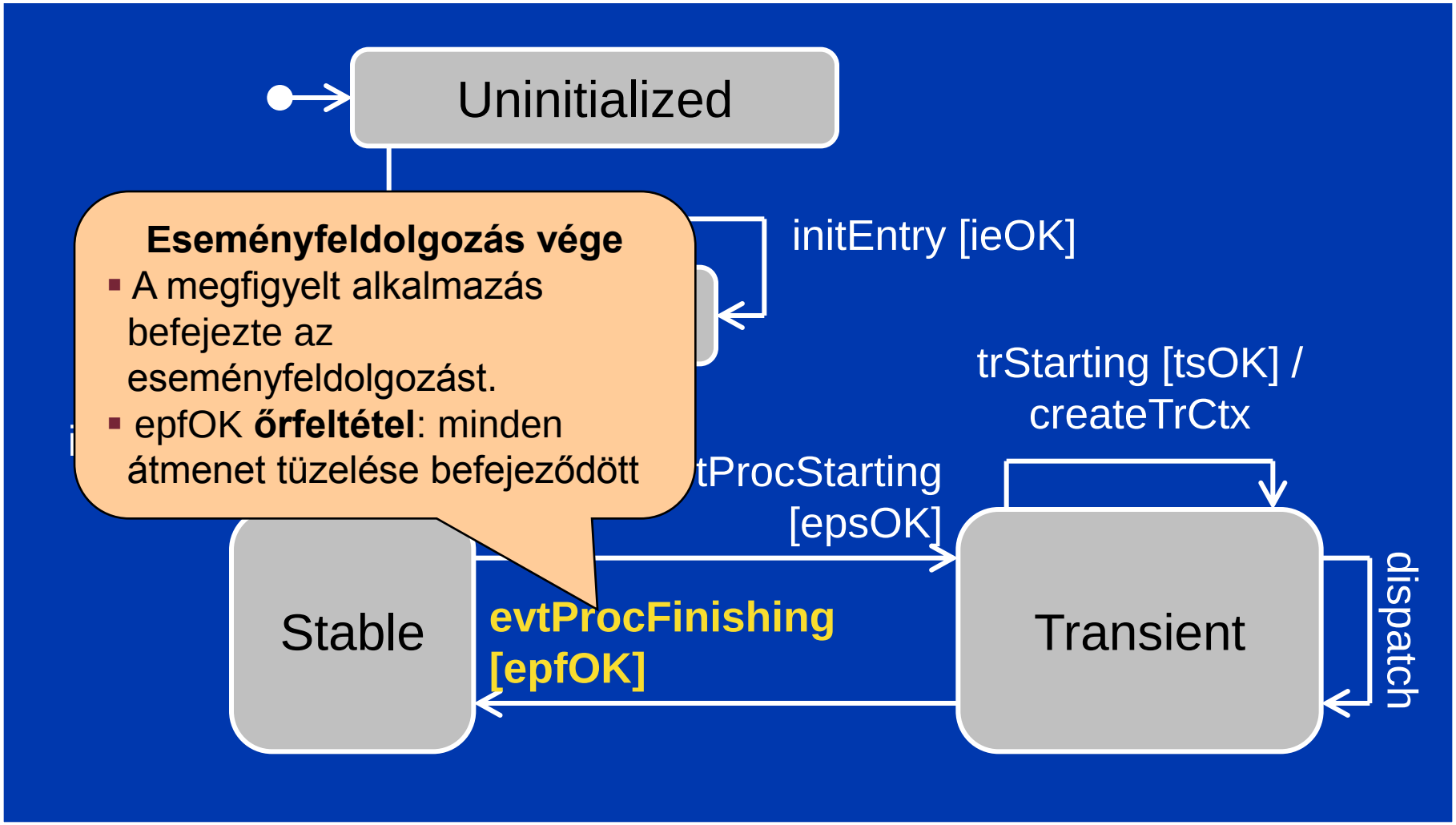
Run-to-c… …TCContext)

Átmenet tüzelése

- A megfigyelt alkalmazás elkezdte egy átmenet tüzelését.
- tsOK **őrfeltétel**: (i) a trigger esemény az éppen feldolgozott esemény, (ii) forrás állapotok aktívak, (iii) prioritások és őrfeltételek megfelelőek.
- createTrCtx **akció**: új átmenet kontextus létrehozása (transition context). **Ebben folytatódik az ellenőrzés** (dispatch adja át az információt)

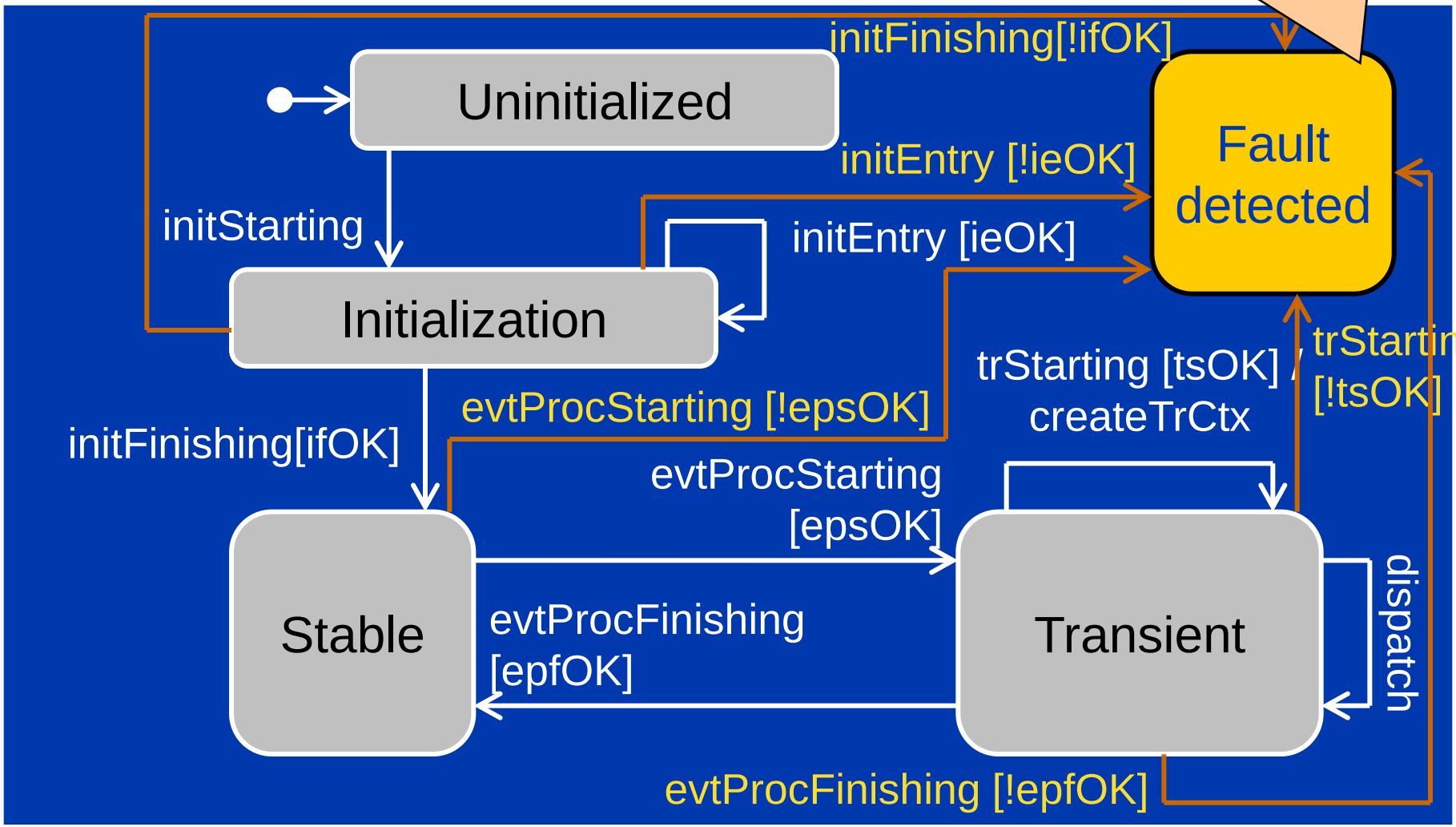


Run-to-completion context (RTCContext)

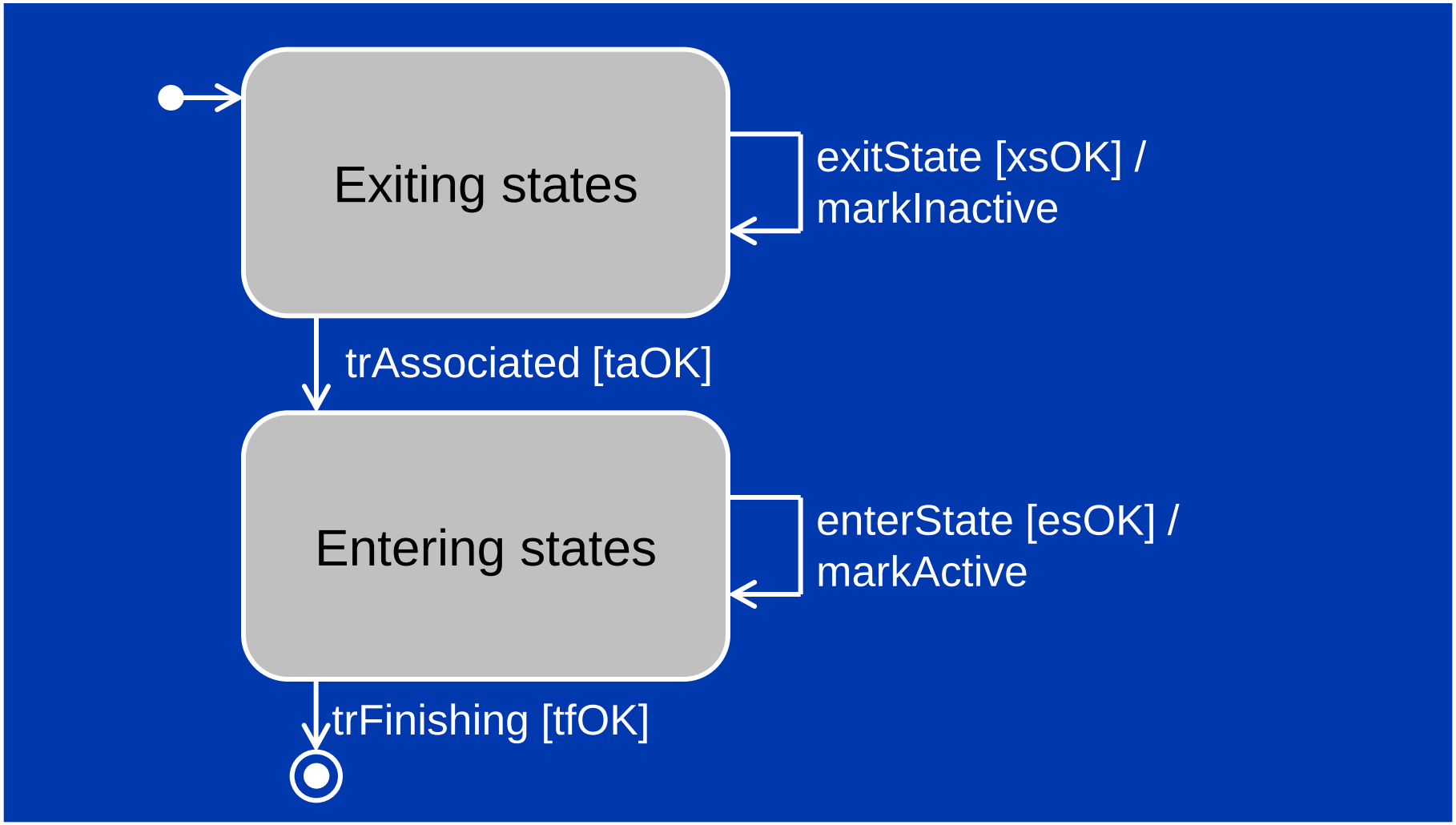


Hibadetektálás az RTCContext esetén

Hibadetektálás
- Őrfeltétel hamis, vagy
- nem várt üzenet jön



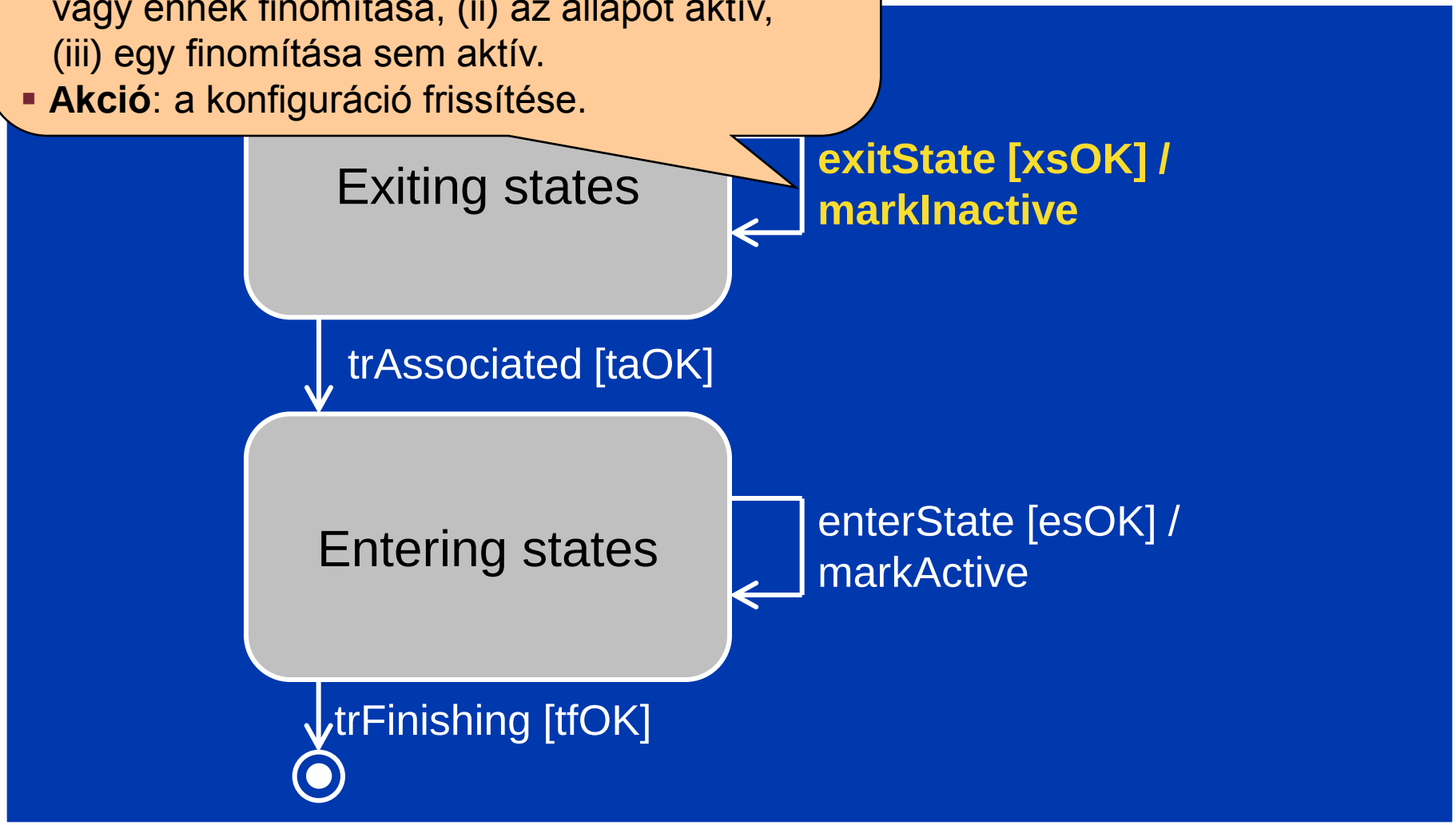
Állapotátmenet kontextus (TransitionContext)



