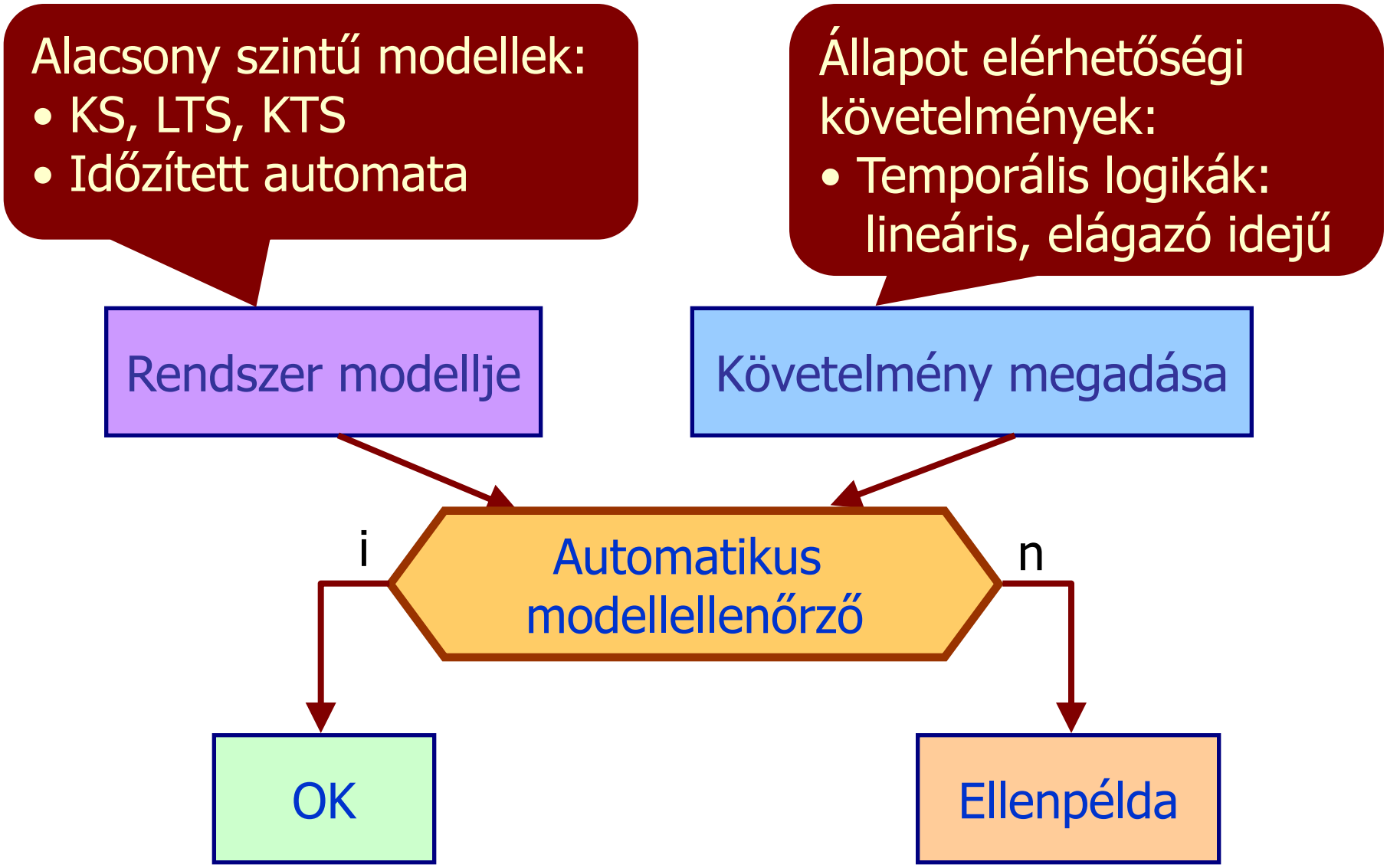
Követelmények formalizálása: Elágazó idejű temporális logikák

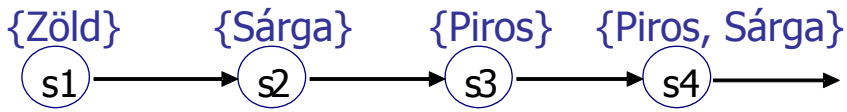
dr. Majzik István
BME Méréstechnika és Információs Rendszerek Tanszék

Ismétlés: Mit szeretnénk elérni?

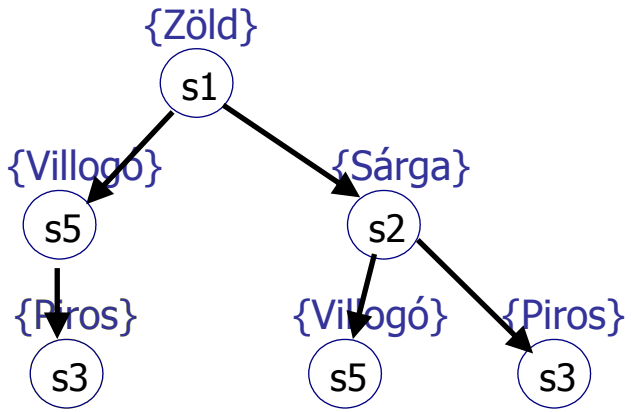


Ismétlés: Temporális logikák osztályozása

- Lineáris idejű:
  - A modell egy-egy végrehajtását (lefutását) tekintjük
  - Minden állapotnak egy rákövetkezője van
  - Logikai idő egy idővonal mentén (állapotsorozat)

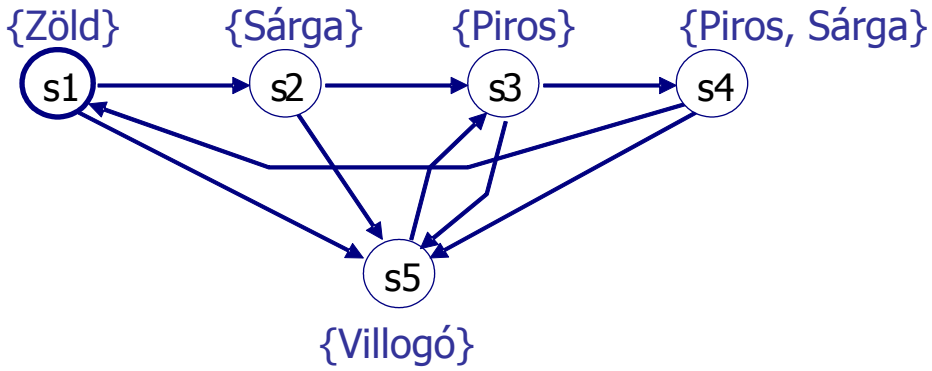


- Elágazó idejű:
  - A modell minden lehetséges végrehajtását tekintjük
  - Az állapotoknak több rákövetkezője lehet
  - Logikai idő elágazó idővonalak mentén jelenik meg (számítási fa)

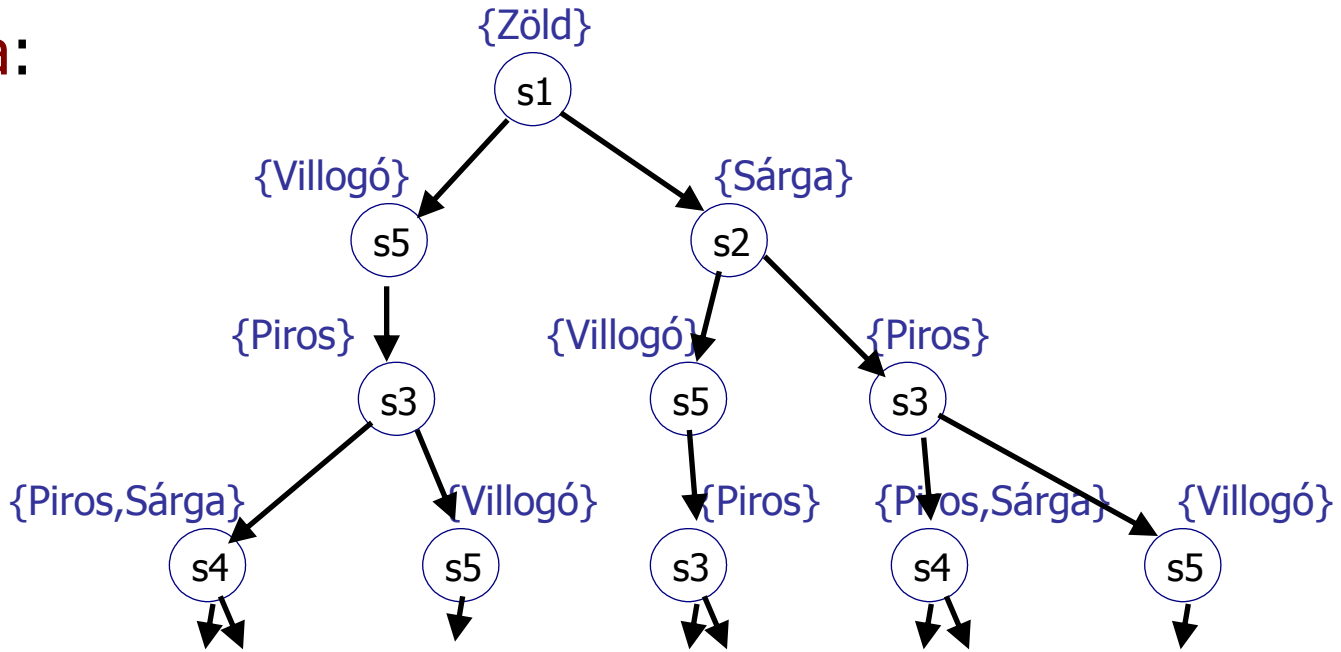


Számítási fa konstrukciója

Kripke-struktúra:



