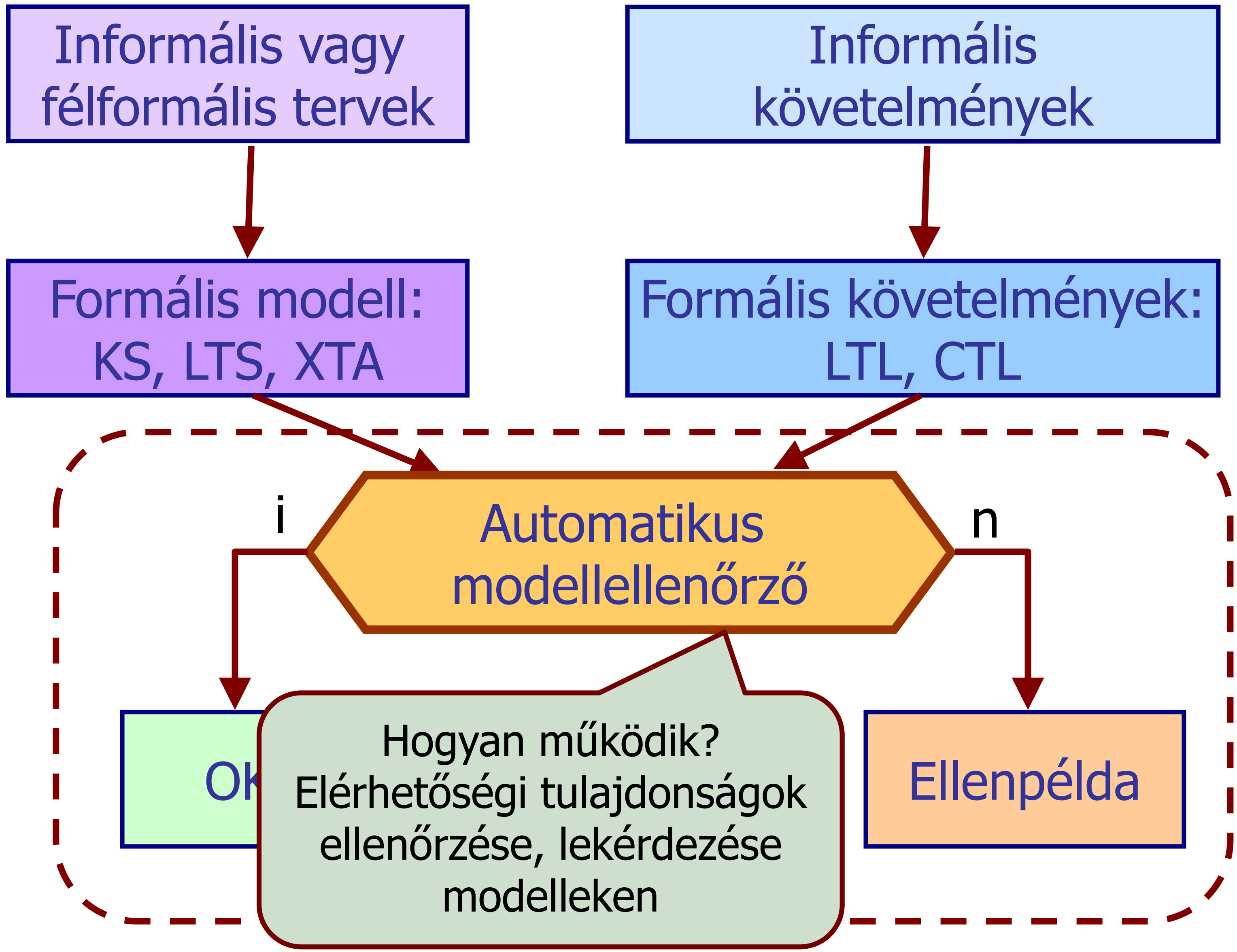
# Modellellenőrzés

dr. Majzik István
BME Méréstechnika és Információs Rendszerek Tanszék

# Mit szeretnénk elérni?



# Az előadás áttekintése

## Hogyan működik a modellellenőrzés?

- A modellellenőrzés technikái
  - LTL modellellenőrzés: Tabló módszer
  - CTL modellellenőrzés: Szemantika alapú módszer