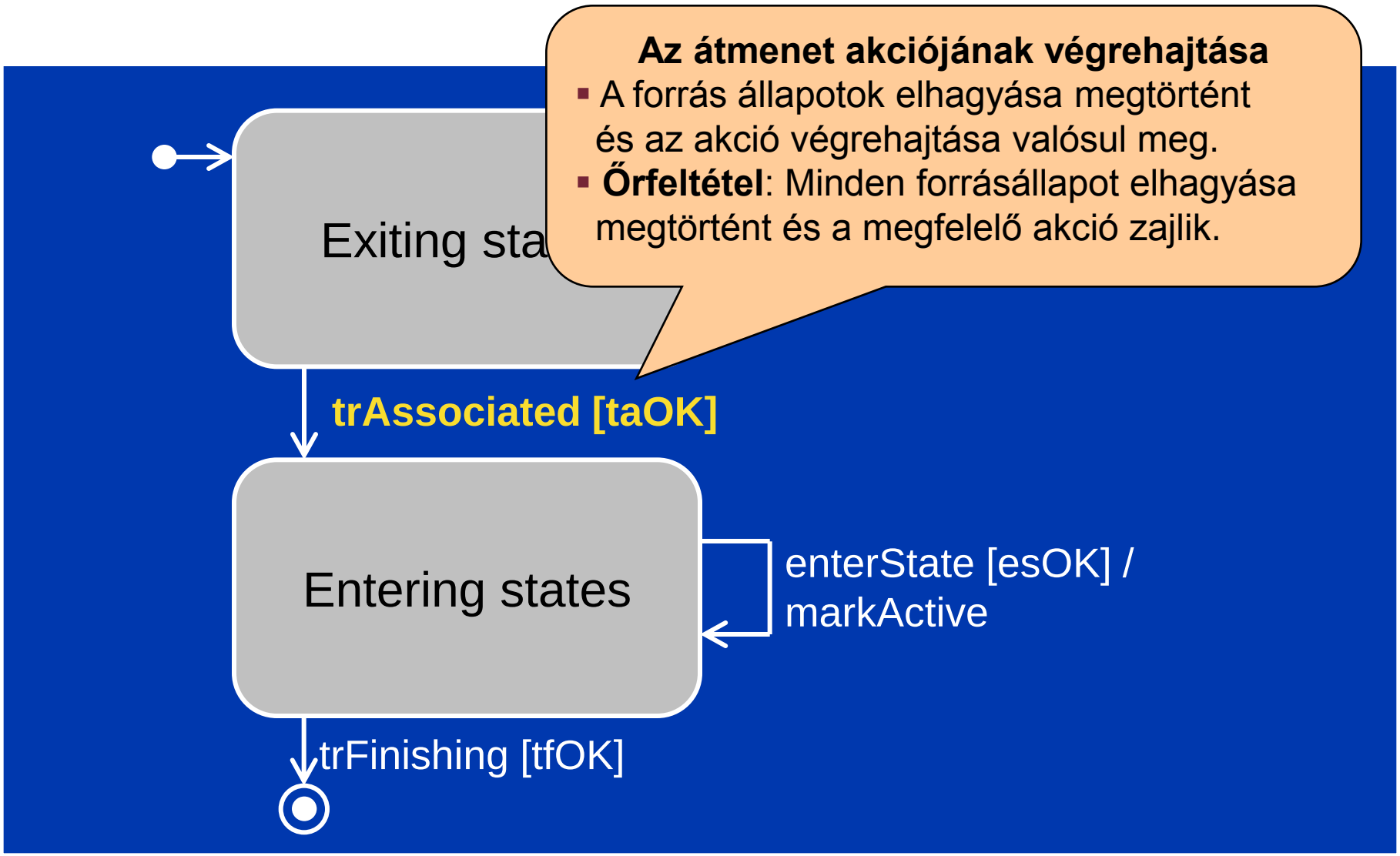
(TransitionContext)

Állapot elhagyása

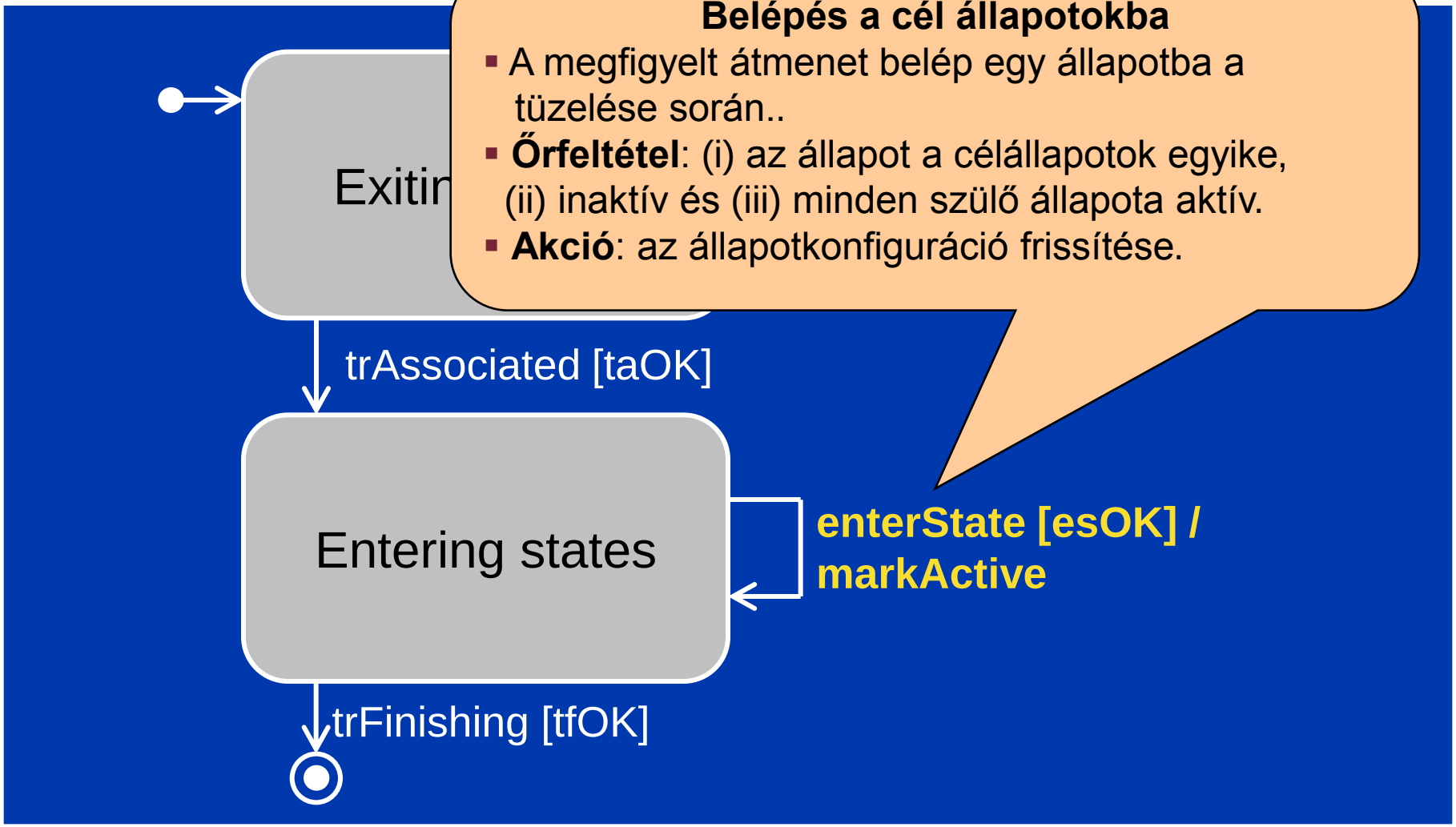
- A megfigyelt alkalmazás elhagyott egy állapotot átmenet tüzelés közben.
- **Őrfeltétel**: (i) az állapot az átmenet forrása vagy ennek finomítása, (ii) az állapot aktív, (iii) egy finomítása sem aktív.
- **Akció**: a konfiguráció frissítése.



Állapotátmenet kontextus (TransitionContext)



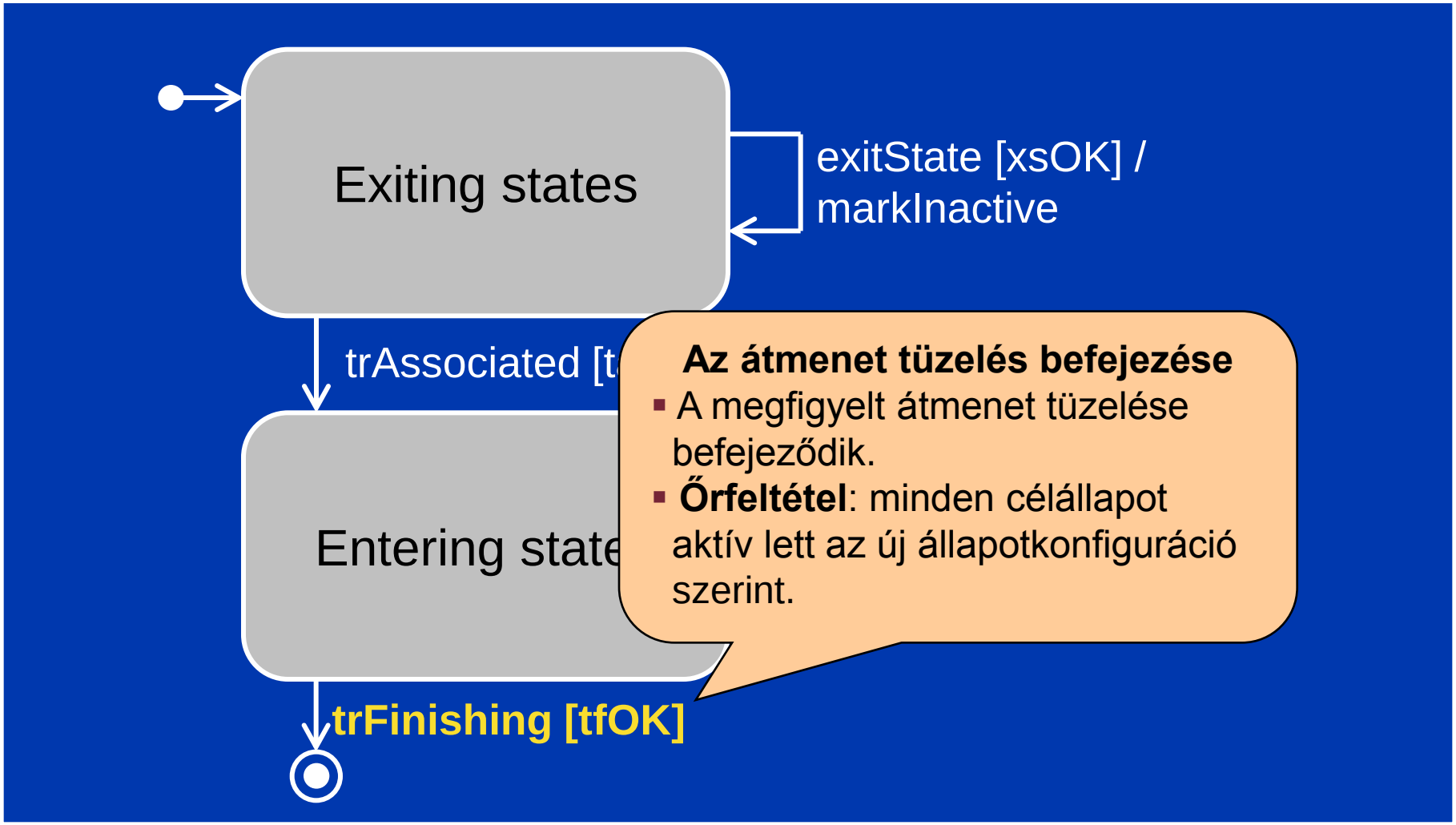
Állapotátmenet kontextus (TransitionContext)



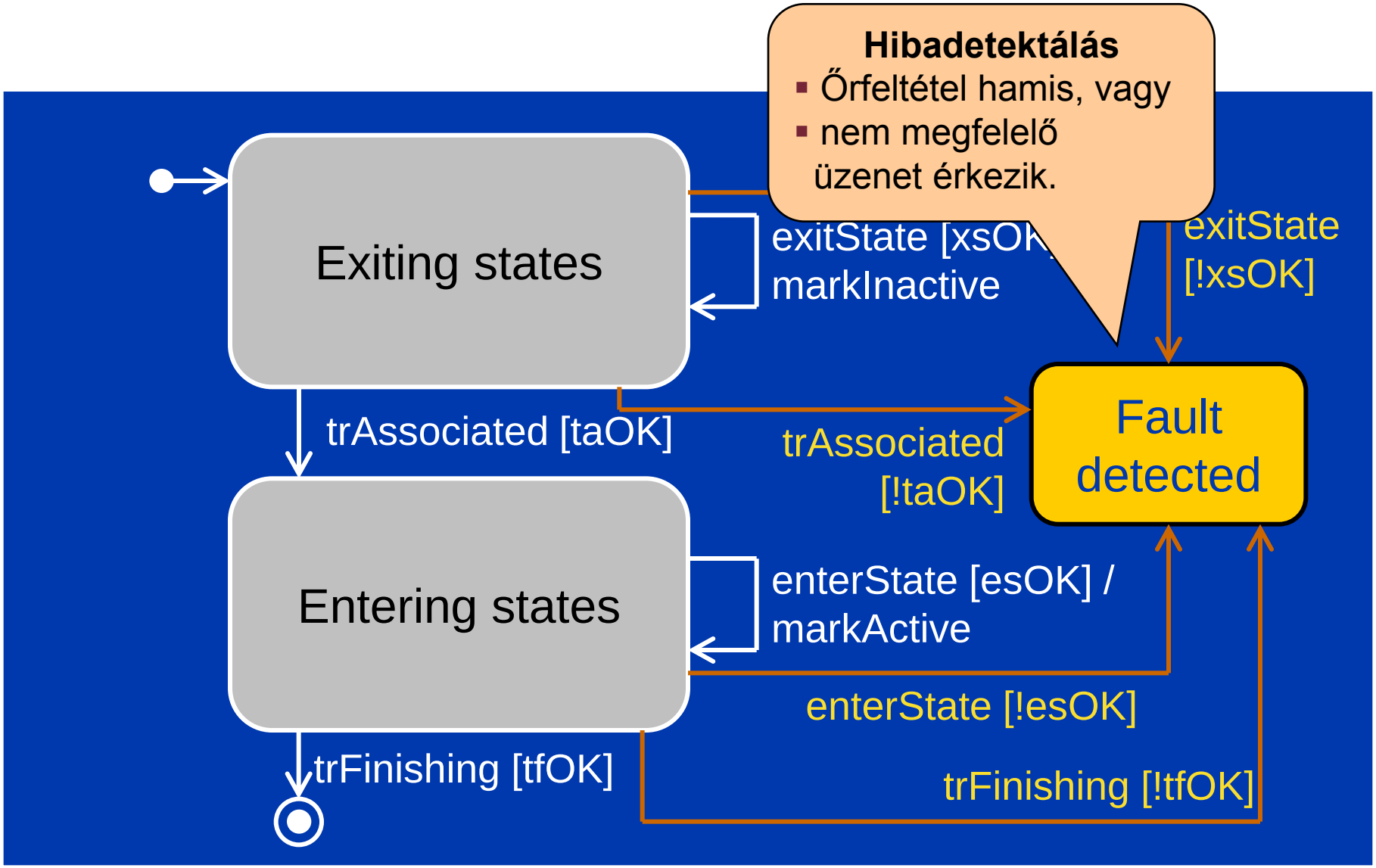
Belépés a cél állapotokba

- A megfigyelt átmenet belép egy állapotba a tüzelése során..
- **Őrfeltétel**: (i) az állapot a célállapotok egyike, (ii) inaktív és (iii) minden szülő állapota aktív.
- **Akció**: az állapotkonfiguráció frissítése.