Számítási fa: Lehetséges elágazások

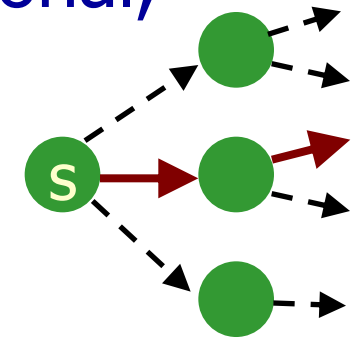


Elágazó idejű temporális logikák: CTL, CTL*

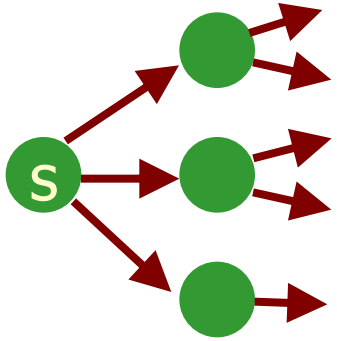
Elágazások vizsgálata

Egy-egy állapotban előírható,
hogy az útvonalakra vonatkozó p követelmény
hány onnan kiinduló útvonal mentén teljesüljön:

- E p (Exists p): Létezzen legalább egy útvonal, ahol a p követelmény teljesül
  - Egy lehetséges továbblépés mentén vizsgál
  - Egzisztenciális operátor



- A p (forAll p): Minden útvonalra fennálljon, hogy a p követelmény teljesül
  - Minden lehetséges továbblépést magába foglal
  - Univerzális operátor



Elágazó idejű temporális logikák

- CTL*: Computational Tree Logic *
  - Útvonal kvantorok (E, A), és
  - útvonalakon értelmezett temporális operátorok (X, F, G, U)

  tetszőleges kombinációja

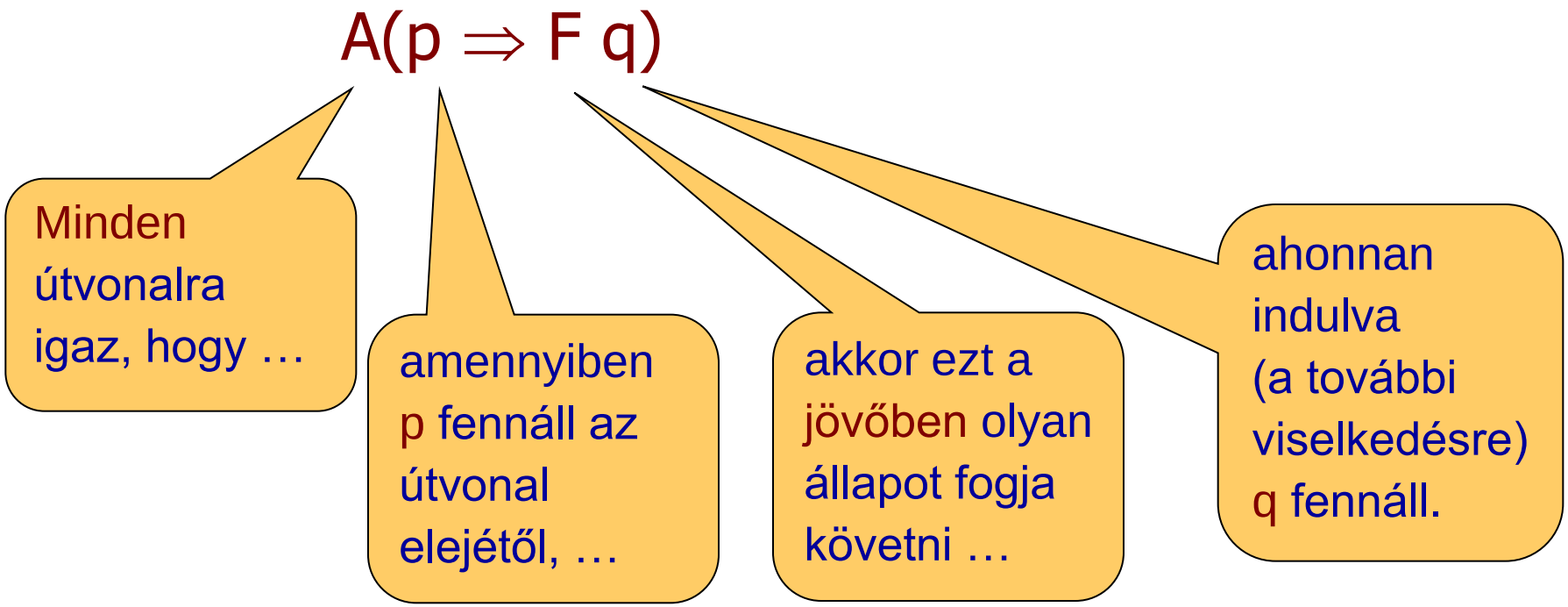
- CTL: Computational Tree Logic
  - Útvonalakon értelmezett temporális operátorokat mindig közvetlenül meg kell előznie útvonal kvantoroknak
  - Útvonalakon értelmezett operátorok nem kombinálhatók

CTL*: Computational Tree Logic *

CTL* operátorok (informális)

- Útvonalak kvantorai (állapotokon értelmezett):
  - A: „for All futures”,
    minden lehetséges útra az adott állapotból kiindulva
  - E: „Exists future”, „for some future”,
    legalább egy útra az adott állapotból kiindulva
- Útvonalakon értelmezett operátorok:
  - X p: „neXt”, a következő állapotban p igaz
  - F p: „Future”, valamikor az útvonal egy állapotán p igaz
  - G p: „Globally”, az útvonal minden állapotán p igaz
  - p U q: „p Until q”, az útvonal egy állapotán igaz lesz q,
    és addig minden állapotban igaz p

CTL* kifejezések



Példa CTL* kifejezések

- $E(p \wedge G\ q)$

  Létezik olyan útvonal, hogy
  ezen p fennáll (az útvonal elejétől nézve) és az útvonal minden állapotából (az útvonal szuffixén) q is fennáll

- $E(XXX\ p \vee F\ q)$

  Létezik olyan útvonal, hogy

  - vagy ennek negyedik állapotán fennáll p,
  - vagy valamikor q fennáll az útvonalon

A CTL* formális kezelése

- Eddigiek: Csak informális bevezetés volt
- Az automatikus ellenőrzést is lehetővé tevő precíz megadáshoz szükséges:
  - Formális szintaxis szabályok:
    Mik az érvényes CTL* kifejezések?
  - Formális szemantika szabályok:
    Adott modellen mikor igaz egy CTL* kifejezés?

CTL* szintaxis

- Állapot-kifejezések: Állapotokon kiértékelhető
  - **S1**: Minden P atomi kijelentés egy állapot-kifejezés
  - **S2**: Ha p és q állapot-kifejezések, $\neg p$ és $p \wedge q$ is
  - **S3**: Ha p útvonal-kifejezés, akkor E p és A p állapot-kifejezések.

- Útvonal-kifejezések: Útvonalakon kiértékelhető
  - **P1**: Minden állapot-kifejezés útvonal-kifejezés
  - **P2**: Ha p és q útvonal-kifejezések, akkor $\neg p$ és $p \wedge q$ is
  - **P3**: Ha p és q útvonal-kifejezések, akkor X p és p U q is

Érvényes CTL* kifejezések:
A szabályok alapján generált állapot-kifejezések

CTL* szemantika: Jelölések

- $M = (S, R, L)$ Kripke-struktúra
- $\pi = (s_0, s_1, s_2, \ldots)$ az M egy útvonala, ahol s_0 a kezdőállapot és $\forall i \geq 0: (s_i, s_{i+1}) \in R$
  - $\pi^i = (s_i, s_{i+1}, s_{i+2}, \ldots)$ a π útvonal szuffixe i-től

- $M, \pi \models p$ jelöli (ahol p útvonal-kifejezés): az M modellben a π útvonalon igaz p

- $M, s \models p$ jelöli (ahol p állapot-kifejezés): az M modellben az s állapotban igaz p

CTL* szemantika: Állapot-kifejezések

- **S1**:

  $M,s \models P$ a.cs.a. $P \in L(s)$

- **S2**:

  $M,s \models \neg p$ a.cs.a. $M,s \models p$ nem igaz

  $M,s \models p \wedge q$ a.cs.a. $M,s \models p$ és $M,s \models q$

- **S3**:

  $M,s \models E\ p$ (ahol p útvonal-kifejezés)

  a.cs.a. létezik $\pi=(s_0, s_1, s_2,\ldots)$ útvonal M-ben

  $s= s_0$ mellett, hogy $M,\pi \models p$.