## Miért jó ezt tudni?

- Lehetőségek felmérése
  - Korlátok becslése (pl. ellenőrizhető modellek mérete)
  - Hatékony megvalósítás ($10^{69.000}$ állapot? – következő ea.)
- Érdekes alkalmazások alapja
  - Automatikus teszteset-generálás
  - Lépéssorozat generálás szintézis feladathoz

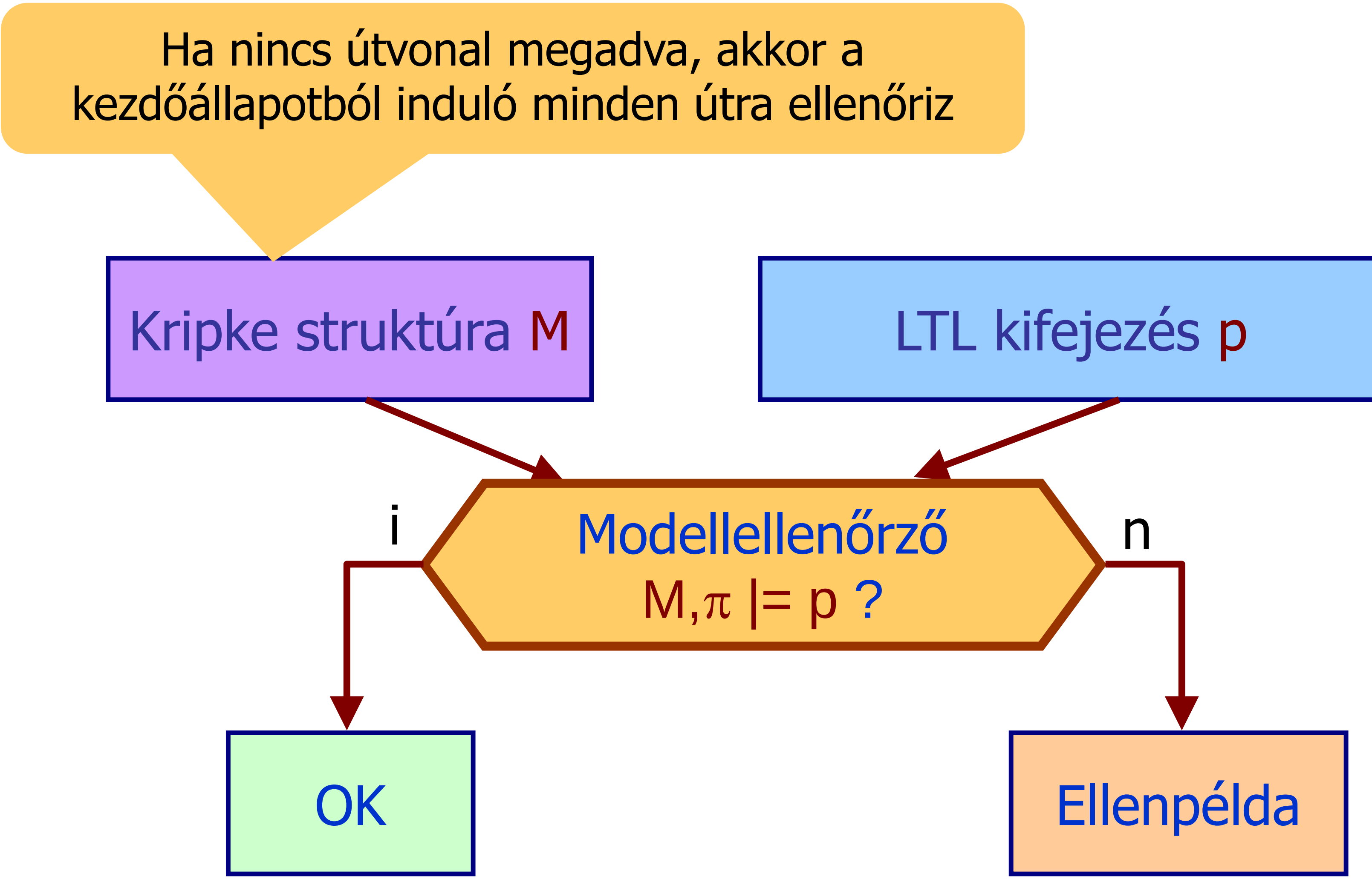
# Modellellenőrzők

| Name | Model Checking | | | Equivalence checking | | GUI | | | Availability | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| | Plain, Probabilistic, Stochastic, ... | Modelling language | Properties language | Supported equivalences | Counter example generation | GUI | Graphical Specification | Counter example visualization | Software license | Programming language used | Platform / OS |
| BLAST | Code analysis | C | Monitor automata | | Yes | No | No | No | Free | OCaml | Windows and Unix related |
| CADP | Plain and probabilistic | LOTOS, FC2, FSP, LNT | AFMC, MCL, XTL | SB, WB, BB, OE, STE, WTE, SE, tau*E | Yes | Yes | No | Yes | FUSC | C, Bourne shell, Tcl/Tk, LOTOS, LNT | Mac OS, Linux, Solaris, Windows |
| CPAchecker | Code analysis | C | Monitor automata | | Yes | Yes | No | Yes | Free | Java | Any |
| DREAM | Real-time | C++, Timed automata | Monitor automata | | Yes | No | No | No | Free | C++ | Windows and Unix related |
| Java Pathfinder | Plain and timed | Java | unknown | | No | Yes | No | No | Open Source Agreement | Java | Mac OS, Windows, Linux |