Állapotátmenet kontextus (TransitionContext)



Hibadetektálás a TransitionContext esetén

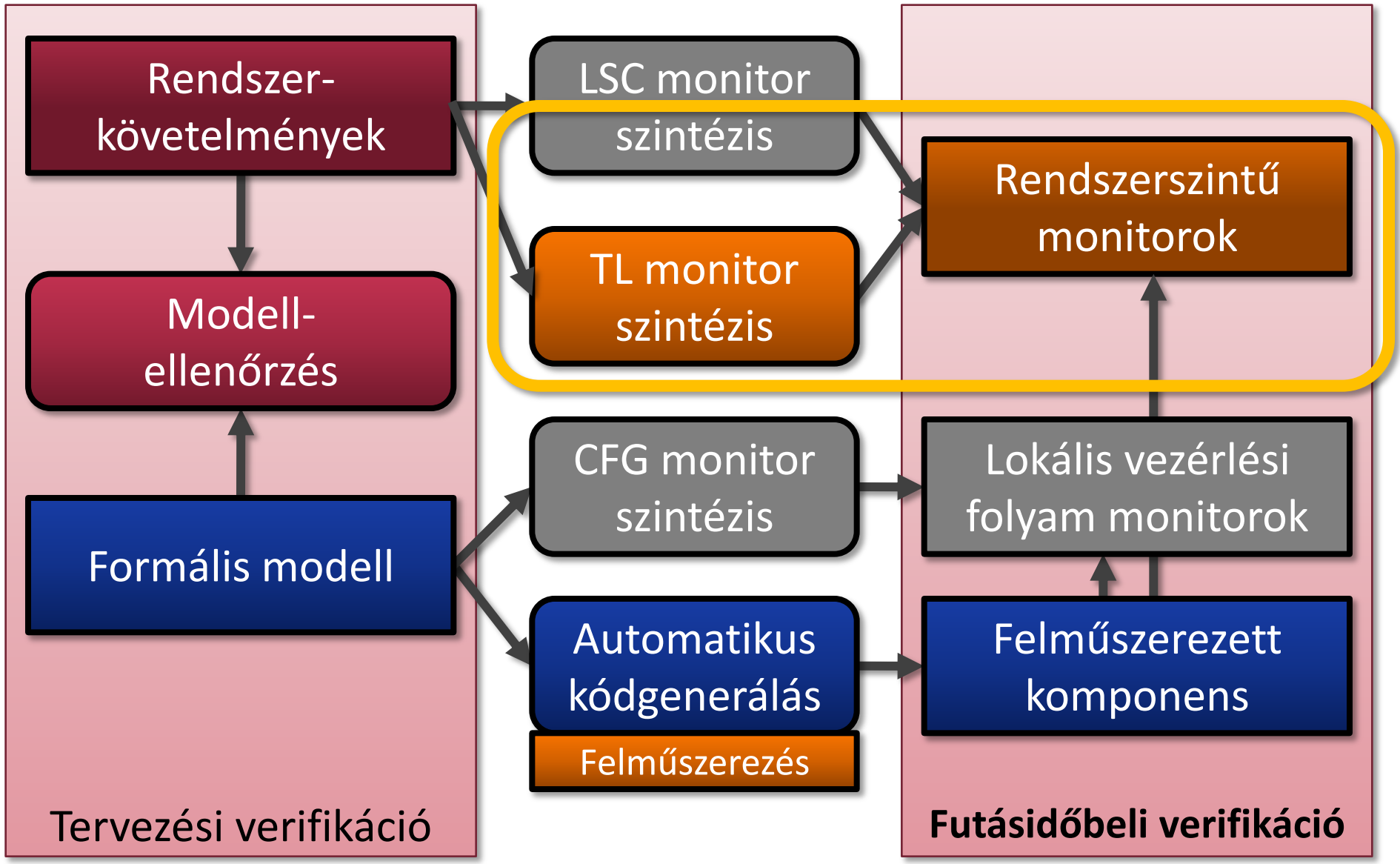


Futásidejű verifikáció temporális követelmények alapján

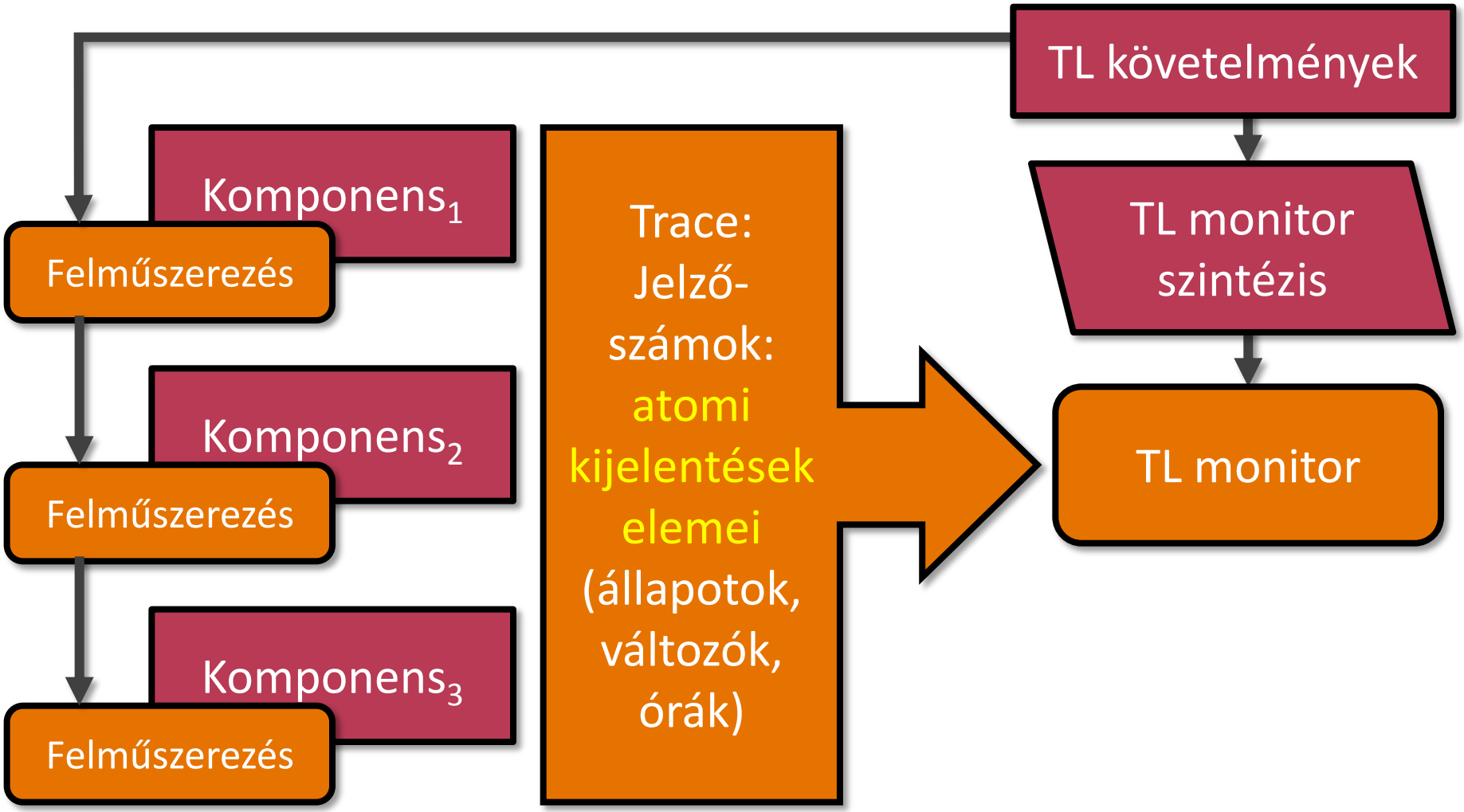
Általános lineáris temporális logikai követelmények

UPPAAL CTL követelmények (teszteléshez is)

Hierarchikus monitorozás



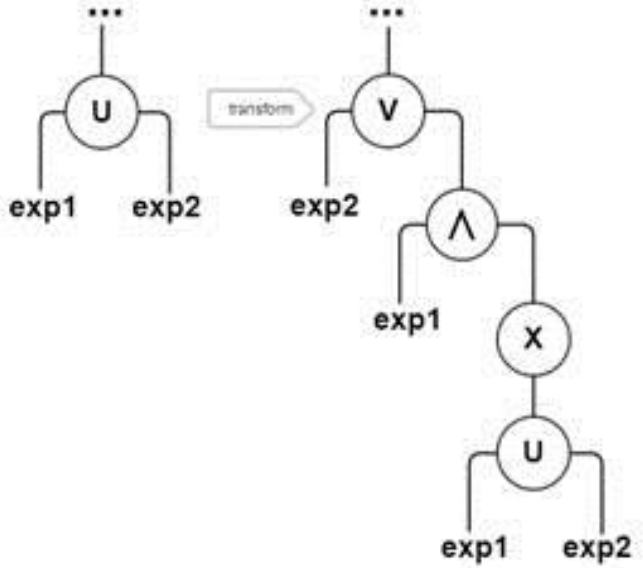
TL követelmények ellenőrzése



Monitorozás LTL kifejezések alapján

- Egy LTL kifejezés ellenőrzése a trace egy lépése esetén
  - Függőség az aktuális lépéstől: Boole logikai operátorok a lokális atomi kijelentéseken + lokális (“jelen idejű”) része a temporális operátoroknak
  - Függőség jövőbeli lépésektől: “jövő idejű” része a temporális operátoroknak
- A temporális kifejezés újraírása: Részkifejezések elválasztása
  - Az aktuális lépésben (Boole operátorokkal) kiértékelhető részkifejezések
  - A következő lépésben (egy X operátor után) kiértékelhető részkifejezések

exp1 U exp2 =
exp2 ∨ (exp1 ∧
X(exp1 U exp2))



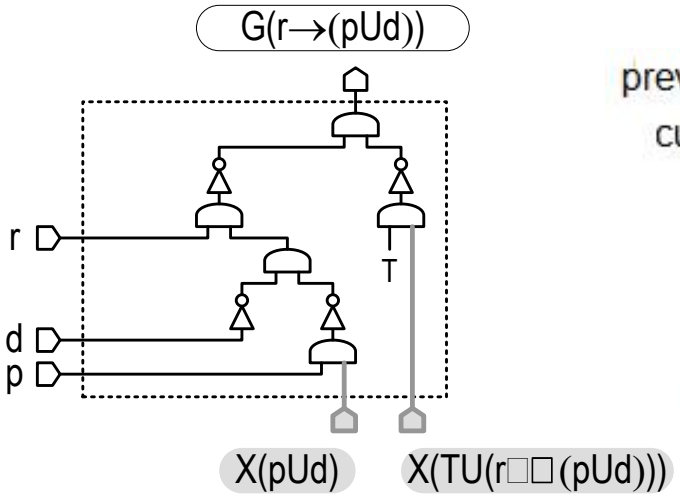
Következő állapotra vonatkozó kifejezések

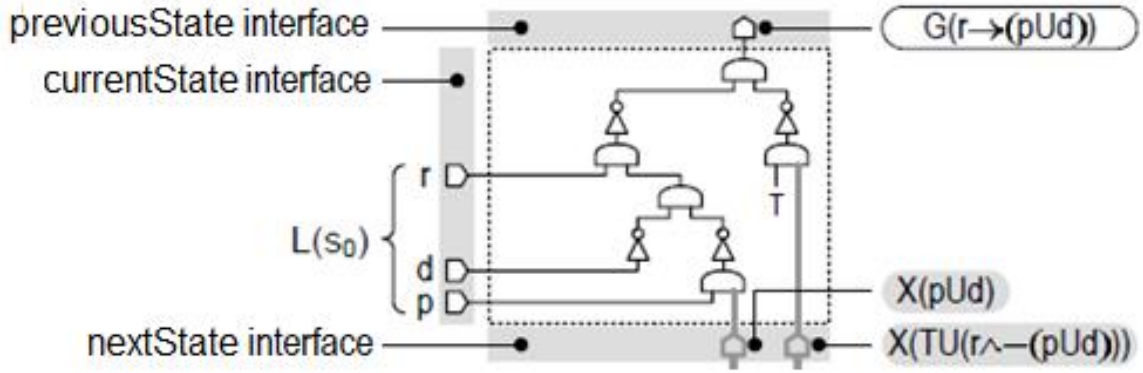
Kifejezések újraírása

- Definíció: Csatlakoztatási normálforma
  - A kifejezésben szereplő összes *UNTIL* (U) temporális operátor valamely *NEXT* (X) temporális operátor hatókörében van, és
  - a kifejezés minden olyan része, amely nem esik *NEXT* temporális operátor hatókörébe, kizárólag *AND*, valamint *NOT* operátorokat tartalmaz
- Átalakítás csatlakoztatási normálformára
  - Boole operátorokra: De Morgan azonosságok
  - Temporális operátorokra:
    - Visszavezetés X és U operátorokra
    - Átalakítás: $P \, U \, Q = Q \vee (P \wedge X(P \, U \, Q)) = \neg(\neg Q \wedge \neg(P \wedge X(P \, U \, Q)))$

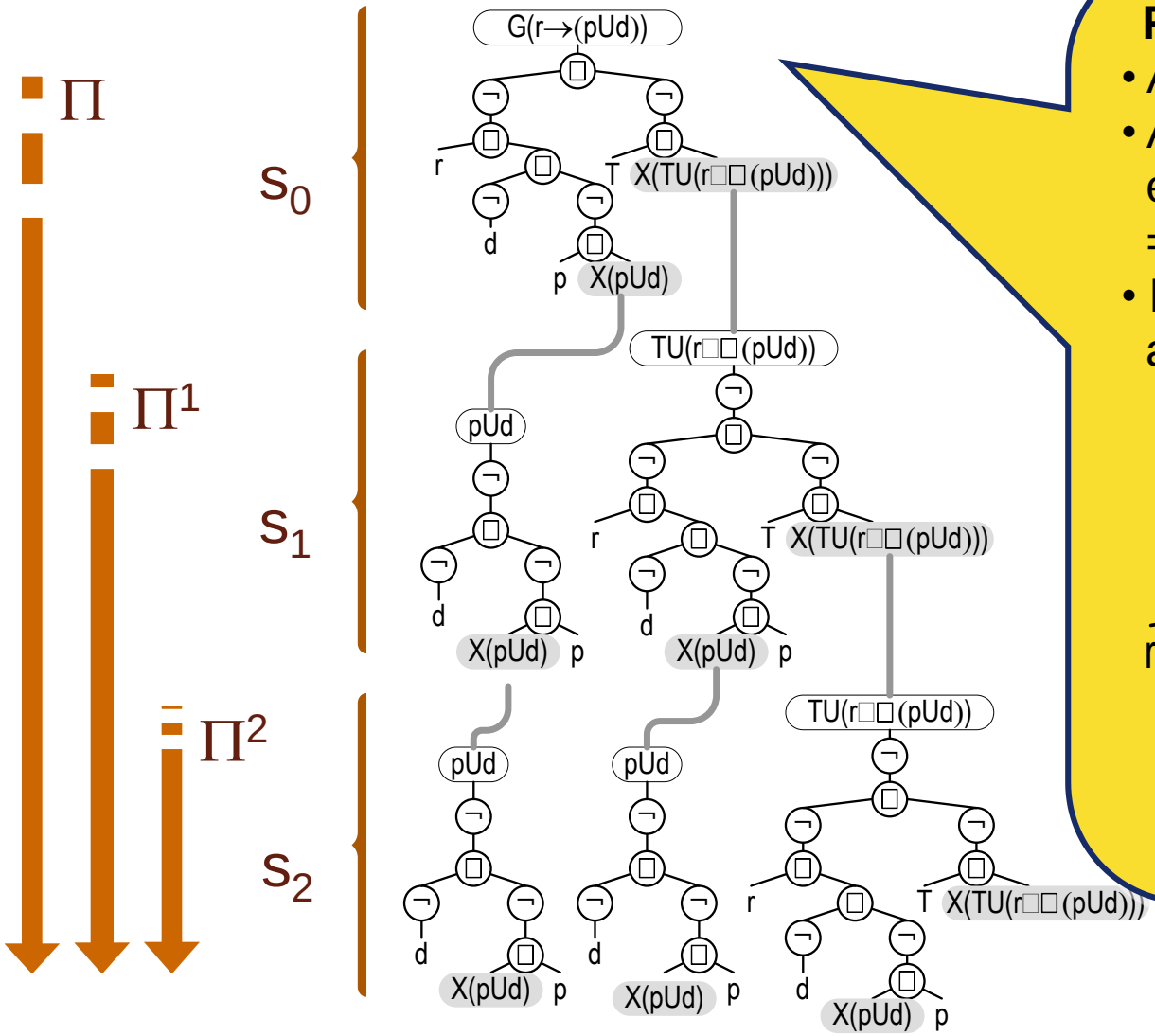
Kiértékelő blokkok

- Belső logika: Az aktuális lépés kiértékelése (Boole logika)
- Kimenet: A trace jelen lépésében a kifejezés kiértékelése
- Bemenetek:
  - Lokális atomi kijelentések
  - Következő lépésben történő kiértékelés eredménye
    ← Ez a trace következő lépésében, az X operátor utáni kifejezéseket kiértékelő blokkok kimenete lesz (így blokkok láncolata képezhető)