  $M,s \models A\ p$ (ahol p útvonal-kifejezés)

  a.cs.a. minden $\pi=(s_0, s_1, s_2,\ldots)$ útvonalra M-ben

  ahol $s= s_0$ fennáll igaz, hogy $M,\pi \models p$.

CTL* szemantika: Útvonal-kifejezések

- **P1**:

  $M,\pi \models p$ (p állapot-kifejezés) a.cs.a. $M, s_0 \models p$

- **P2**:

  $M,\pi \models \neg p$ a.cs.a. $M,\pi \models p$ nem igaz

  $M,\pi \models p \wedge q$ a.cs.a. $M,\pi \models p$ és $M,\pi \models q$

- **P3**:

  $M, \pi \models X\, p$ a.cs.a. $M,\pi^1 \models p$

  $M, \pi \models p\, U\, q$ a.cs.a

$$\exists j \geq 0 : (M, \pi^j \models q \text{ valamint } \forall 0 \leq k < j : M, \pi^k \models p)$$

Háttér: Az ellenőrzés komplexitása

- Worst-case időkomplexitás: Legalább $O(|S|^2 \times 2^{|p|})$
  - $|S|^2$ az átmenetek száma a modellben (Kripke-struktúrában) worst case esetben
  - $|p|$ az operátorok száma a temporális logikai kifejezésben
- Az exponenciális komplexitás riasztó
  - Bár a temporális logikai kifejezések tipikusan rövidek
- Célkitűzés: CTL* egyszerűsítése
  - Gyakorlati problémák esetén használható maradjon
  - Az ellenőrzés worst-case időkomplexitása csökkenjen

CTL: Computational Tree Logic

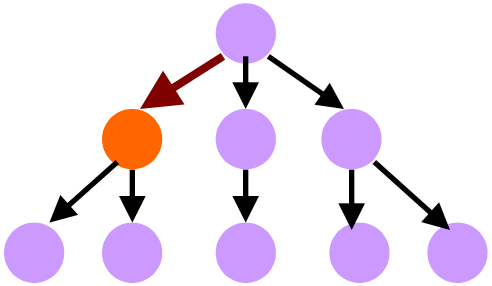
CTL operátorok (informális bevezető)

Állapotokon értelmezhető összetett operátorok:

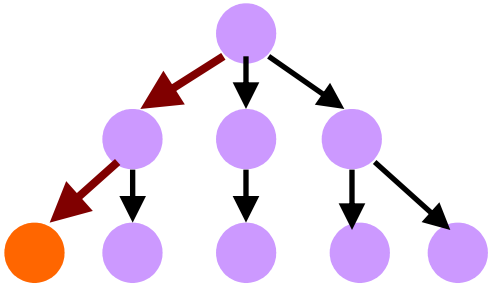
- EX p: létezik útvonal, aminek következő állapotán p
- EF p: létezik útvonal, aminek jövőbeli állapotán p
- EG p: létezik útvonal, aminek minden állapotán p
- E(p U q): létezik útvonal, amin p amíg q

- AX p: minden útvonal következő állapotán p
- AF p: minden útvonal egy-egy jövőbeli állapotán p
- AG p: minden útvonal minden állapotán p
- A(p U q): minden útvonalon p amíg q

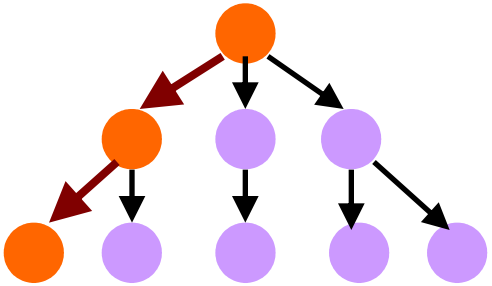
CTL operátorok (példák)



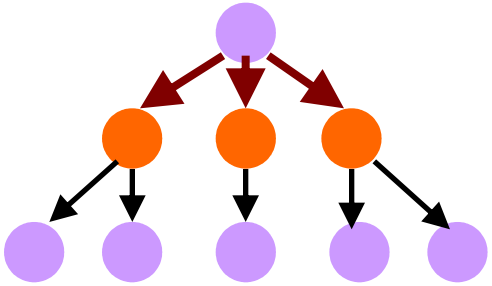
EX P



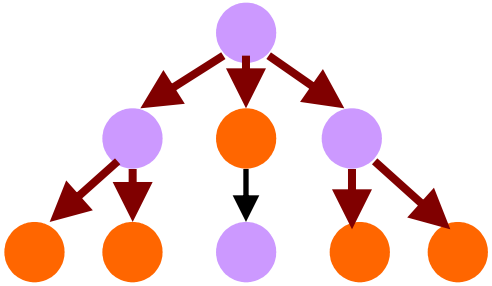
EF P



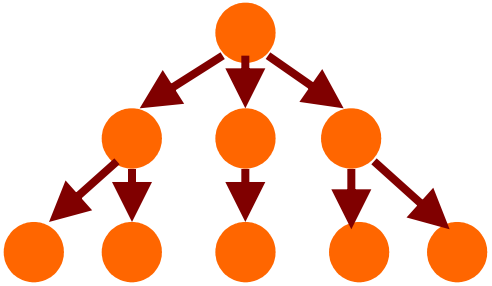
EG P



AX P



AF P



AG P

CTL kifejezések (példák)

- AG EF p

  Bárhonnan indulva olyan állapotba vihető a rendszer, ahol p igaz

  - Példa: AG EF Reset
- AG AF p

  Bárhonnan indulva mindenképpen eljutunk olyan állapotba, ahol p igaz

  - Példa: AG AF Terminated
- AG (p $\Rightarrow$ AF q)

  Bárhonnan indulva teljesül, hogy ha ott p igaz, akkor valamikor mindenképpen elérünk olyan állapotba, ahol q igaz.

  - Példa: AG (Request $\Rightarrow$ AF Reply)

CTL kifejezések (példák)

- EF AG p

  Lehetséges, hogy a rendszer olyan állapotba kerül, hogy utána p minden állapotban igaz lesz

- AF AG p

  Bármely úton haladva eljutunk olyan állapotba, hogy utána p mindig igaz lesz

  - Példa: AF AG Normal

- AG (p $\Rightarrow$ A (p U q))

  Bármelyik elérhető állapotban ha p igaz, akkor minden úton p fennáll q eléréséig

  - „p fennáll q eléréséig” pontosabban: elérünk egy olyan állapotba, ahol q igaz, és addig minden állapotban p igaz

Követelmények formalizálása: Egy példa

- Két processzből álló rendszer: P1 és P2
- Processz állapotok a követelmények szempontjából:
  - Kritikus szakaszban van: C1, C2
  - Nem-kritikus szakaszban van: N1, N2
  - Kritikus szakaszba belépésre kész: W1, W2
- Atomi kijelentések:
  AP = {C1, C2, N1, N2, W1, W2}

Mintapélda (folytatás)

- Egyszerre csak egy processz lehet a kritikus szakaszban:
  $AG\ (\neg(C1 \wedge C2))$
- Ha egy processz be akar lépni a kritikus szakaszba, akkor előbb-utóbb mindig beléphet:
  $AG\ (W1 \Rightarrow AF(C1))$
  $AG\ (W2 \Rightarrow AF(C2))$
- A processzek mindig felváltva kerülnek a kritikus szakaszba; egyikük kilép majd a másik lép be:
  $AG(C1 \Rightarrow A(C1\ U\ (\neg C1 \wedge A((\neg C1)\ U\ C2))))$
  $AG(C2 \Rightarrow A(C2\ U\ (\neg C2 \wedge A((\neg C2)\ U\ C1))))$

Mintapélda (folytatás)

- Egyszerre csak egy processz lehet a kritikus szakaszban:
  $AG\ (\neg(C1 \wedge C2$
- Ha egy process[illegible]ni a kritikus szakaszba, akkor előbb-utóbb [illegible] belé[illegible]
  [illegible]C1))
  [illegible]C2))
- A process[illegible]k mindig f[illegible]váltva kerülnek a kritikus szakaszba; egyikük kilép majd a másik lép be:
  $AG(C1 \Rightarrow A(C1\ U\ (\neg C1 \wedge A((\neg C1)\ U\ C2))))$
  $AG(C2 \Rightarrow A(C2\ U\ (\neg C2 \wedge A((\neg C2)\ U\ C1))))$

P1 nincs a kritikus szakaszban

P1 van a kritikus szakaszban

P2 lép a kritikus szakaszba

CTL formális szintaxis I.