| LTSmin | Plain, Real-time | Promela, μCRL, mCRL2, DVE Input Language, UPPAAL timed automata, Event-B, CSP, TLA+, Z, Petri net | μ-calculus, LTL, CTL* | SB, BB | Yes | No | No | No | Free | C, C++ | Unix, Mac OS X, Windows |
| mCRL2 | Plain, Real-time | mCRL2 | μ-calculus | SB, BB, t*E, STE, WTE | Yes | Yes | No | Yes | Free | C++ | Mac OS, Linux, Solaris, Windows |
| MRMC | Real-time, Probabilistic | Plain MC | CSL, CSRL, PCTL, PRCTL | SB | No | No | No | No | Free | C | Windows, Linux, Mac OS |
| Murφ (Murphi) | Plain | Murφ | Invariants, Assertions | | Yes | No | No | No | Free | C++ | Linux |
| NuSMV | Plain | SMV input language | CTL, LTL, PSL | | Yes | No | No | No | Free | C | Unix, Windows, Mac OS X |
| PAT | Plain,Real-time,Probabilistic | CSP#, Timed CSP, Probabilistic CSP | LTL, Assertions | | Yes | Yes | Yes | Yes | Free | C# | Windows, other OS with Mono |
| PRISM | Probabilistic | PEPA, PRISM language, Plain MC | CSL, PLTL, PCTL | | No | Yes | No | No | Free | C++, Java | Windows, Linux, Mac OS |
| SPIN | Plain | Promela | LTL | | Yes | Yes | No | Yes | FUSC | C, C++ | Windows and Unix related |
| TAPAAL | Real-time | Timed-Arc Petri Nets, age invariants, inhibitor arcs, transport arcs | TCTL subset | | No | Yes | Yes | Yes | Free | C++, Java | Mac OS, Windows, Linux |