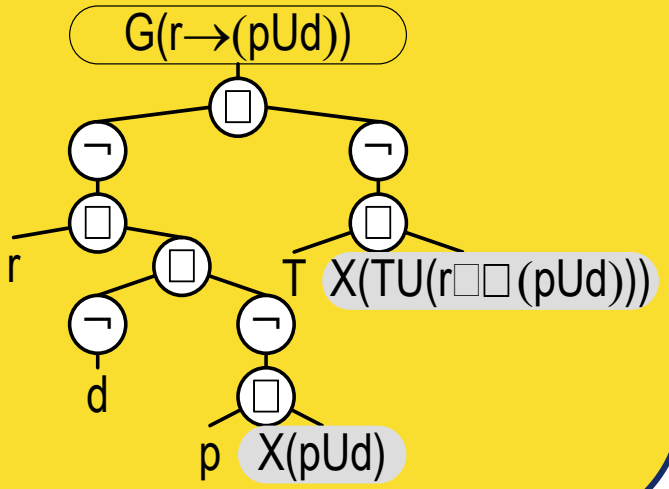


Kifejezések kiértékelése a trace lépésein



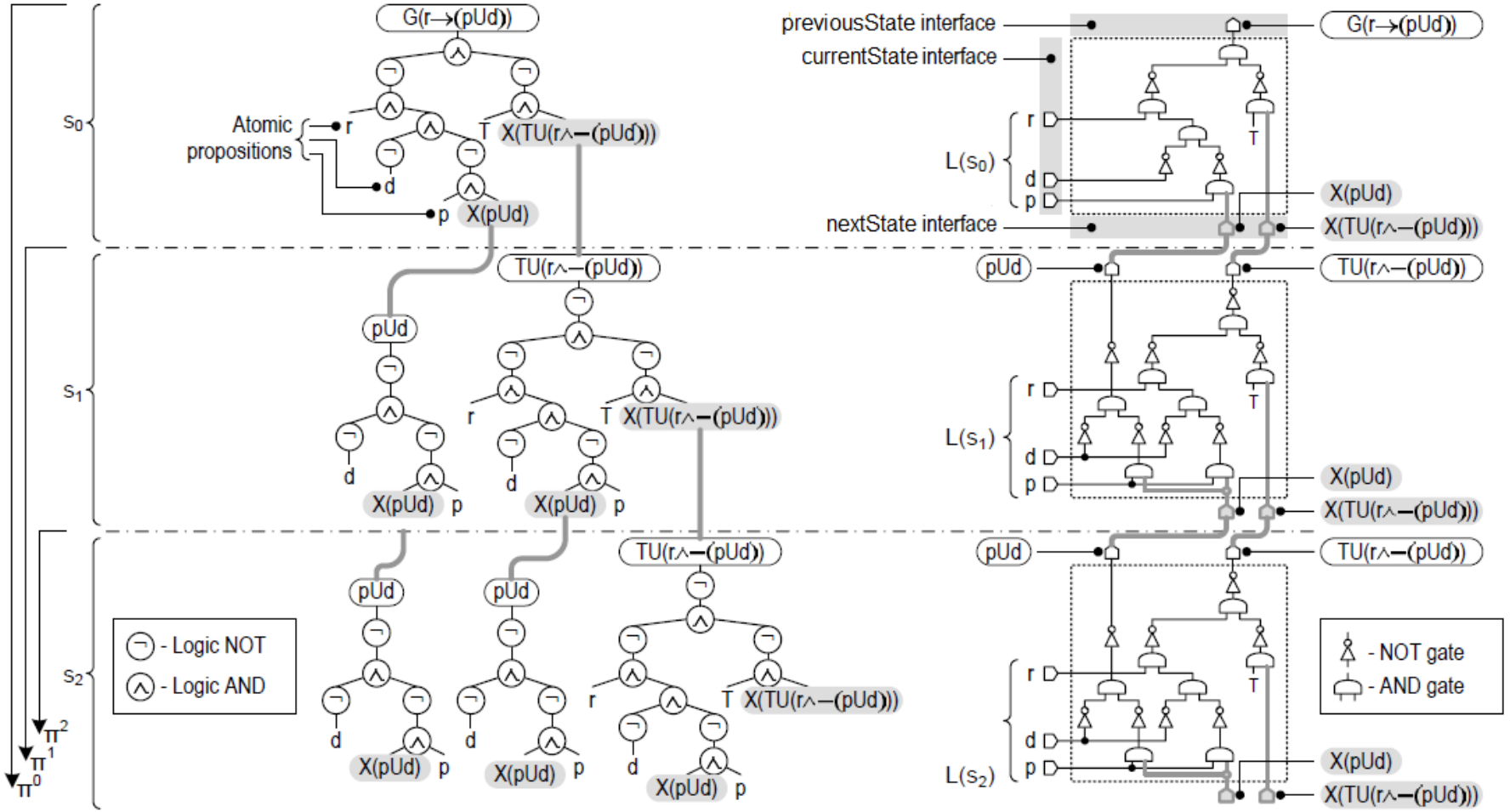
Példa: G(r→(pUd)) felírása

- A G operátor definíciója
- Az U operátor felbontása
  $e_1 U e_2 = e_2 \vee (e_1 \wedge X(e_1 U e_2)) = \neg(\neg e_2 \wedge \neg(e_1 \wedge X(e_1 U e_2)))$
- Normál forma: U csak X argumentumaként



A blokkok optimalizálása

- Ismétlődő részkifejezések egyszeri kiértékelése



Háromértékű logika

- Blokk belseje: Háromértékű (ternáris) logika
  - A következő lépésben történő kiértékelés eredménye „ismeretlen”
  - Az „ismeretlen” legkésőbb a trace végén feloldható
- Műveletek háromértékű logikával:

TERNÁRIS KONJUNKCIÓ:

a	b	$a \wedge_3 b$
$\top$	$\top$	$\top$
$\bot$	$\{\top, \bot, ?\}$	$\bot$
$\{\top, \bot, ?\}$	$\bot$	$\bot$
$?$	$\{\top, ?\}$	$?$
$\{\top, ?\}$	$?$	$?$

TERNÁRIS NEGÁCIÓ:

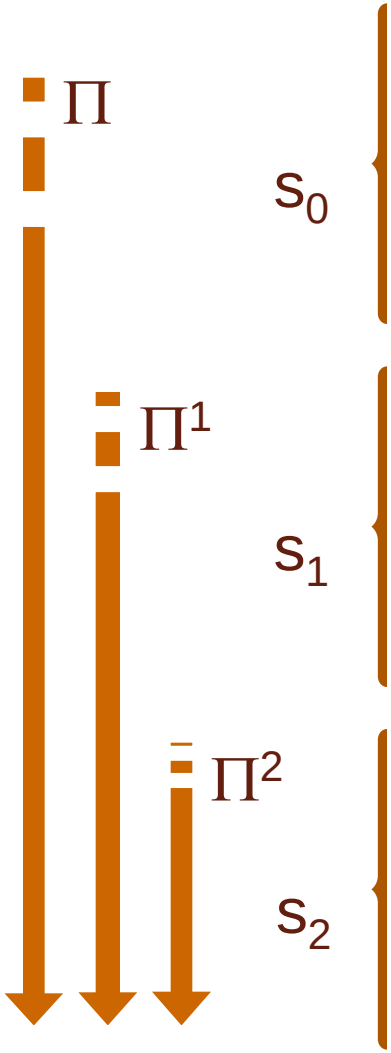
a	$\neg_3 a$
$\top$	$\bot$
$\bot$	$\top$
$?$	$?$