Állapot-kifejezések:

- CTL* esetén volt:
  - **S1**: Minden P atomi kijelentés egy állapot-kifejezés
  - **S2**: Ha p és q állapot-kifejezések, $\neg p$ és $p \wedge q$ is
  - **S3**: Ha p útvonal-kifejezés, akkor E p és A p állapot-kifejezések.

- CTL esetén ezek (**S1**, **S2**, **S3**) változatlanok!

CTL formális szintaxis II.

Útvonal-kifejezések:

- CTL* esetén volt:
  - **P1**: Minden állapot-kifejezés útvonal-kifejezés
  - **P2**: Ha p és q útvonal-kifejezések, akkor $\neg$p és p$\wedge$q is
  - **P3**: Ha p és q útvonal-kifejezések, akkor X p és p U q is

- CTL esetén ezek helyett egy szabály lesz:
  - **P0**: Ha p és q állapot-kifejezések,
    akkor X p és p U q útvonal-kifejezések.

Összefoglalva: CTL formális szintaxis

Állapot-kifejezések:

- **S1**: Minden P atomi kijelentés egy állapot-kifejezés
- **S2**: Ha p és q állapot-kifejezések, $\neg p$ és $p \wedge q$ is
- **S3**: Ha p útvonal-kifejezés, akkor E p és A p állapot-kifejezések.

Útvonal-kifejezések:

- **P0**: Ha p és q állapot-kifejezések, akkor X p és p U q útvonal-kifejezések.

- Útvonal-kifejezések nem kombinálhatók
- Útvonal-kifejezéseket csak az **S3** szabály használja
- X p és p U q útvonal-kifejezések elé csak valamelyik útvonal kvantor kerülhet

CTL formális szintaxis következményei

- Útvonal-kifejezések nem kombinálhatók:
  - X és U csak állapot-kifejezéseken alkalmazhatók
  - Boole operátorok csak állapot-kifejezéseken alkalmazhatók
- Útvonal-kifejezéseket csak az **S3** szabály használja:
  - Ld. **S3**: Ha p útvonal-kifejezés, akkor E p és A p állapot-kifejezések.
- Az **S3** szabály miatt X p és p U q útvonal-kifejezések elé csak valamelyik útvonal kvantor kerülhet (mást nem tehetünk útvonal-kifejezésekkel): ezért „összenőnek” az operátorok
  - EX, E(. U .),
  - AX, A(. U .)

CTL és CTL* kifejezések

- „Kimaradt” CTL operátorok
  - EF p jelentése E (true U p)
  - AF p jelentése A (true U p)
  - EG p jelentése $\neg$AF ($\neg$p)
  - AG p jelentése $\neg$EF ($\neg$p)

- CTL* de nem CTL
  - E(X Piros $\vee$ F Sárga)

    Boole operátor van útvonal-kifejezések között
  - A(X G (Piros $\wedge$ Sárga)),
    E(XXX Piros)

    Egymásba ágyazott útvonal-kifejezések vannak

CTL formális szemantika

- Állapot-kifejezések:
  - **S1**, **S2**, **S3** szabályok (lásd CTL*) változatlanok
- Útvonal-kifejezések:
  - **P1**, **P2**, **P3** helyébe egy új **P0** szabály lép:

P0:

- $M,\pi \models X\ p$ ahol p állapot-kifejezés a.cs.a. $M,s_1 \models p$
- $M,\pi \models p\ U\ q$ ahol p,q állapot-kifejezés a.cs.a.

$$\exists j \geq 0 : (M, s_j \models q \text{ valamint } \forall 0 \leq k < j : M, s_k \models p)$$

Itt állapot-kifejezések vannak a szintaxis szabály szerint!

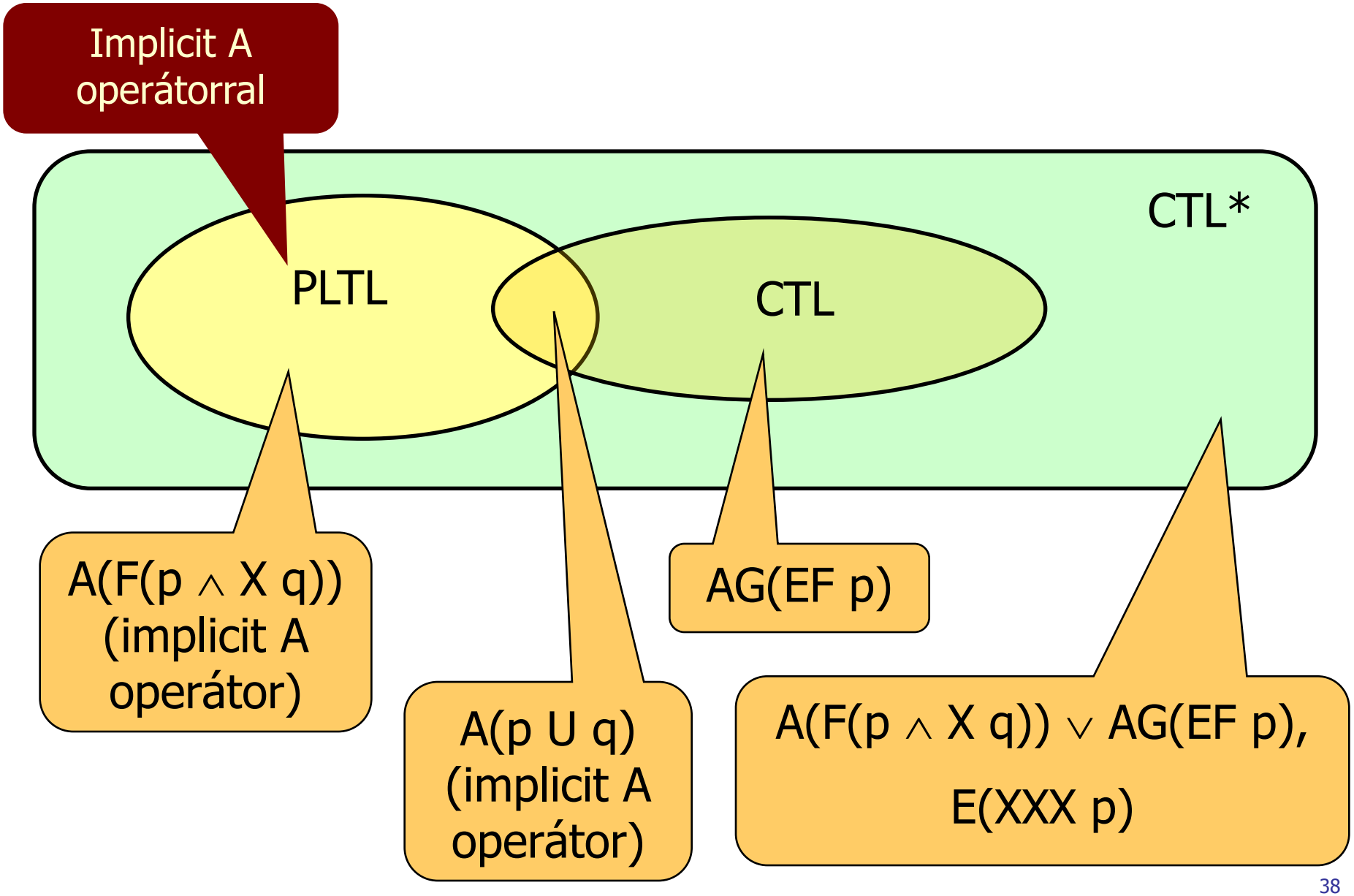
Háttér: Az ellenőrzés komplexitása

- CTL* worst-case időkomplexitás: $O(|S|^2 \times 2^{|p|})$
- CTL worst case időkomplexitás: $O(|S|^2 \times |p|)$
  - $|S|^2$ az átmenetek száma a modellben (Kripke-struktúra) worst case esetben
  - $|p|$ az operátorok száma a temporális logikai kifejezésben
- CTL*-nál kedvezőbb a CTL esetén:
  - Itt nincs $2^{|p|}$ tag
  - Sok gyakorlati követelmény esetén jól használható
    - Biztonsági követelmények: AG
    - Élőségi követelmények: EF, AF
- Mi ennek az ára?

Kifejezőképesség

- Egy temporális logika kifejezőképessége nagyobb egy másikénál, ha képes olyan tulajdonság megfogalmazására, amire a másik nem.
- Eddigi tapasztalatok:
  - Lineáris idejű logika nem tudja figyelembe venni a lehetséges elágazásokat
    (implicit „minden útra” jellegű vizsgálat lehetséges)
  - CTL kötöttebb, mint a CTL*,
    ezért kevesebb tulajdonság megfogalmazására képes
  - CTL* magába foglalja a lehetséges PLTL kifejezéseket

LTL, CTL, CTL* kifejezőképessége



Kifejezőképesség - formálisan

- TL2 kifejezőképessége nagyobb vagy egyenlő (nem kisebb) mint TL1 kifejezőképessége,
  ha minden M modellre és annak minden s állapotára:

$$\forall p \in TL1:$$

$$\exists q \in TL2:\ (M, s \models p \ \Leftrightarrow\ M, s \models q)$$

- Ha ez kölcsönösen fennáll, akkor TL2 és TL1 azonos kifejezőképességűek.