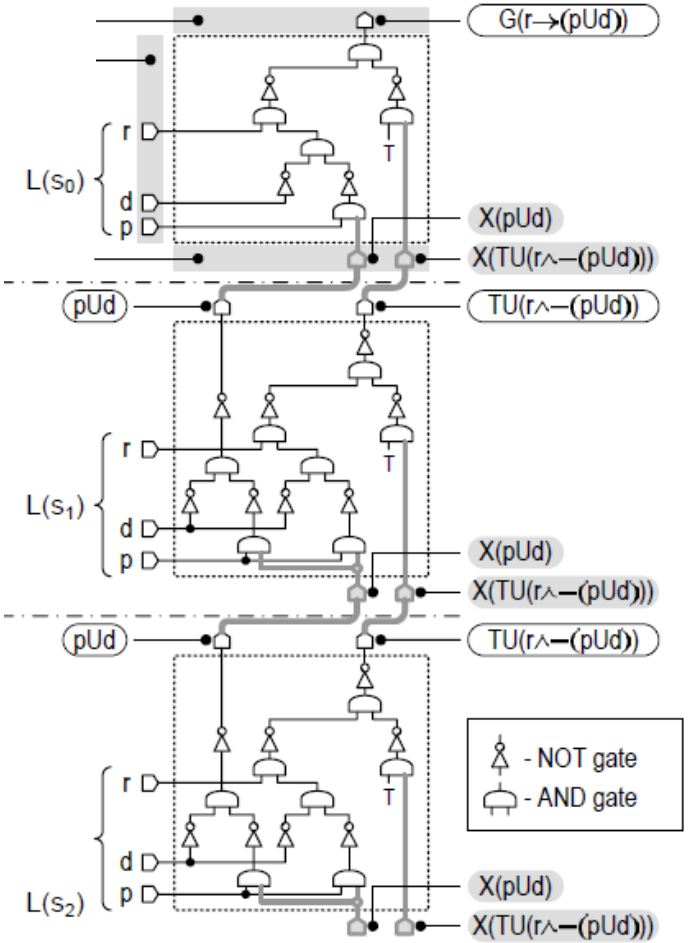
$\top$: IGAZ

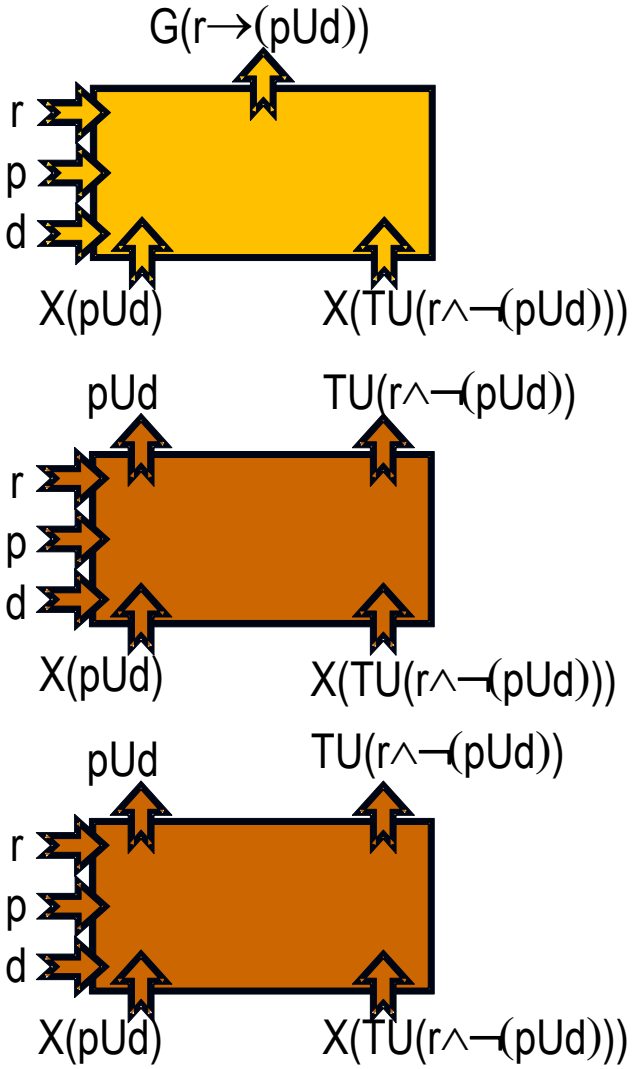
$\bot$: HAMIS

$?$: ISMERETLEN

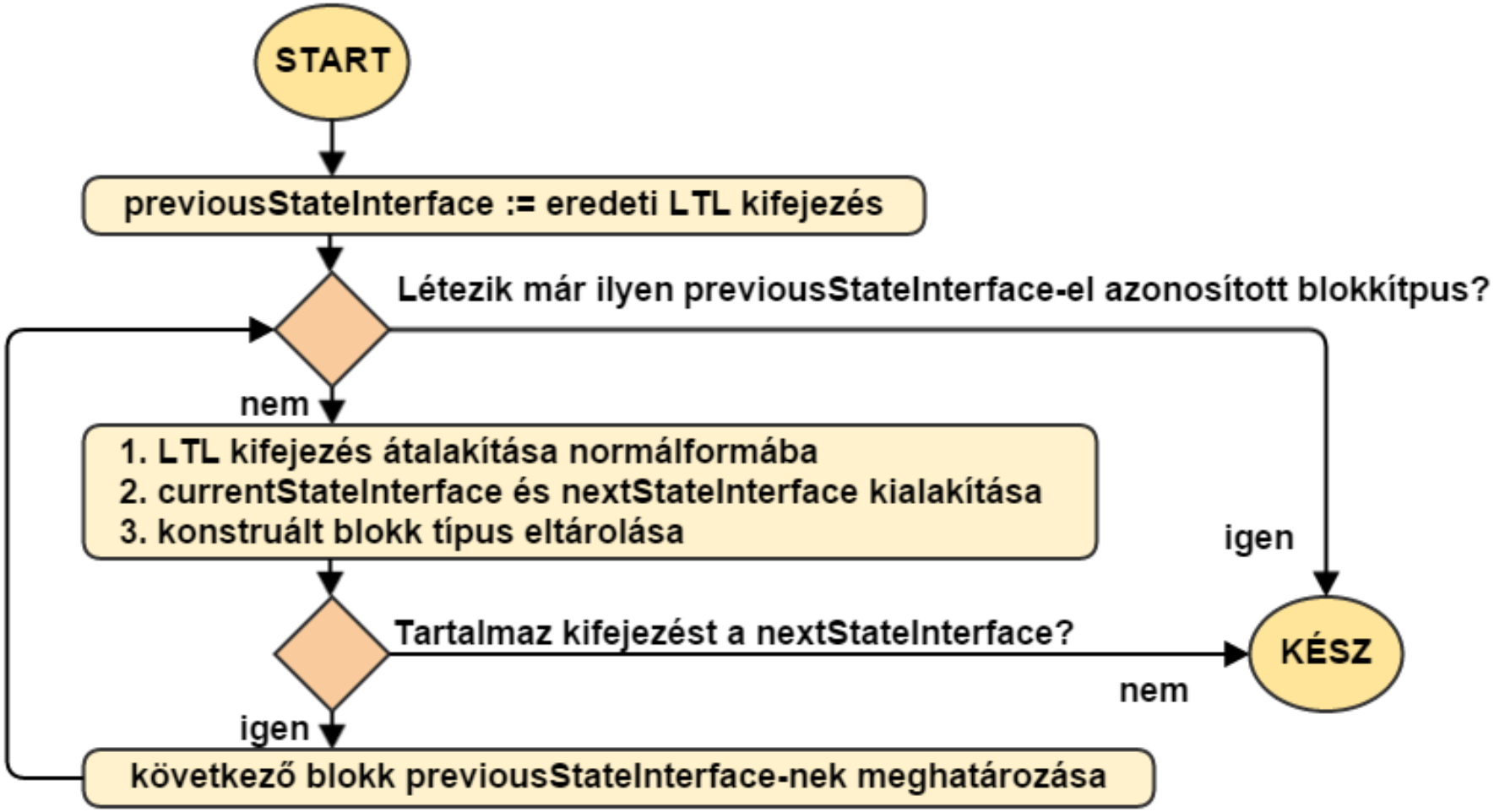
Blokk típusok azonosítása



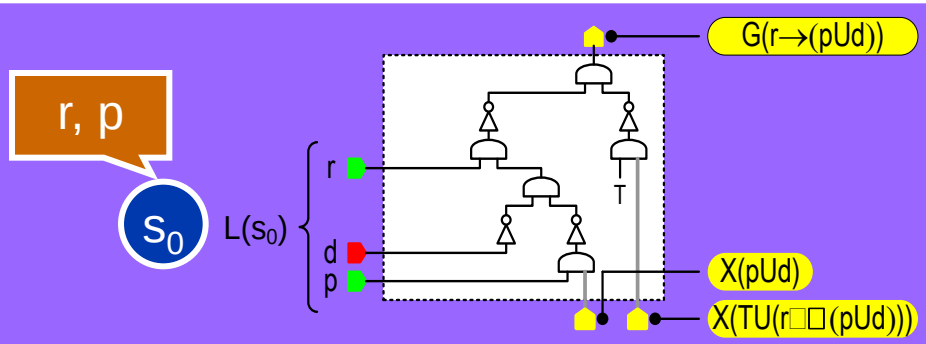




A blokk típusok meghatározása

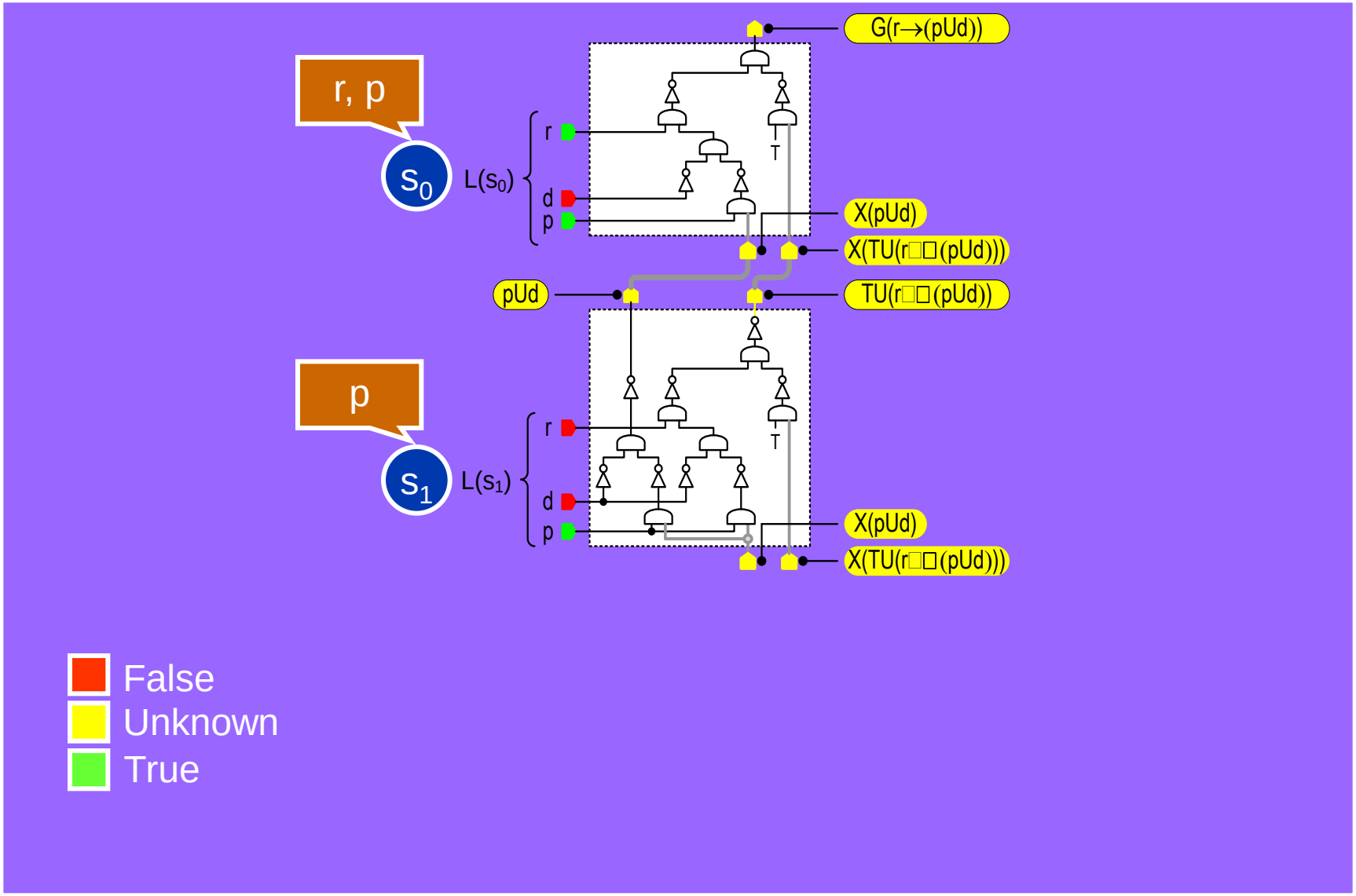


Működési logika: 1. lépés

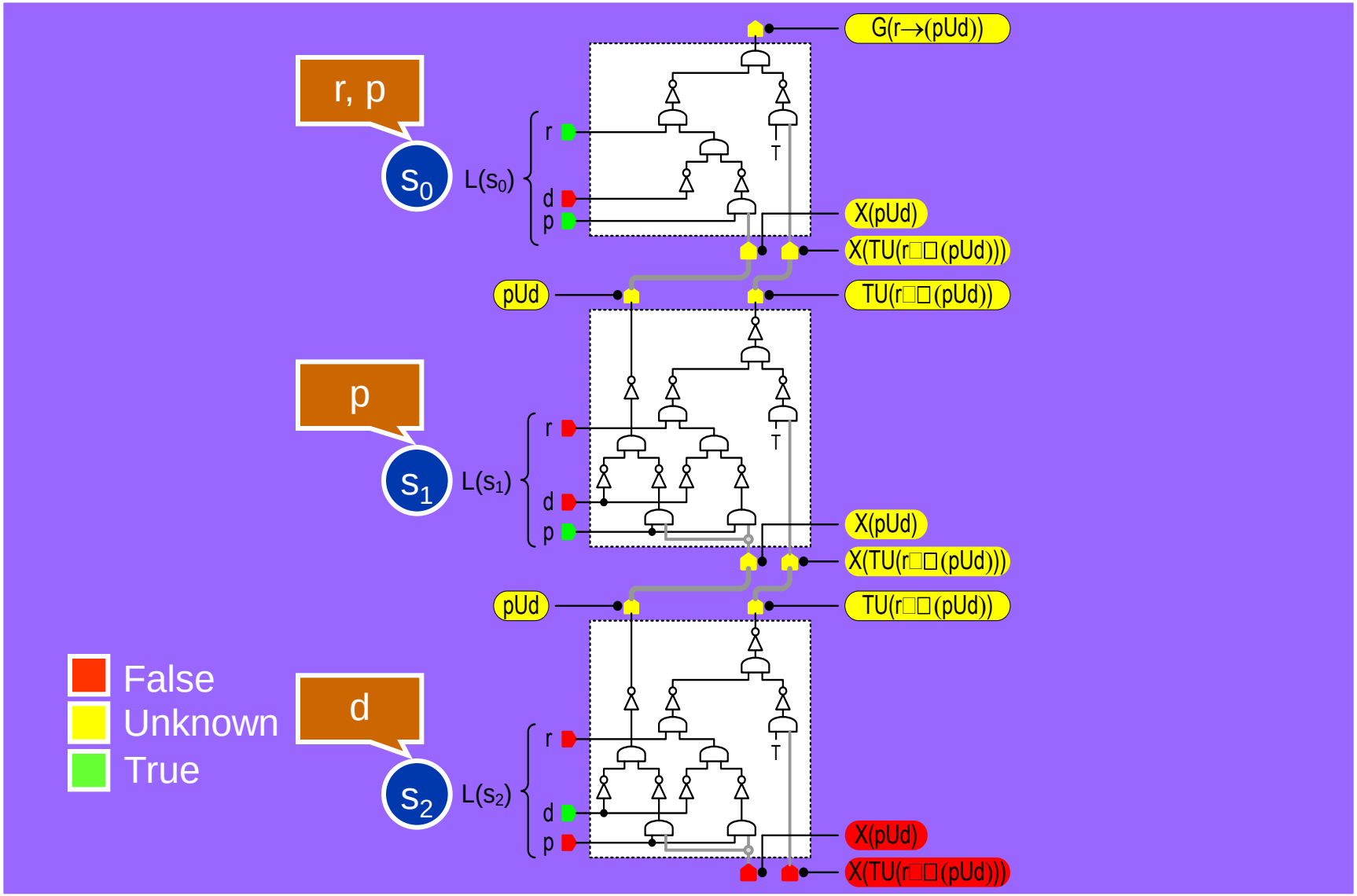


False
Unknown
True

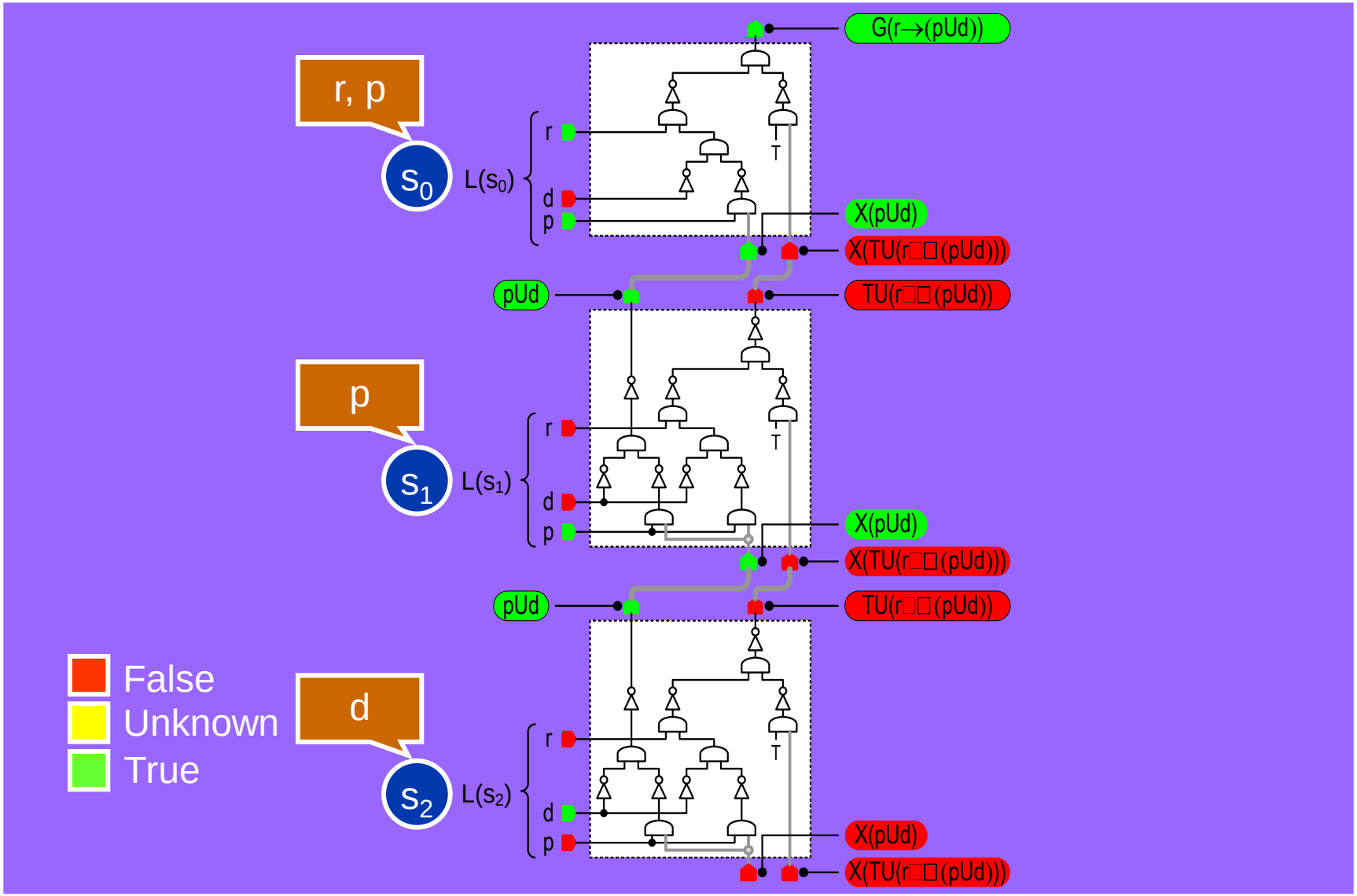
Működési logika: 2. lépés



Működési logika: 3. lépés (trace vége)



Működési logika: Ismeretlenek feloldása

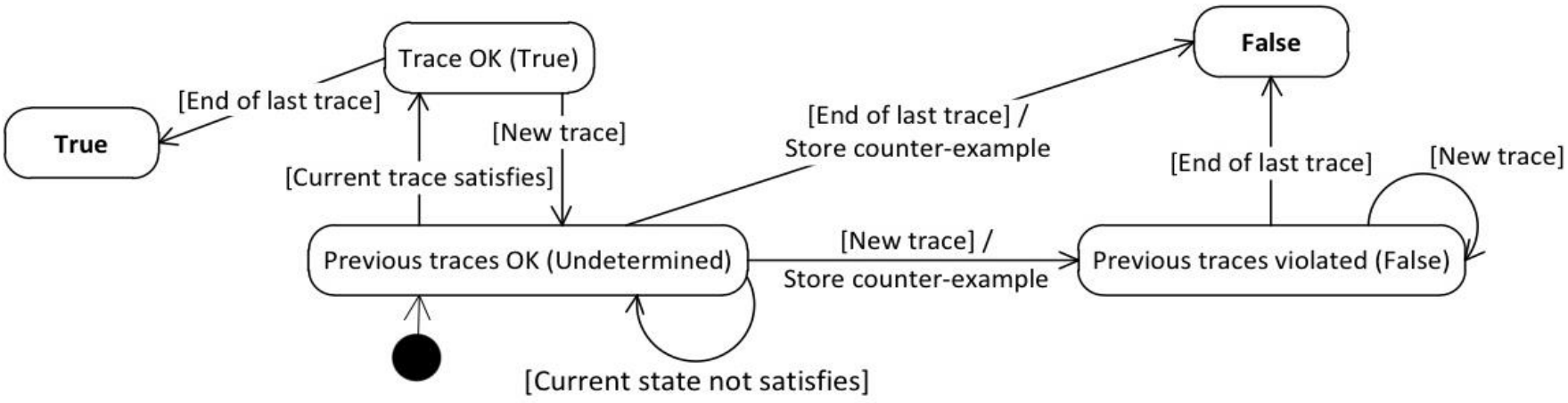


Monitor szintézis LTL alapú monitorozáshoz

- Monitor szintézis lépései
  - Kiértékelő blokk típusok generálása az LTL kifejezés átírása alapján (csatlakoztatási normál forma)
  - A kiértékelő blokk típusok belső logikájához tartozó forráskód generálása (adatstruktúra és kiértékelő fügvények háromértékű logikával)
  - Kiértékelő kontextus generálása a kiértékelő blokk típusok futásidőbeli példányosítására (a blokkok láncának építése)
- A monitorozás memóriaigénye
  - Legfeljebb lineáris a trace hosszával
  - Ismétlődő blokkok redukálhatók
- Összevetés más implementációkkal:
  - Symbolic term-rewriting: 30…160-szor lassabb
  - Alternating automata: 15…40-szer lassabb