Kitekintés

Sztochasztikus logikák:

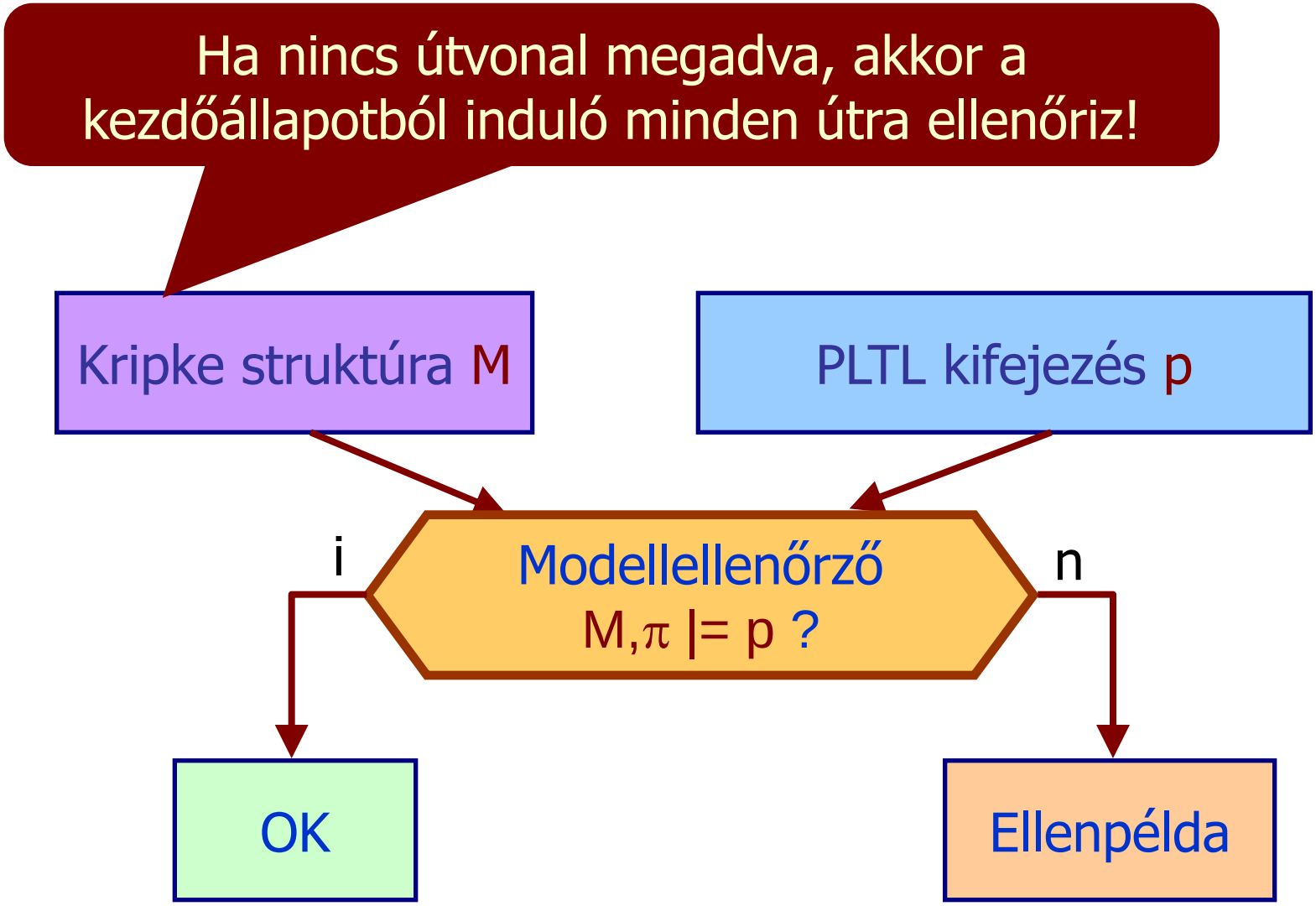
- Megbízhatósági illetve idő követelményekhez használható
  - Pl.: Ha HIBA az aktuális állapot, akkor ez kisebb mint 30% valószínűséggel áll fenn 2 időegység múlva is
- CTL kiterjesztése:
  - Folytonos idejű Markov-láncokon értelmezett (nem Kripke-struktúra)
  - Valószínűségi kritérumok állapotok eléréséhez (állandósult állapot), útvonalak bejárásához
  - Idő kritériumok (időintervallumok) X és U operátorokhoz

Valósidejű logikák:

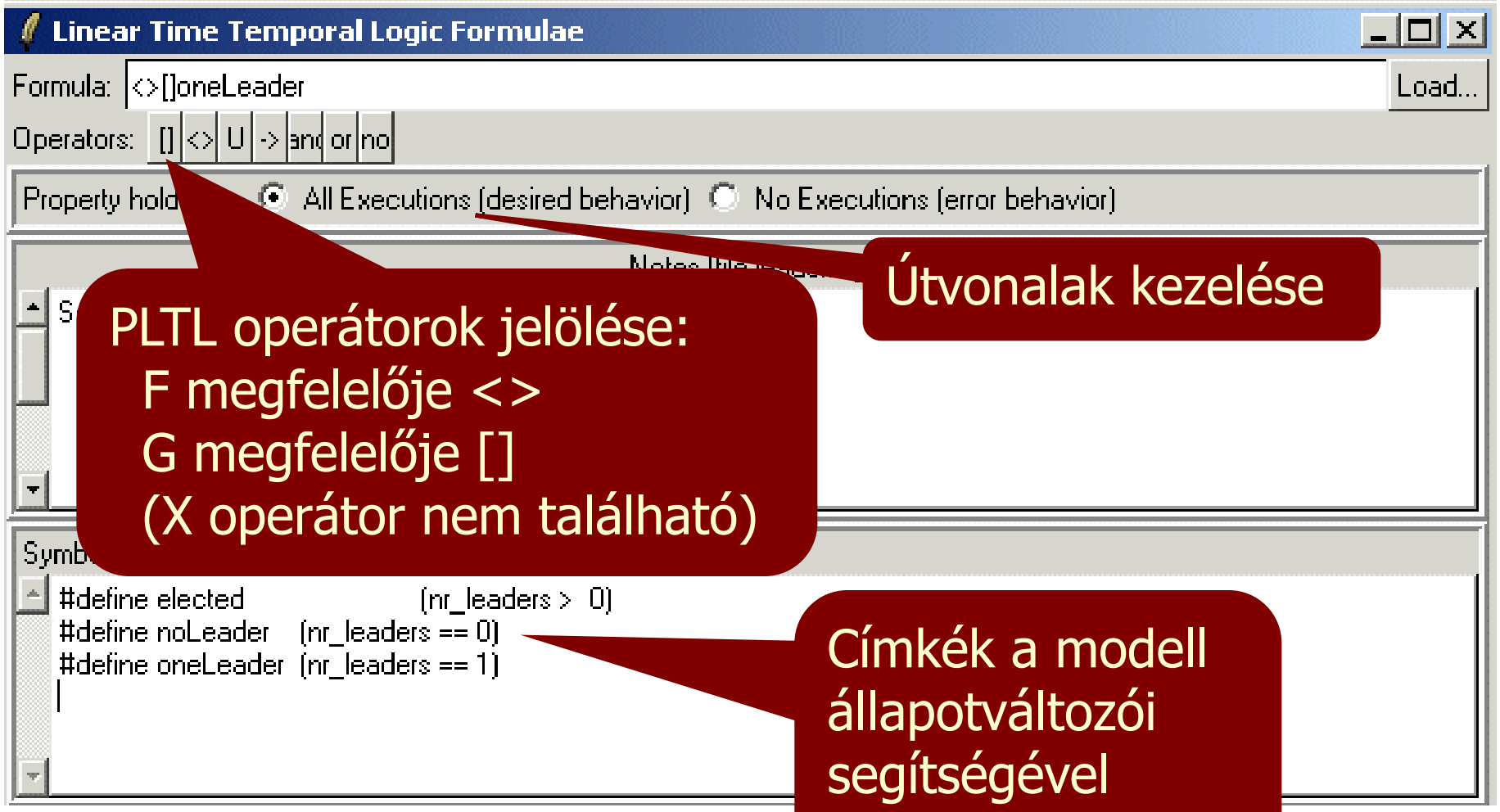
- Valósidejű rendszerek követelményeinek megadásához
  - Óraváltozókra hivatkozhat a logika
  - Időintervallumok kezelése kidolgozott

A modellellenőrzési feladat

PLTL modellellenőrzés

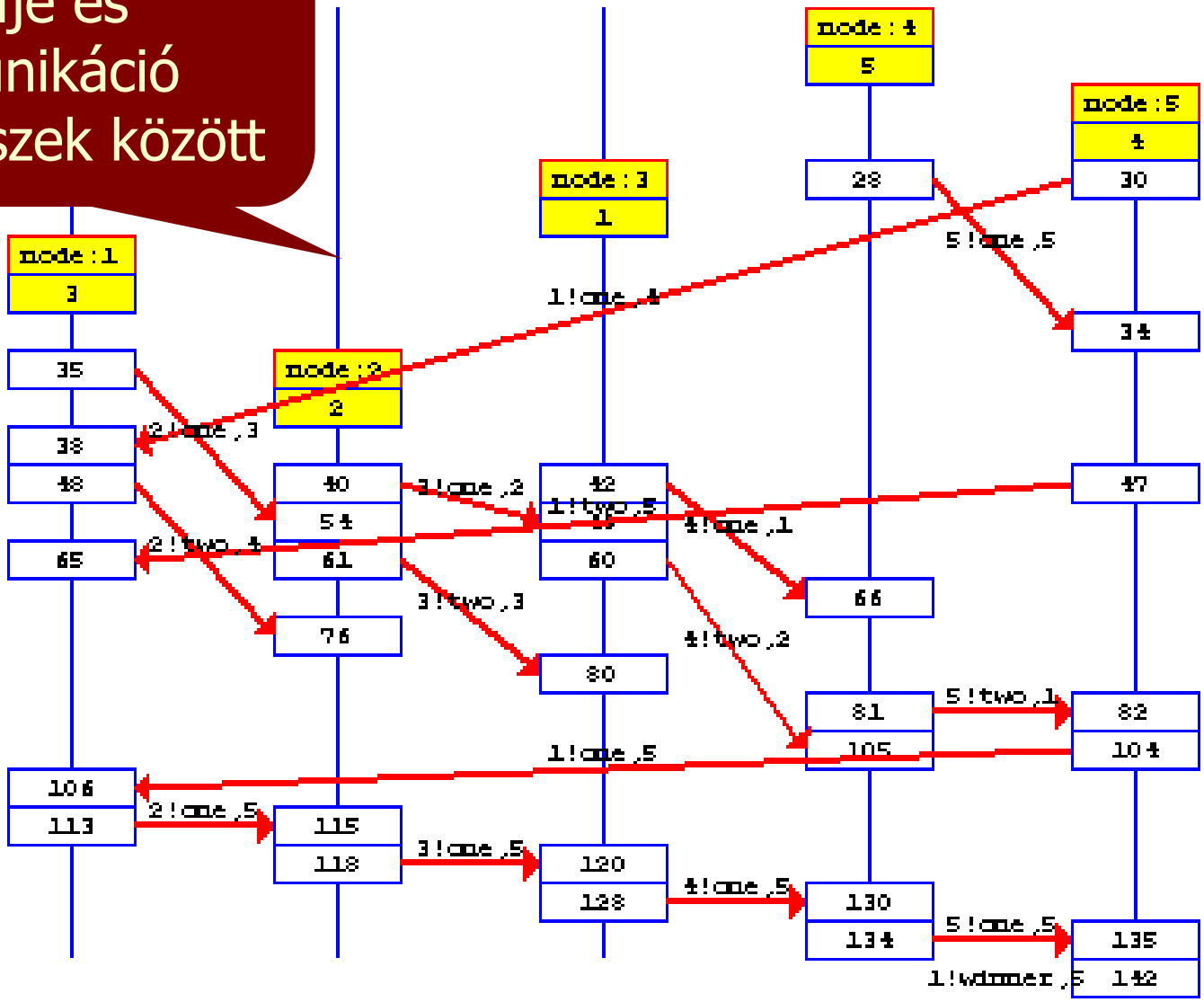


A SPIN modellellenőrző (régi felület)