CTL alapú monitorozás (UPPAAL TCTL)

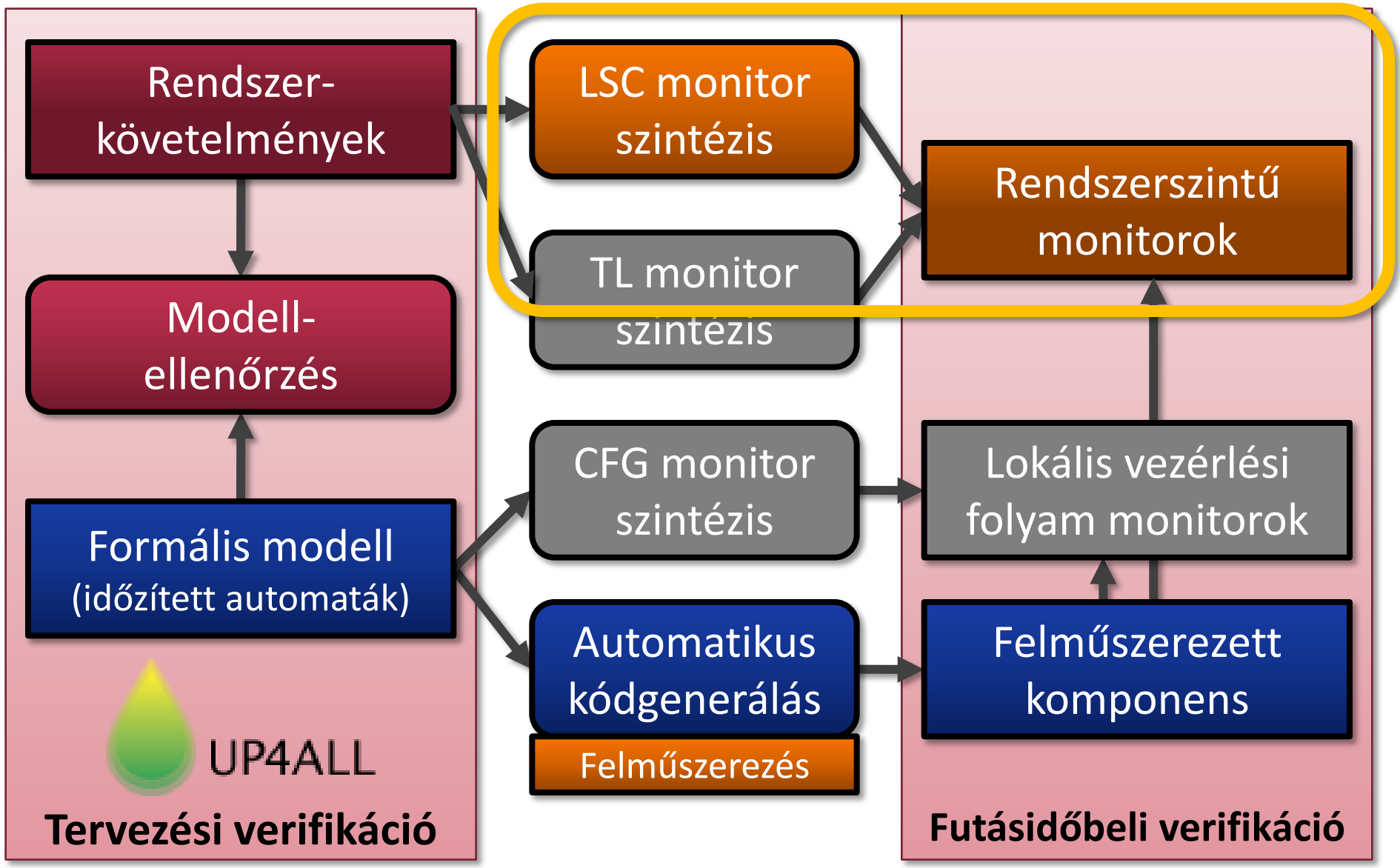
- Használható trace-ek halmazának ellenőrzésére
  - Útvonal kvantorok: A: “Minden útvonal …”, E: “Létezik útvonal …”
- A monitorok mint teszt oracle használhatók egy teszt készlet esetén a lefutások halmazának ellenőrzésére
  - Speciális események: <New trace>, <End of last trace>
- Monitor szintézis
  - Egy trace ellenőrzése: Mint az LTL alapú ellenőrzés
  - Trace-ek sorozatának ellenőrzése (test suite): Megfigyelő automata
- Példa: Megfigyelő automata AF temporális operátorhoz



Futásidejű verifikáció LSC követelmények alapján

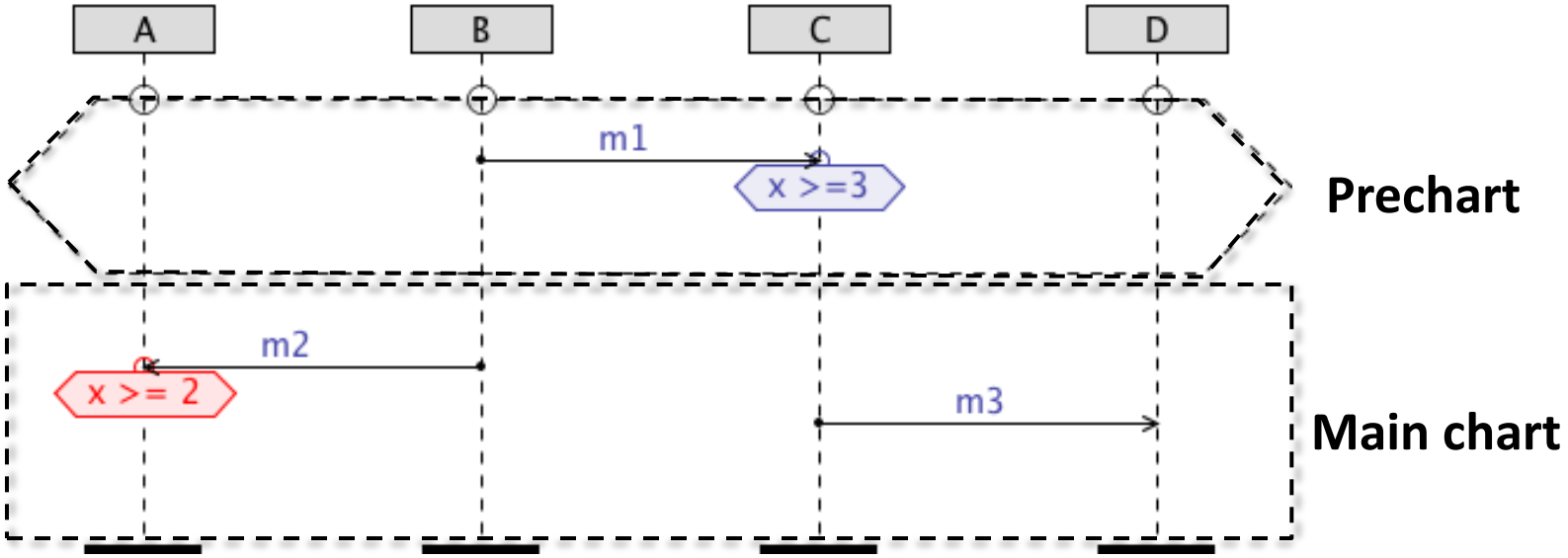
Live Sequence Charts követelmények verifikációja

Hierarchikus monitorozás

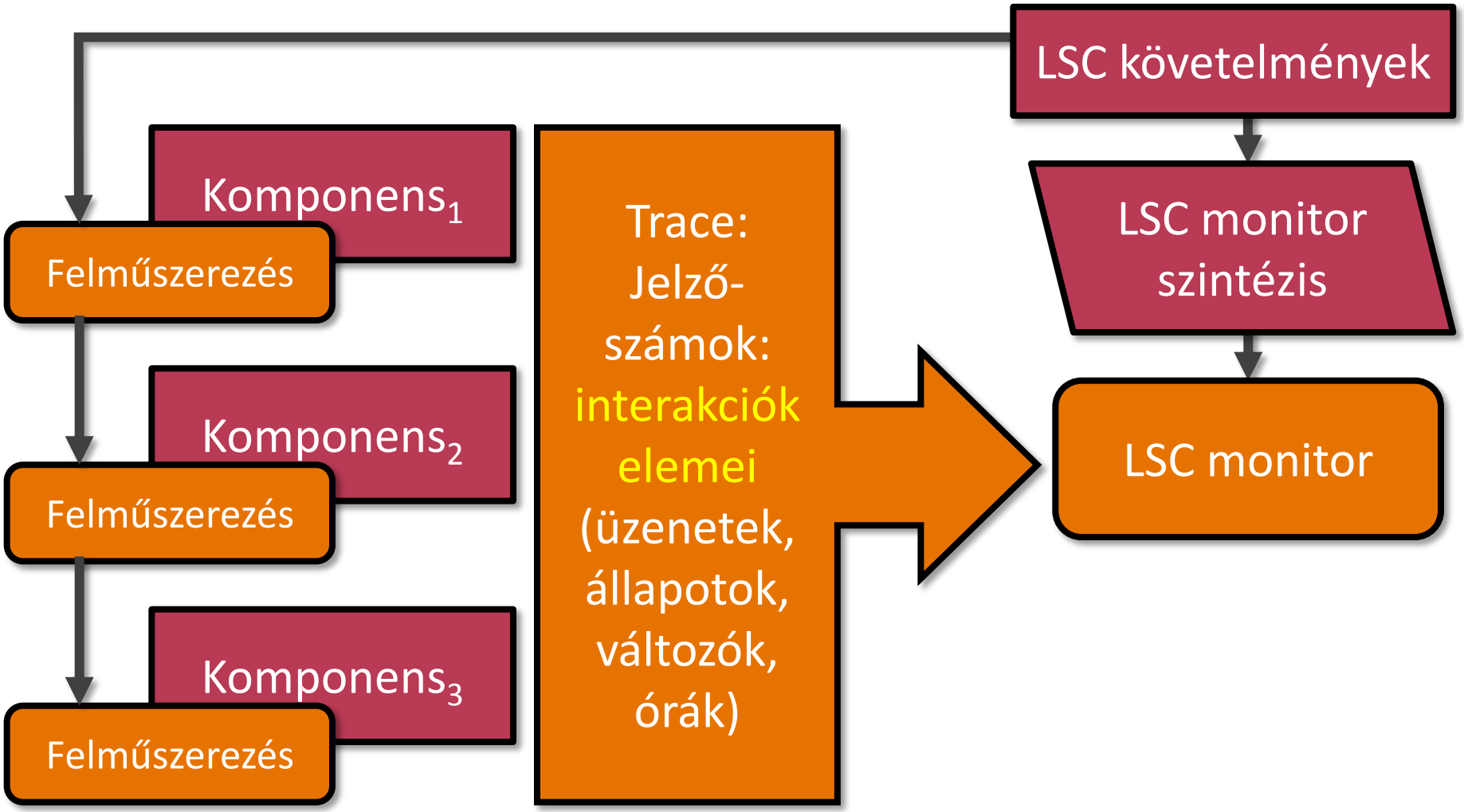


LSC alapú követelmények

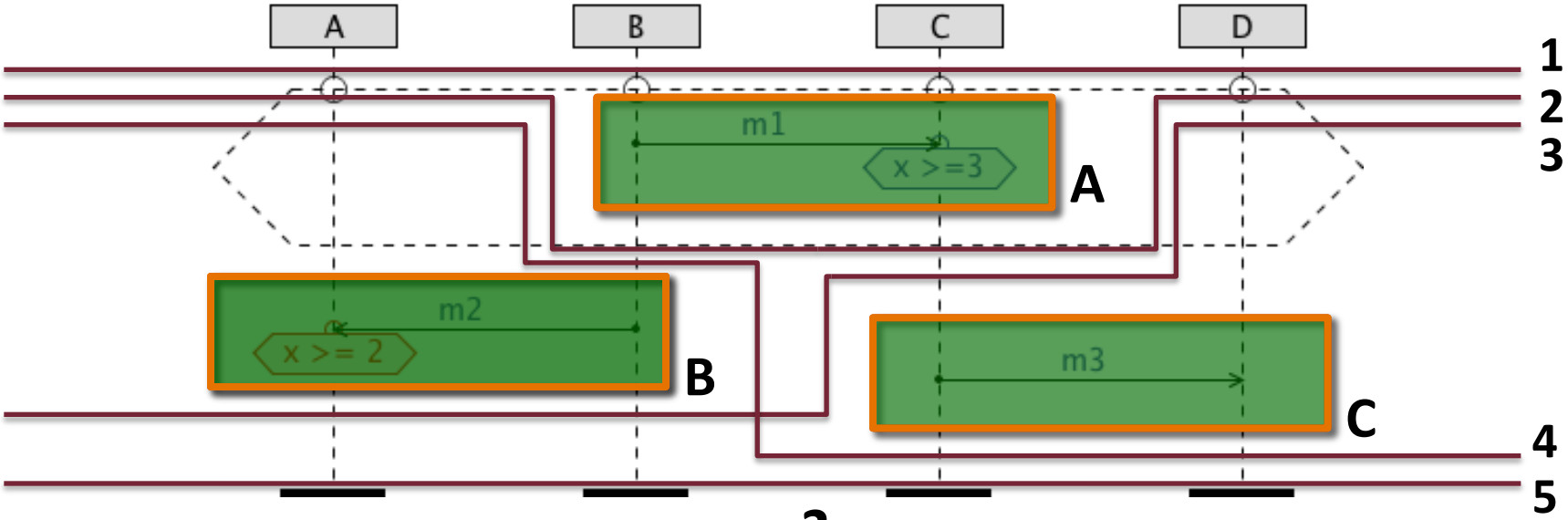
- Cél: Interakciók ellenőrzése intuitív leírás alapján
  - Szinkronizáció, üzenetek küldése és fogadása, lokális feltételek
- Formalizmus: Live Sequence Charts variáns
  - Message Sequence Chart kiterjesztése (Damm és Harel, 2001)
  - Precíz szemantika (UPPAAL: korlátozásokkal)
  - LSC alaptípusok, aktiválási módok, hideg és forró feltételek

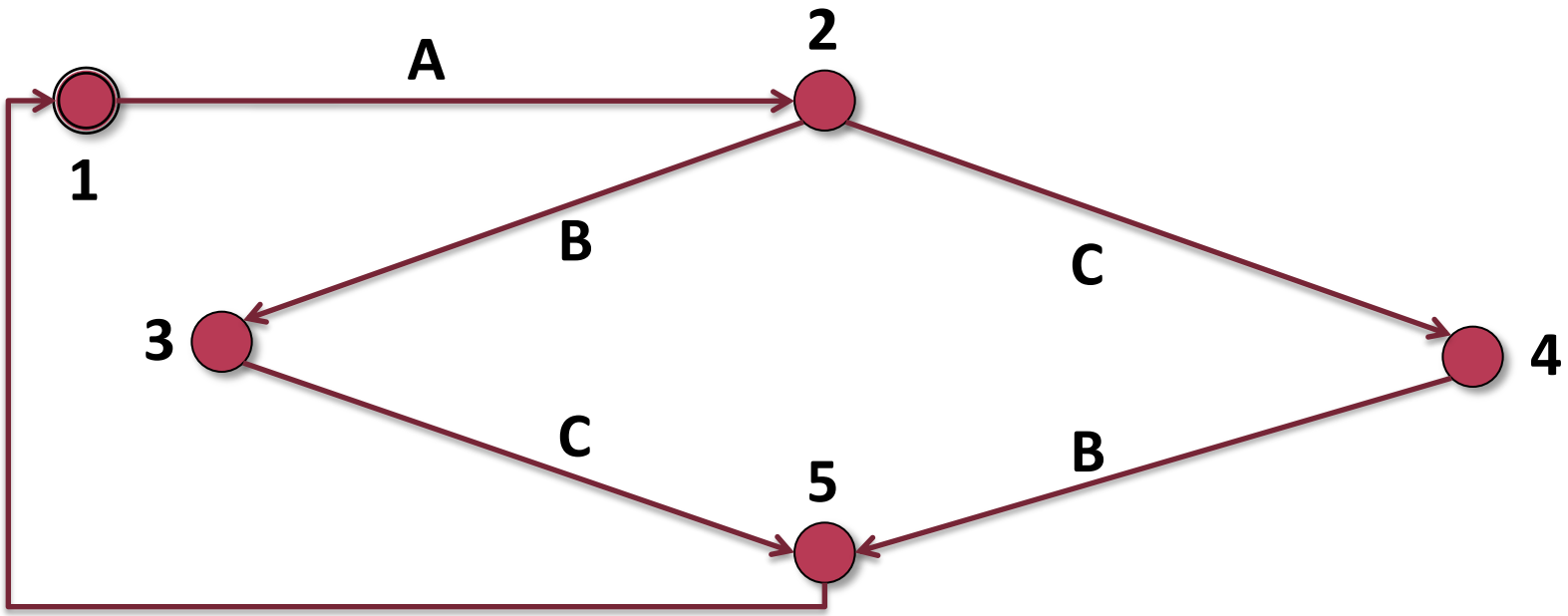


LSC követelmények ellenőrzése



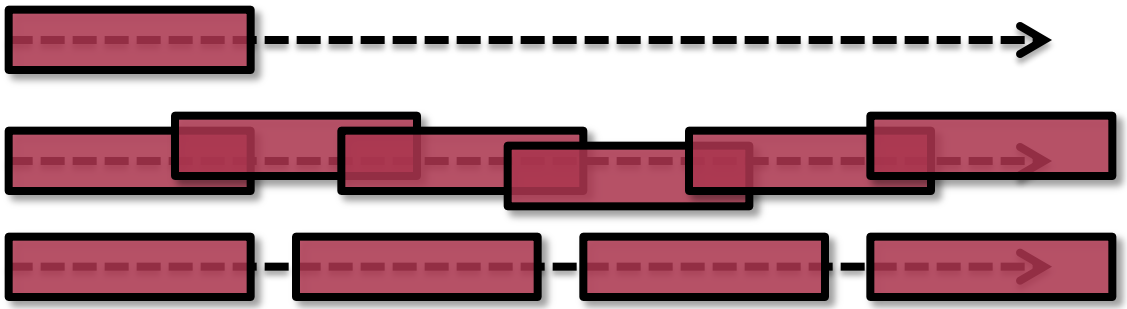
LSC monitor szintézise





LSC monitorok végrehajtása

- Ütemező (végrehajtási környezet) szükséges
  - Egy-egy monitor indítása, leállítása
  - Visszajelzések kiértékelése, továbbítása
- LSC alaptípusok támogatása
  - Egzisztenciális: minta viselkedés (lefutásokra)
  - Univerzális: szükséges viselkedés (minden lefutásra)
- LSC aktiválási módok támogatása
  - Kezdeti
  - Invariáns
  - Iteratív



Futásidejű verifikáció scenario követelmények alapján

Kontextusfüggő viselkedés monitorozása

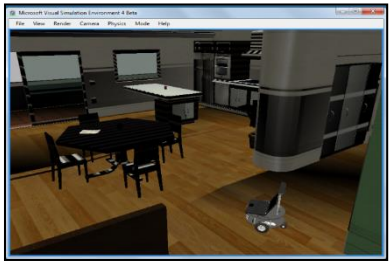
Speciális kihívások

- Autonóm rendszerek (pl. robotok) ellenőrzése
  - Kontextusfüggő viselkedés (környezet érzékelése)
  - Adaptív rendszer (célok, végrehajtási stratégia)
- Követelmények specifikálása: Scenariok
  - Viselkedés: Események és akciók szekvenciája; Feltétel rész és kötelező (assert) rész
  - Hivatkozások a kontextusra (szituációkra)
- Kontextusfüggő viselkedés monitorozása:
  - Megfigyelni a futásidőbeli kontextus változásait
  - Ellenőrizni a rendszer viselkedését

A monitorozás szerepe

Valós környezet

vagy



Szimulátor

Követelmények

Futási trace-ek

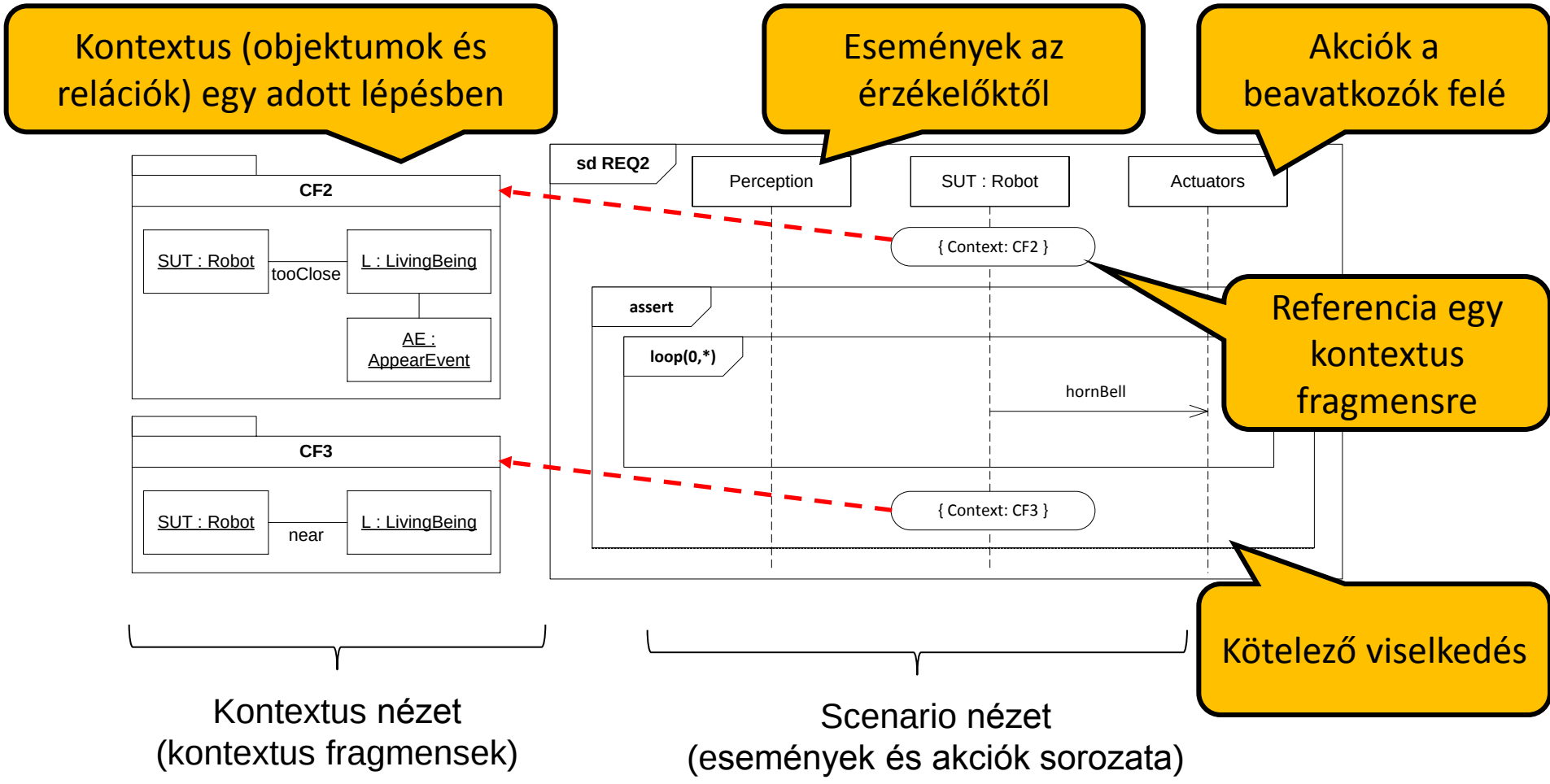
Kihívás:
Egy-egy trace hatékony ellenőrzése a scenario követelmények alapján

Megfigyelt események, akciók, kontextus változások

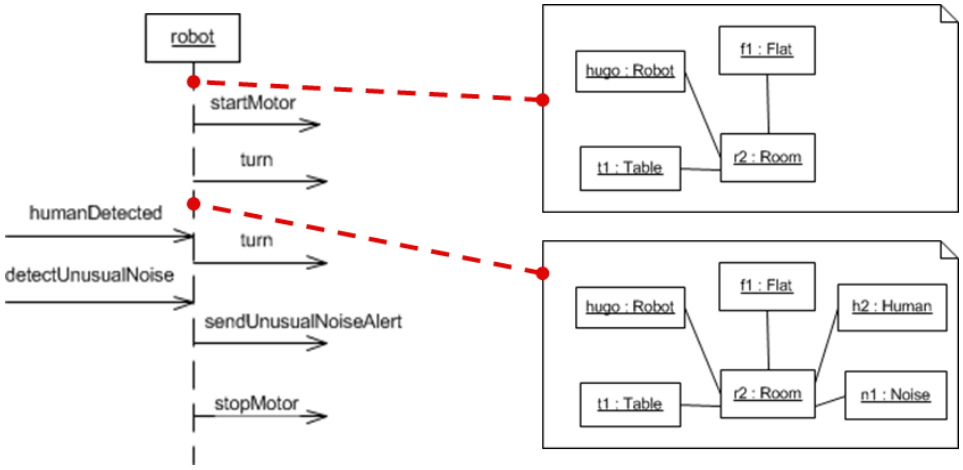
Kiértékelés a monitorral

Követelmények formalizálása

- MSC (vagy LSC) alapú esemény- és akció szekvenciák
- Kiterjesztve kontextus hivatkozásokkal

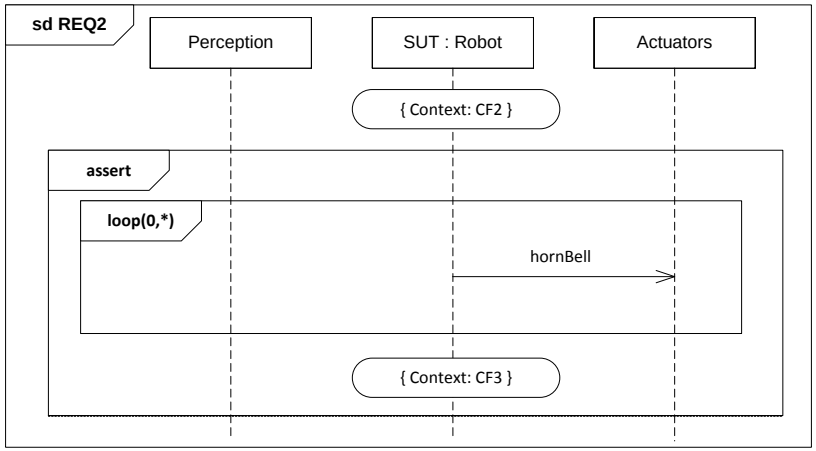


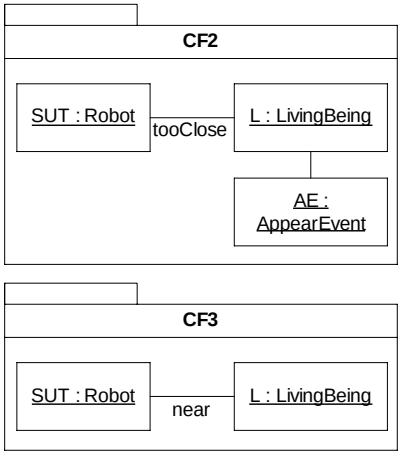
A monitor feladatai



Megfigyelt trace:

- Események és akciók a SUT-tól
- A kontextus konkrét konfigurációi



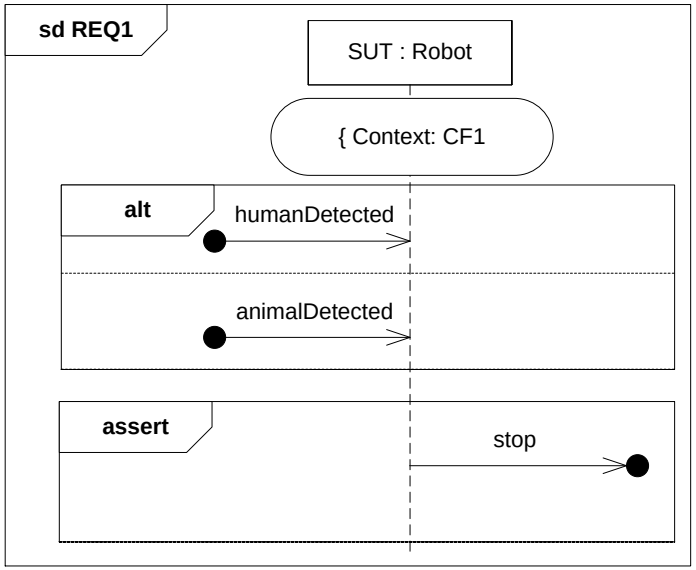


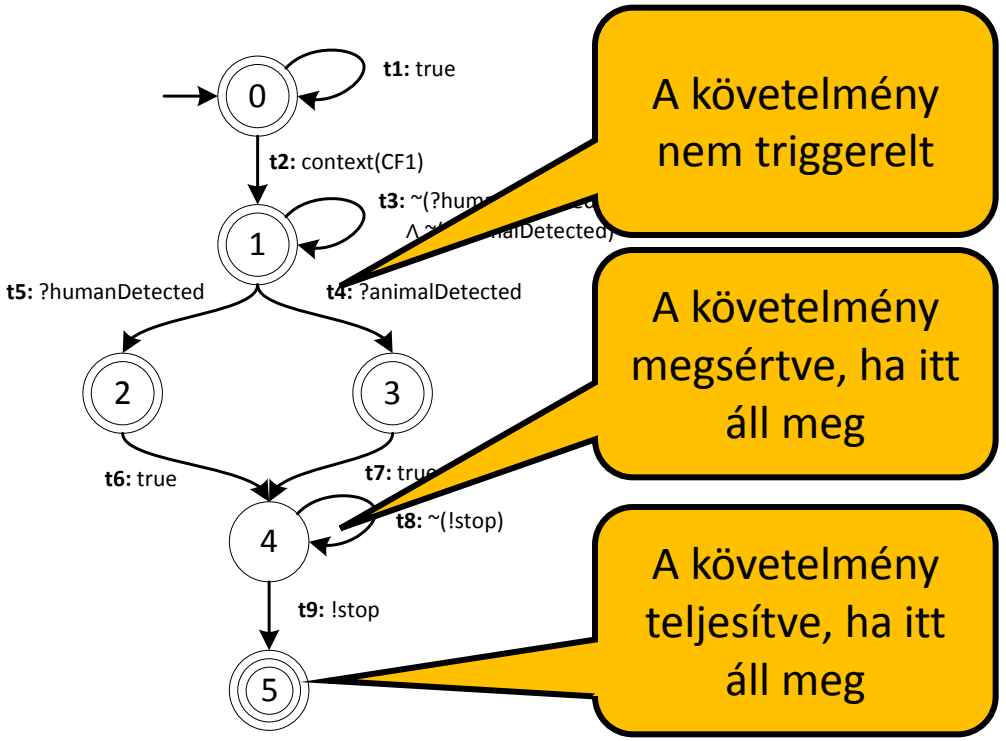
Események, akciók illesztése:
Megfigyelő automata