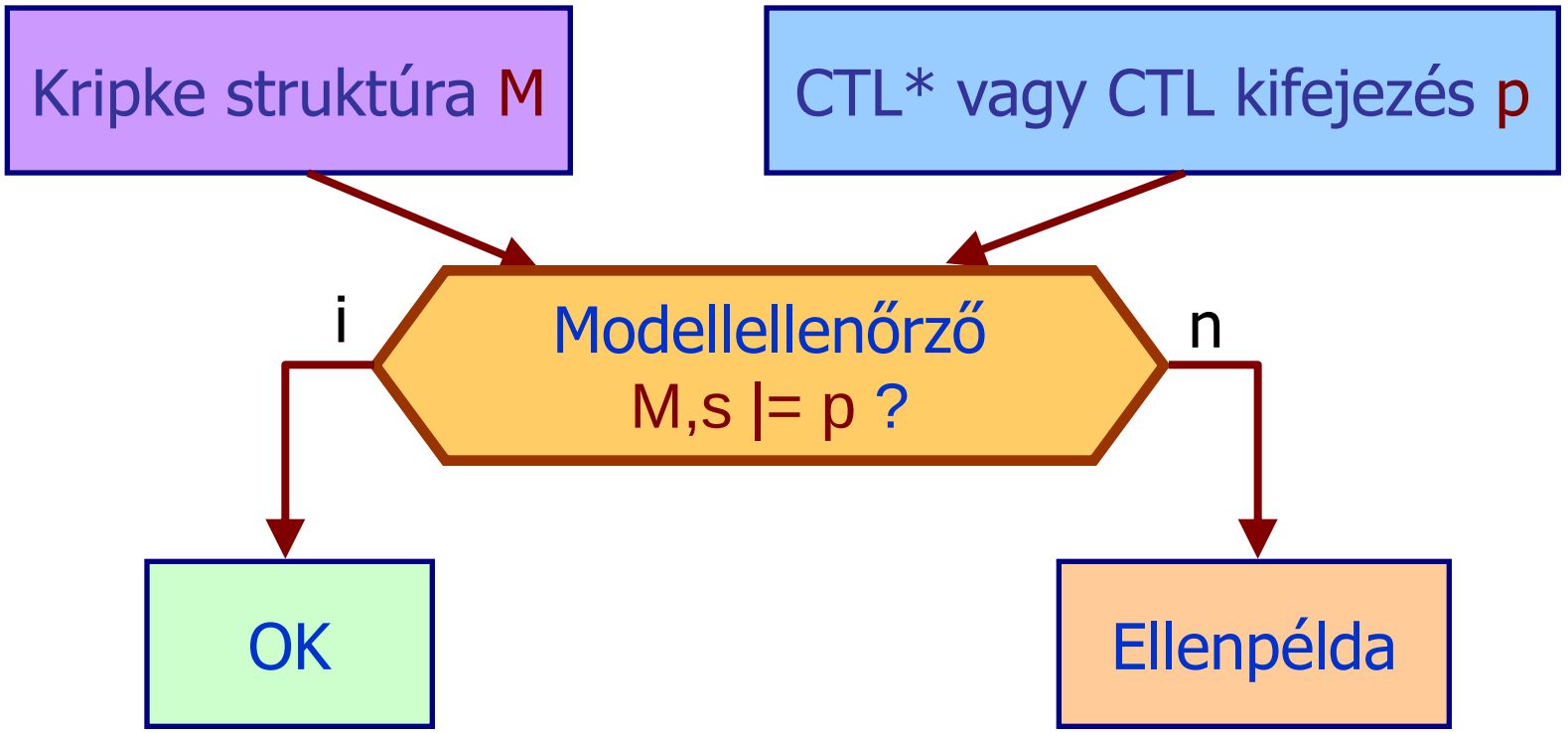


SPIN ellenpélda

Itt: Állapotok sorrendje és kommunikáció processzek között



CTL* vagy CTL modellellenőrzés



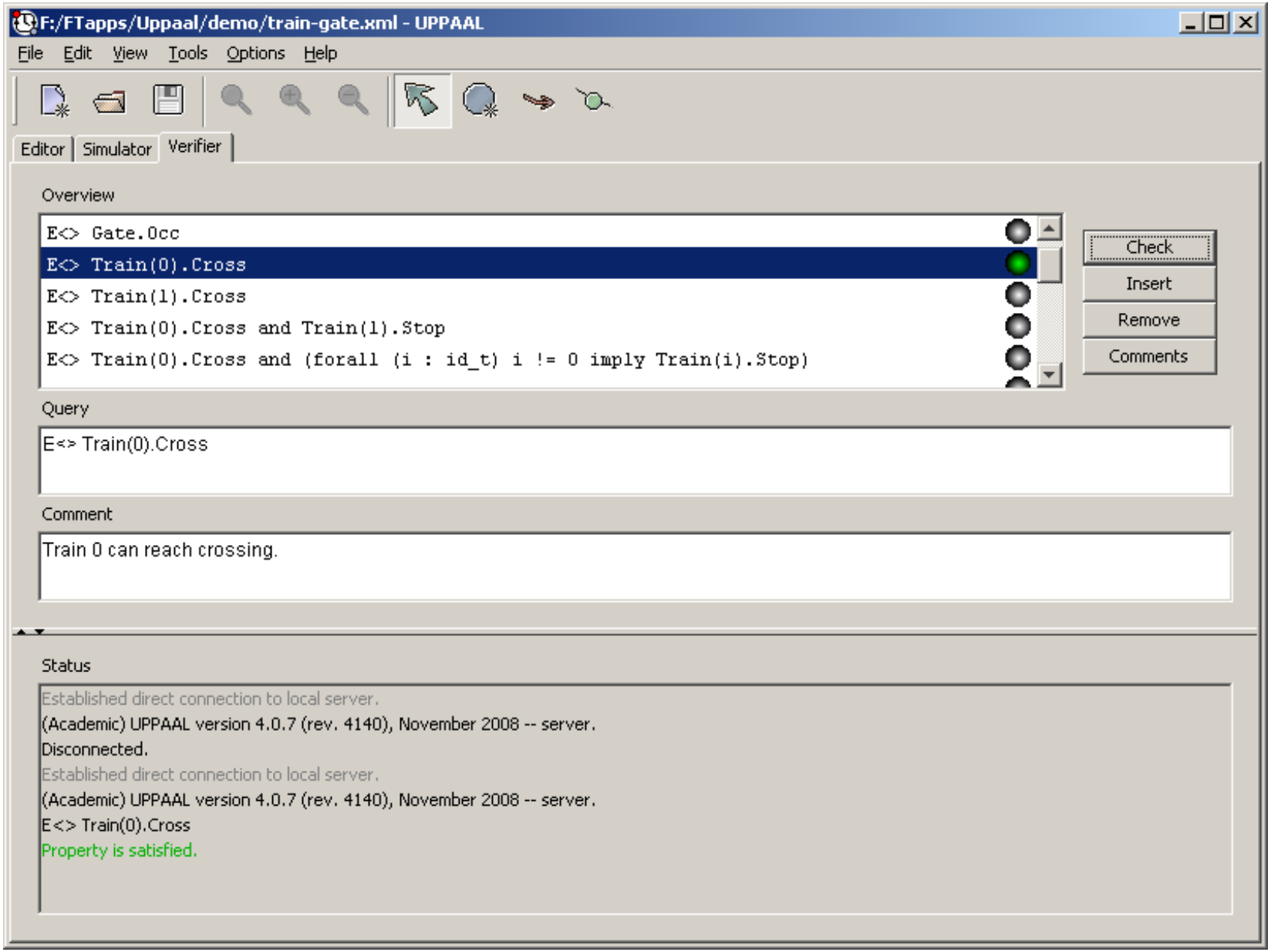
Az UPPAAL modellellenőrző lehetőségei

- Atomi kijelentések:
  - Változók értéke hivatkozható: pl. a!=1
    - Egész aritmetika és bitenkénti műveletek használhatók
  - Állapot hivatkozható: pl. Train(0).cross
    - Paraméterezett processzekre: forall, exists operátorok
  - Holtpont: deadlock kijelentéssel megadható (nincs akció)
- Boole logikai operátorok:
  - and, or, imply, not, ? : (ez utóbbi az if-then-else)
- Temporális operátorok: Korlátozott CTL
  - Jelölés: [] szerepel G helyett, <> szerepel F helyett
    - Így lesz: A[], A<>, E[], E<>
    - E[] esetén véges útvonalon is értelmezett (végállapotig)
  - Egymásba nem ágyazhatók temporális operátorok
    - De egy lehetőség: p-->q rövidítés jelentése A[] (p imply A<> q)

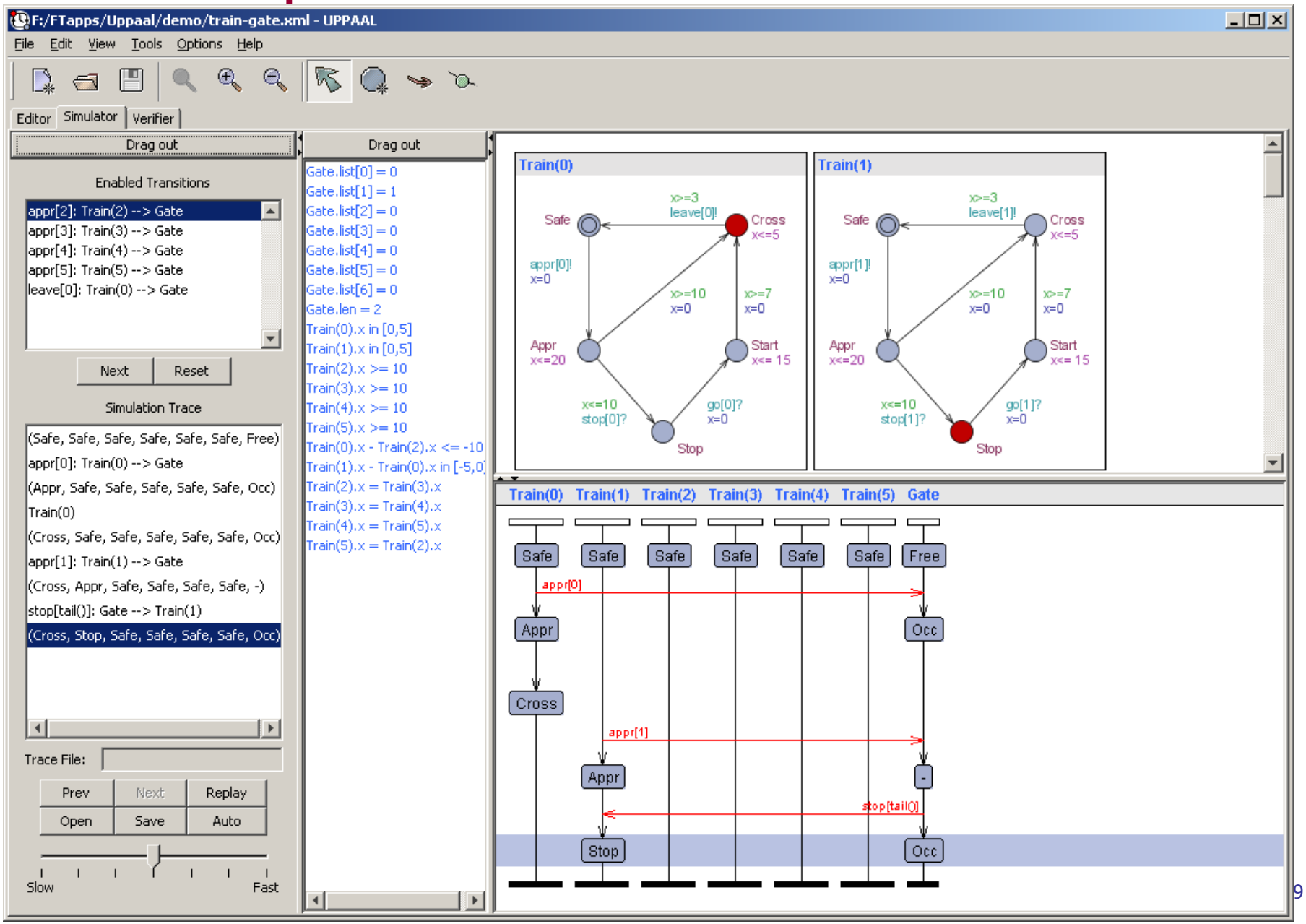
Követelmények ellenőrzése az UPPAAL-ban

- Követelmények halmaza szerkeszthető
- Modell ellenőrzés egyenként is indítható
- Ellenpélda generálható:
  - Legrövidebb, leggyorsabb, vagy akármi
  - Betölthető a szimulátorba (végigjátszható)
- Keresés az állapottérben:
  - Mélységi
  - Szélességi
- Állapottárolás különféle opciókkal:
  - Redukció
  - Közelítő állapottér tárolás (alul- illetve felülbecslés)
  - Hash tábla mérete megadható

Az UPPAAL modellellenőrző ablaka



Ellenpélda az UPPAAL szimulátorban



A motivációs példa befejezése

Motivációs mintapélda: Kölcsönös kizárás

- 2 résztvevőre, 3 megosztott változóval (H. Hyman, 1966)
  - **blocked0**: Első résztvevő (P0) be akar lépni
  - **blocked1**: Második résztvevő (P1) be akar lépni
  - **turn**: Ki következik belépni (0 esetén P0, 1 esetén P1)

P0

```
while (true) {
    blocked0 = true;
    while (turn!=0) {
        while (blocked1==true) {
            skip;
        }
        turn=0;
    }
    // Critical section
    blocked0 = false;
    // Do other things
}
```

P1

```
while (true) {
    blocked1 = true;
    while (turn!=1) {
        while (blocked0==true) {
            skip;
        }
        turn=1;
    }
    // Critical section
    blocked1 = false;
    // Do other things
}
```

Helyes-e ez az algoritmus?

A modell UPPAAL-ban (első változat)

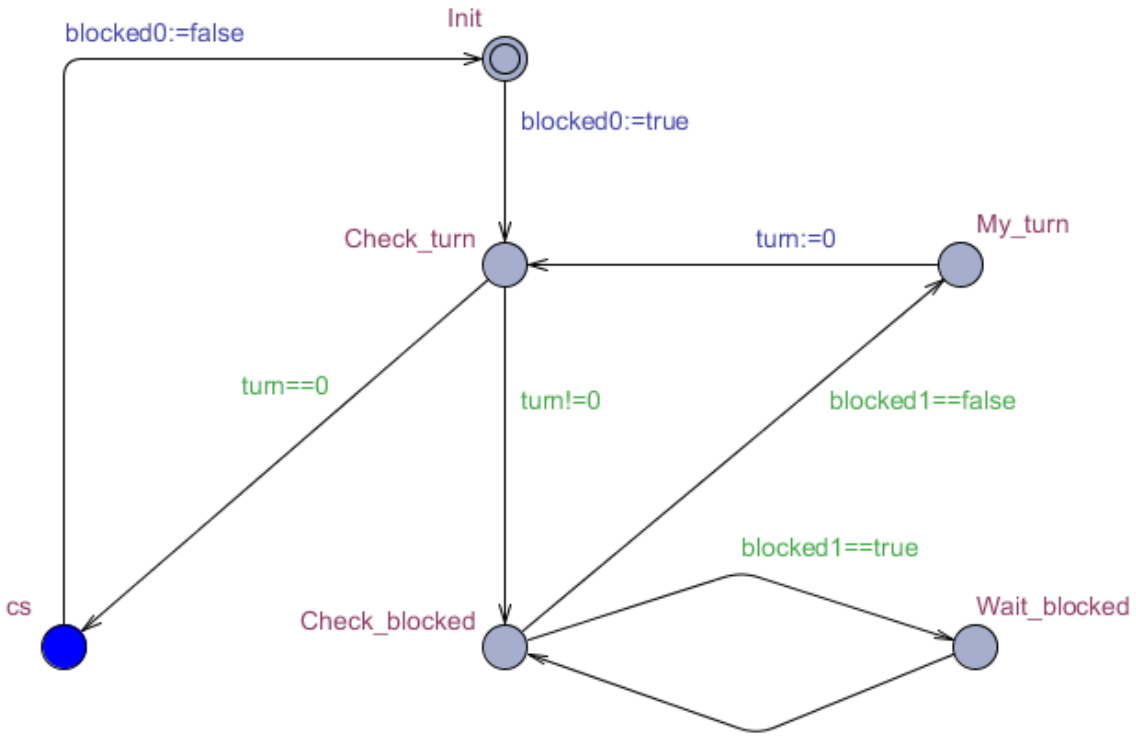
Deklarációk:

```
bool blocked0;
bool blocked1;
int[0,1] turn=0;
system P0, P1;
```

Kihasznált modellezési lehetőségek:

- Közös változók rendszerszintű deklarálása
- Korlátozott értékkészletű változók

A P0 automata:



P0

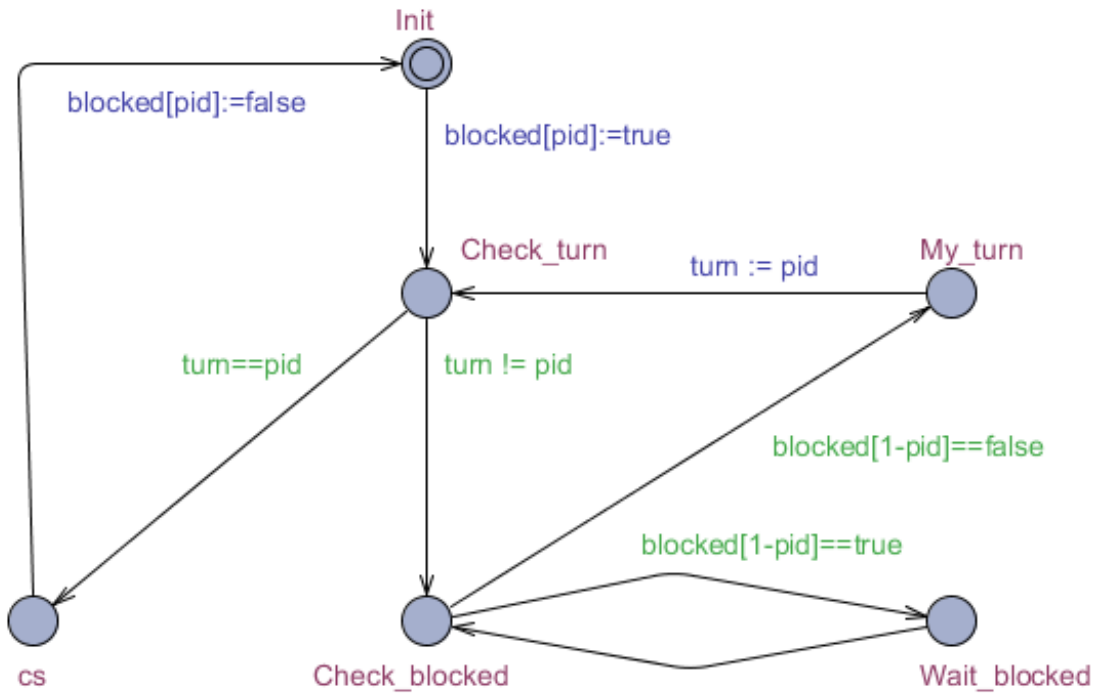
```
while (true) {
    blocked0 = true;
    while (turn!=0) {
        while (blocked1==true) {
            skip;
        }
        turn=0;
    }
    // Critical section
    blocked0 = false;
    // Do other things
}
```

A modell UPPAAL-ban (második változat)

Deklarációk:

```
bool blocked[2];
int[0,1] turn;
P0 = P(0);
P1 = P(1);
system P0,P1;
```

A P automata template pid paraméterrel:



Kihasznált modellezési lehetőségek:

- Közös változók rendszerszintű deklarálása
- Korlátozott értékkészletű változók
- Azonos viselkedésű résztvevők azonos automata template alapján
- Példányosítás paraméterezéssel
- Változó tömbök (résztvevőkhöz)

P0

```
while (true) {
    blocked0 = true;
    while (turn!=0) {
        while (blocked1==true) {
            skip;
        }
        turn=0;
    }
    // Critical section
    blocked0 = false;
    // Do other things
}
```

UPPAAL: A követelmények formalizálása

- Kölcsönös kizárás:
  - Egyszerre csak az egyik résztvevő lehet a kritikus szakaszban: A[] not (P0.cs and P1.cs)
- Holtpontmentesség:
  - Nem alakul ki leállás (amikor nincs továbblépés): A[] not deadlock
- Lehetséges az elvárt viselkedés:
  - P0 egyáltalán beléphet a kritikus szakaszba: E<>(P0.cs)
  - P1 egyáltalán beléphet a kritikus szakaszba: E<>(P1.cs)
- Nincs kiéheztetés:
  - P0 mindenképpen be fog lépni a kritikus szakaszba: A<>(P0.cs)
  - P1 mindenképpen be fog lépni a kritikus szakaszba: A<>(P1.cs)

UPPAAL: A modellellenőrzés eredménye

- A kölcsönös kizárás nem teljesül!
  - Ellenpélda: Átlapolódás a két résztvevő között (végigjátszható a szimulátorban)
- Nincs holtpont
- Lehetséges az elvárt viselkedés
- Kiéheztetés elkerülése időzítések megadása nélkül nem vizsgálható
  - Triviális ellenpélda: A kiinduló állapotban marad
    - Ez az időfüggő viselkedés modellezésének „specialitása”
    - Kilépés: Urgent állapot vagy állapot invariáns megadása esetén
  - Ezután is lehet kiéheztetés?
    - Van rá példa (ciklikus működés)!
    - Az algoritmus önmagában nem garantálja a kiéheztetés elkerülését

Az algoritmus javítása

Peterson algoritmusa

- P0 résztvevőre (P1 értelemszerű):

Hyman:

```
while (true) {
    blocked0 = true;
    while (turn!=0) {
        while (blocked1==true) {
            skip;
        }
        turn=0;
    }
    // Critical section
    blocked0 = false;
    // Do other things
}
```

Peterson:

```
while (true) {
    blocked0 = true;
    turn=1;
    while (blocked1==true &&
        turn!=0) {
            skip;
    }

    // Critical section
    blocked0 = false;
    // Do other things
}
```

Összefoglalás

- Lineáris idejű temporális logikák:
  - PLTL
- Elágazó idejű temporális logikák:
  - CTL*
  - CTL (kötöttebb, de egyszerűbben ellenőrizhető)
- Formális szintaxis és szemantika
- Modellellenőrzési feladat
  - Megoldás algoritmusa: Következő előadás!