Kontextus fragmensek illesztése:
Gráfminta illesztés

A megfigyelő automata konstrukciója

- Egy-egy megfigyelő automata a követelmény scenariokhoz
  - Állapotok: “vágások” a diagramon (mint az LSC esetén)
  - Tranzíciók: események, akciók, kontextus változások
  - Állapot típusok: teljesítve / megsértve / nem triggerelt

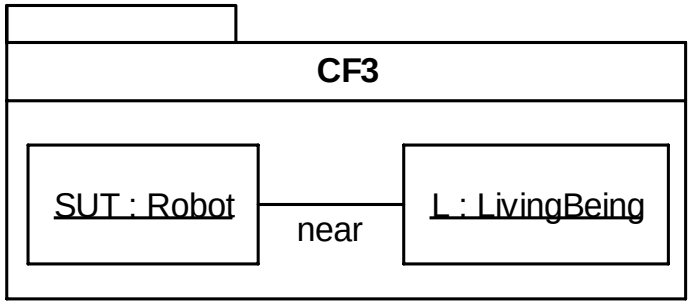




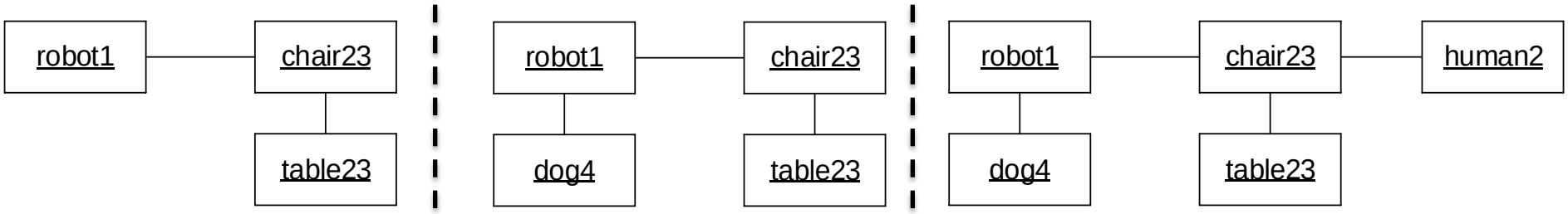
Gráfminta illesztés

- A trace-ben megfigyelt kontextus szekvencia illesztése
  - A kontextusok gráf alapú reprezentálása
  - Gráf minták (követelmény kontextus fragmensek) illesztése gráf szekvenciákhoz (megfigyelt kontextus a trace-ben)

Kontextus fragmens (követelmény):



Megfigyelt kontextus sorozat:



Megoldandó problémák

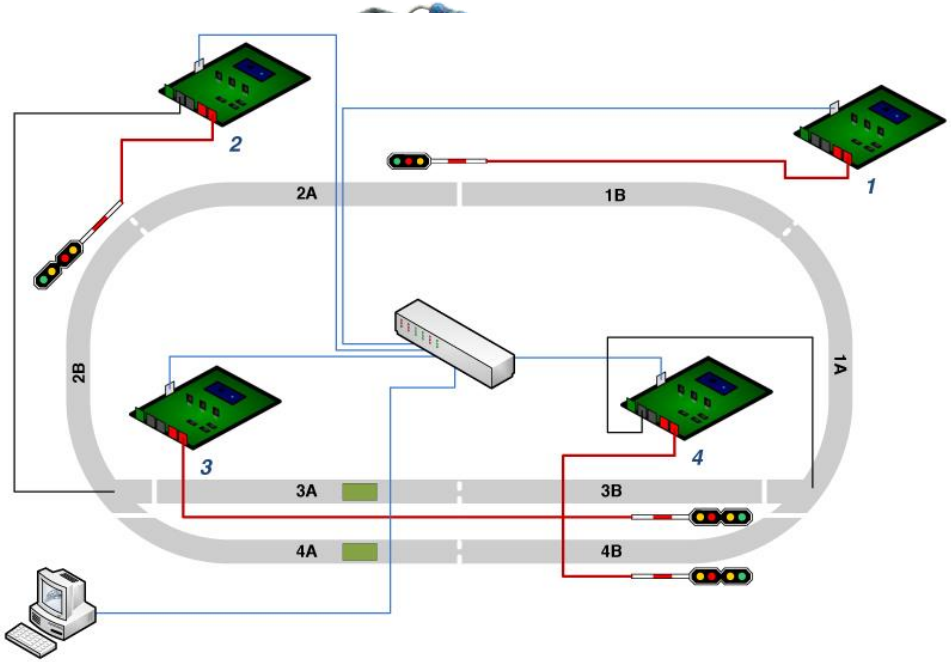
- Minden követelmény scenario illesztése egy trace-hez
  - A kontextus fragmensek dekompozíciója a közös részek egyszeres tárolásához és illesztéséhez
- A követelmény kontextus fragmensek illesztése egy trace minden lépésében
  - Konkurens monitorok indítása, amennyiben illeszkedik
- Az absztrakt relációk (pl. “közel”) és az objektum hierarchia (pl. az “asztal” egy “bútor”) feloldása
  - A trace előfelfolgozása az absztrakt relációk származtatásához
  - A típusok kompatibilitásának kezelése a kontextus elemek illesztésekor

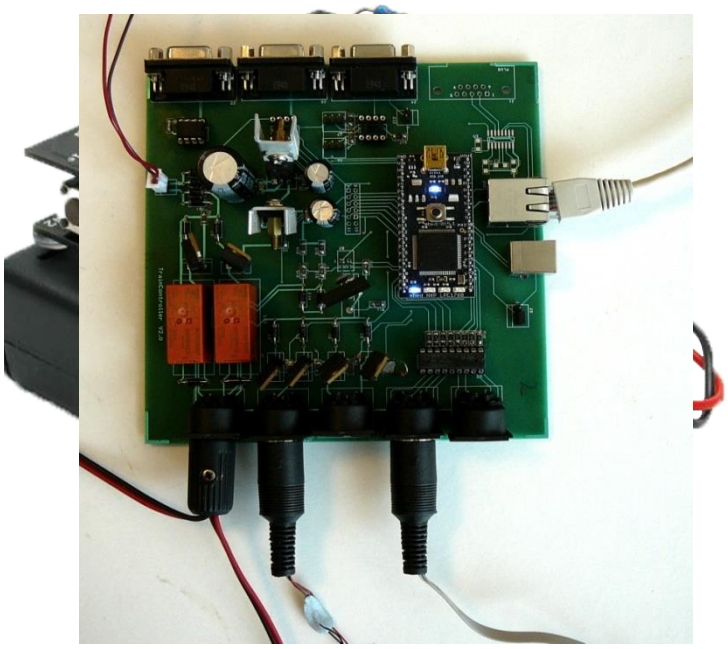
Megvalósítási tapasztalatok

Erőforrásigények felmérése

Megvalósítás

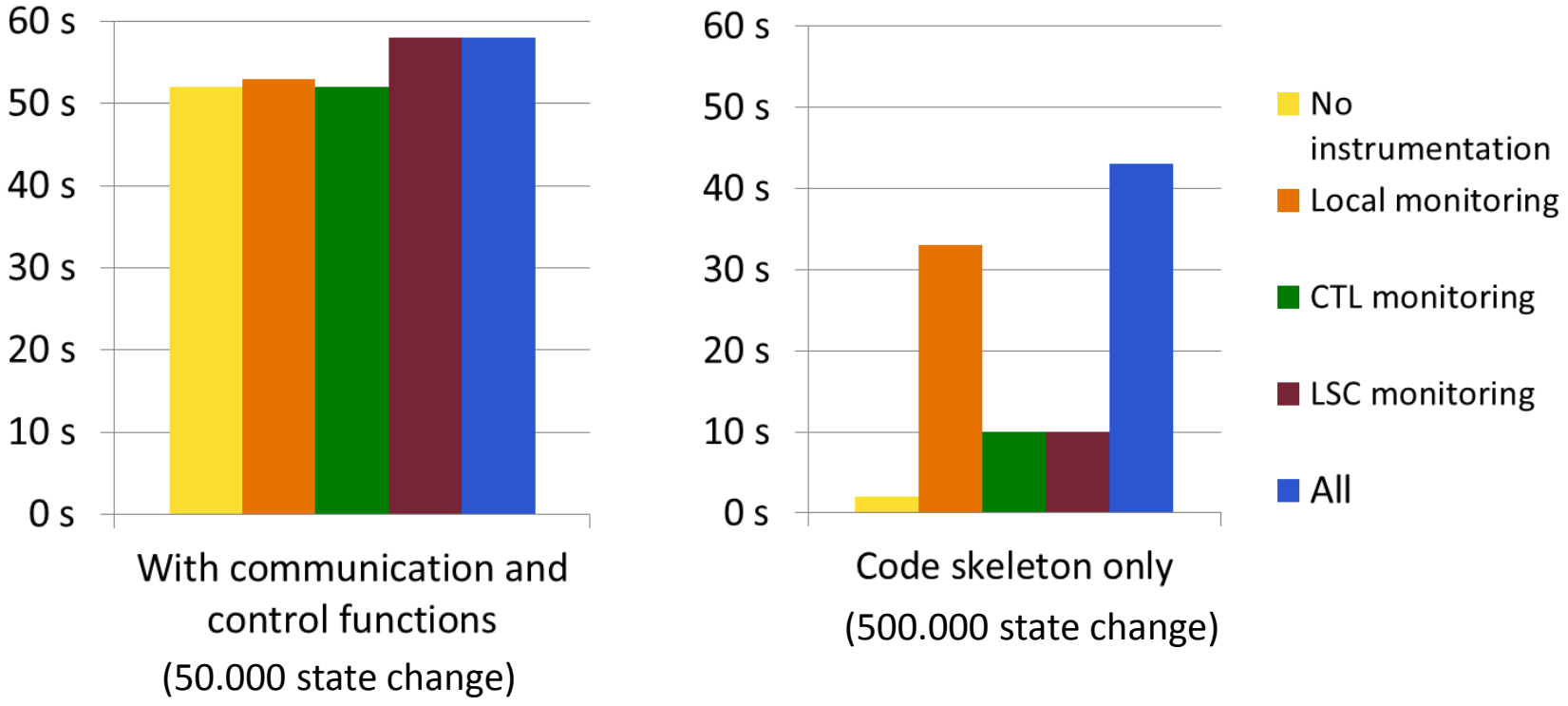
- Beágyazott platformokon
  - ***mitmót*** moduláris beágyazott rendszer vezetéknélküli kommunikációval
    - Ipari esettanulmány: bit szinkronizációs protokoll
  - ***mbed*** mikrokontroller gyors prototípuskészítéshez
    - Oktatási demonstrátor környezet





Jellemző erőforrásigények

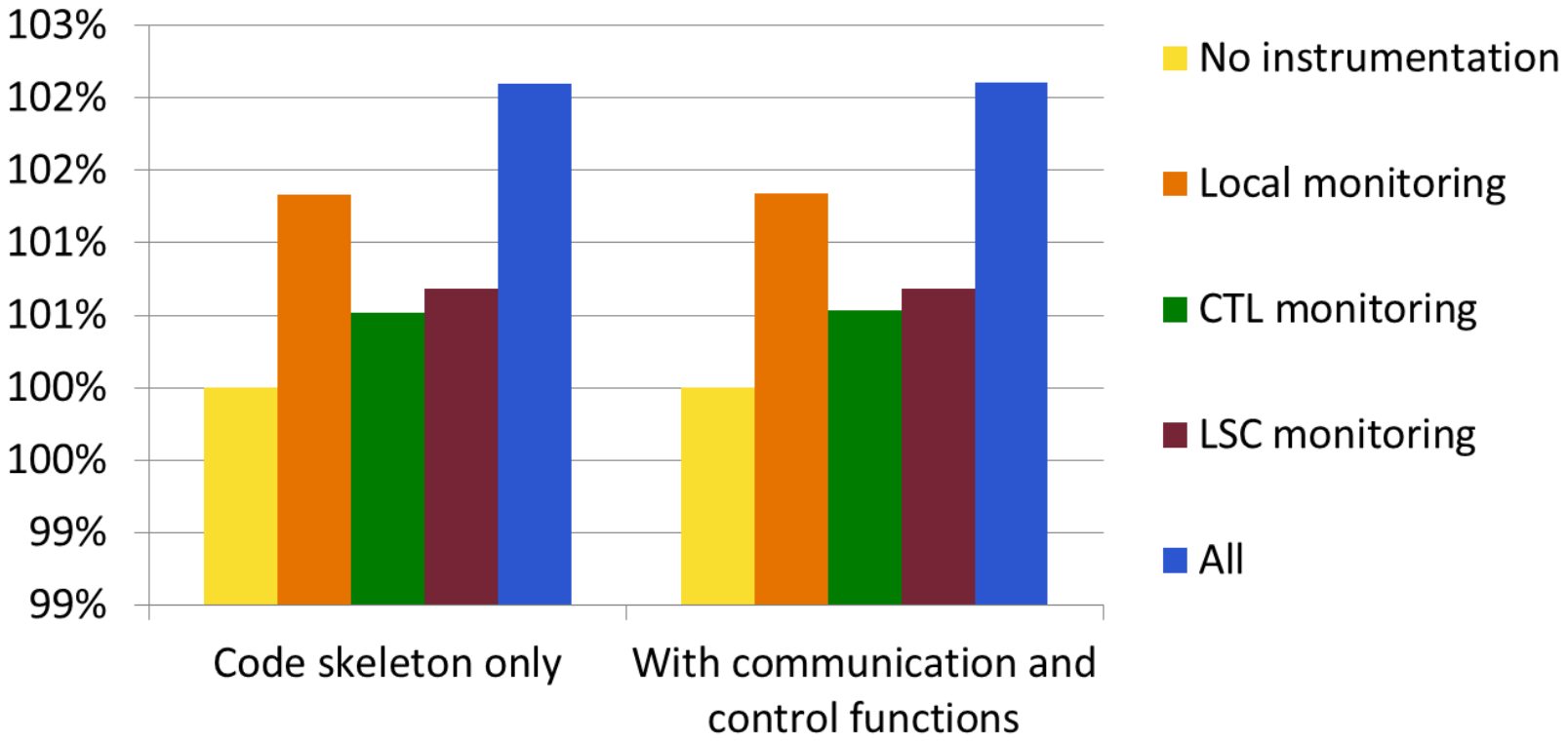
- Az ellenőrzés időigénye (mbed platform)



- Kisebb, mint 12% többletidő
- Nagy overhead gyors vezérlési funkciókra

Jellemző erőforrásigények

- Kódméret növekedés (mbed platform)



- Kisebb, mint 5% kódméret növekedés

Scenario alapú monitorozás

- Prototípus megvalósítás
  - Scenario követelmények: Eclipse UML2 alapon
  - Monitor (teszt oracle): Java alkalmazás
- Komplexitás: a gráfminták illesztése
  - Legjobb eset: $O(IM)$, legrosszabb eset: $O(NI^M M^2)$
    - N: a követelmény gráf minták száma
    - M: a követelmény gráf minták átlagos mérete
    - I: a trace-ben illesztendő kontextus gráf csúcsainak száma
  - A követelmény gráf minták (kontextus fragmensek) általában kicsik (így M értéke alacsony)

Összefoglalás

- Monitor szintézis futásidőbeli verifikációhoz
  - Automata alapú referencia viselkedés ellenőrzés
    - Időzített automata modellek alapján
    - UML állapottérkép modellek alapján:
      Monitorozás precíz szemantika szerint
  - LTL követelmények hatékony verifikációja
    - Ellenőrző blokkok konstrukciója és példányosítása
  - LSC követelmények futásidőbeli verifikációja
  - Kontextusfüggő viselkedés ellenőrzése scenario követelmények alapján
    - Események, akciók és kontextus fragmensek illesztése