https://en.wikipedia.org/wiki/List_of_model_checking_tools

# LTL modellellenőrzés tabló módszerrel

# A modellellenőrzés feladata

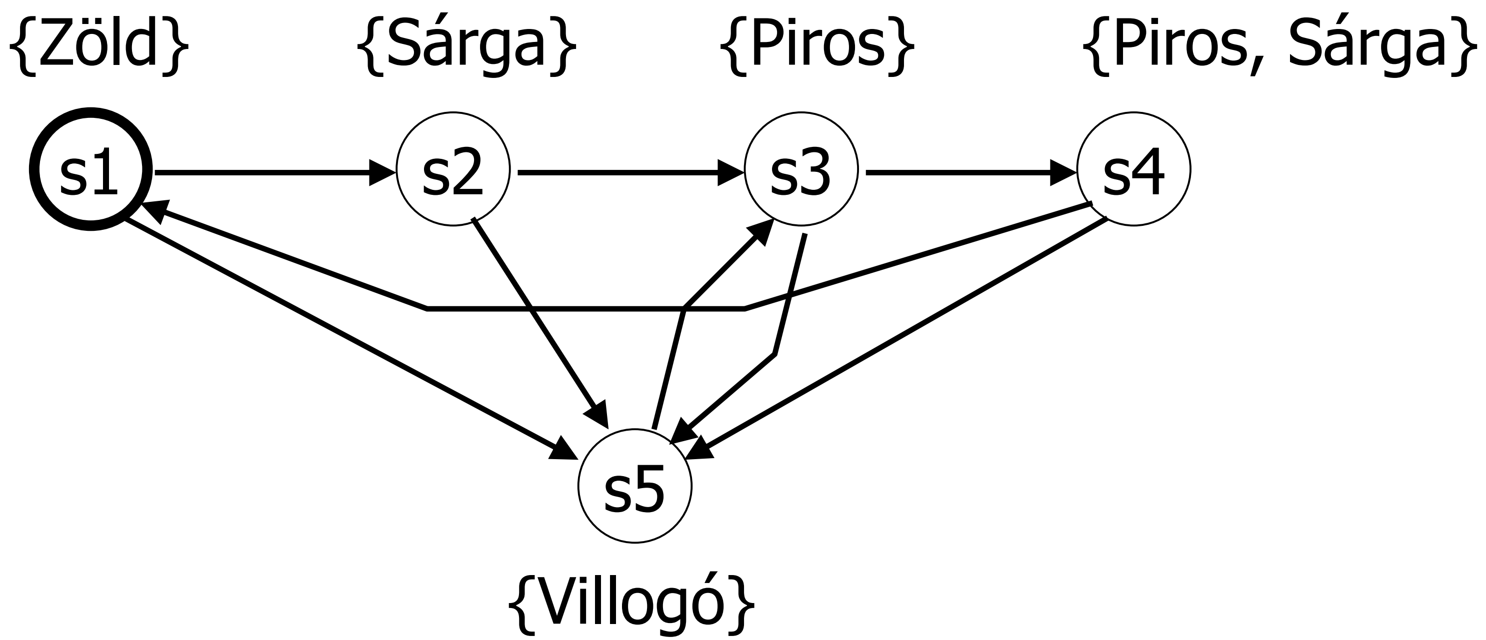


# Ismétlés: Kripke-struktúra

M = (S, R, L)

Példa: Közlekedési lámpa vezérlője

AP = {Zöld, Sárga, Piros, Villogó}



# Ismétlés: Lineáris idejű temporális logika: LTL

## LTL elemei:

- Atomi kijelentések (AP elemei): P, Q, ...
- Boole logikai operátorok: $\wedge$, $\vee$, $\neg$, $\Rightarrow$

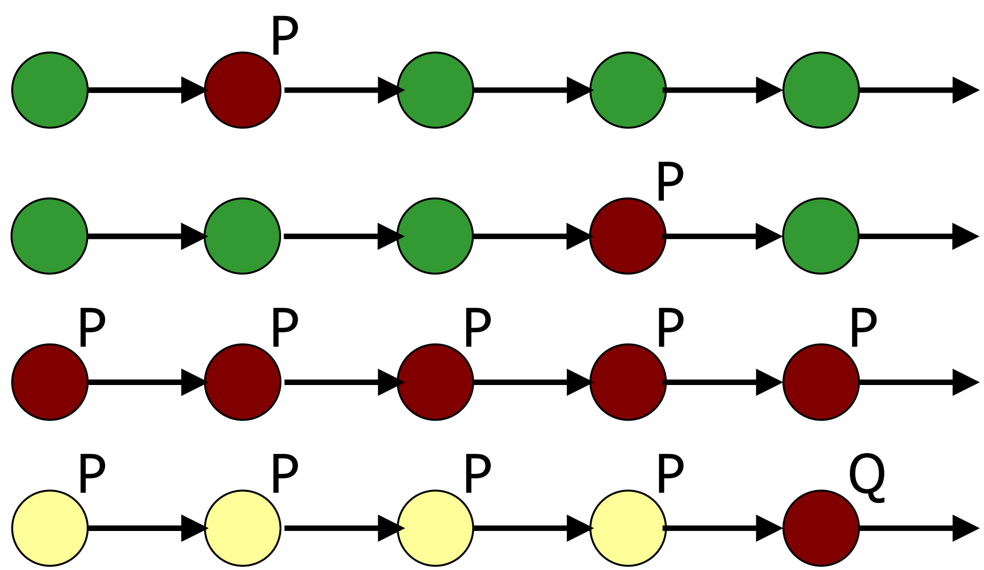
  $\wedge$: És, $\vee$: Vagy, $\neg$: Negálás , $\Rightarrow$: Implikáció
- Temporális operátorok: X, F, G, U:

X P — neXt

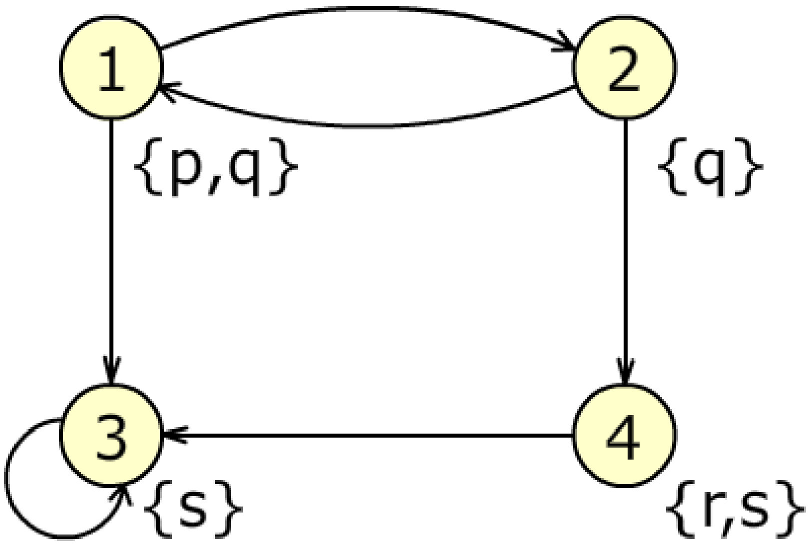
F P — Future

G P — Globally

P U Q — Until



# LTL temporális logika használata




Igaz-e az 1 állapotból induló minden útvonalra:

- X s

  Nem igaz, 2 esetén nincs s
- p U q

  Igaz, mert lokálisan q igaz
- F r

  Nem igaz, mert lehetségesek r elérése nélküli útvonalak

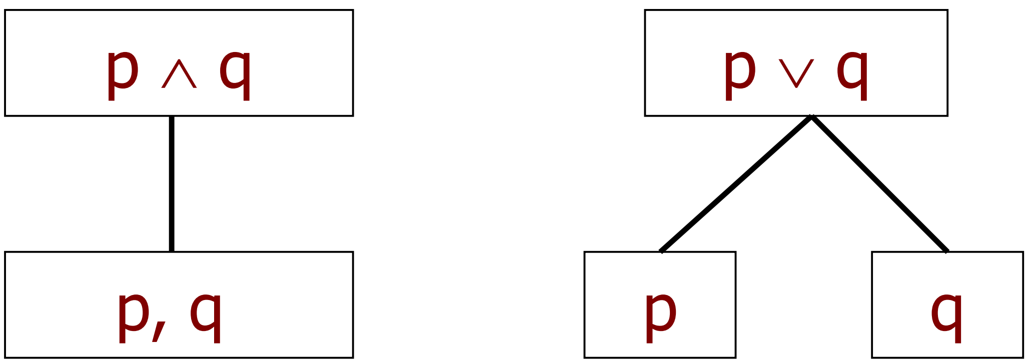
Szisztematikus módszer: Ellenpélda keresés

- Van-e olyan útvonal, ahol a tulajdonság nem áll fenn?
- Teljesíthető-e a negált tulajdonság?

# Bevezető: Tabló módszer a Boole logika esetén

Kérdés: Hogyan teljesíthető (tehető igazzá) egy adott Boole kifejezés?

- Alapötlet: A logikai kifejezés felbontása egy fa struktúrában (ez a tabló)
  - Csomópontok: Kifejezések, amelyeket igazzá akarunk tenni
    - ÉS operátor: Részkifejezések listába gyűjtése, ha mindegyik igaz kell legyen
  - Élek képzése: Lehetőségek a teljesítésre az operátorok jelentése alapján
    - VAGY operátor: Elágazás a fában (többféleképpen igazzá tehető)
- Kifejezés felbontási szabályok Boole logika esetén:




- A felbontás előtt a kifejezést ún. negált normál formára kell hozni: Negálás ne legyen összetett kifejezések előtt, csak változók előtt
  - de Morgan azonosságok: $\neg(p \vee q) = (\neg p) \wedge (\neg q)$, $\neg(p \wedge q) = (\neg p) \vee (\neg q)$

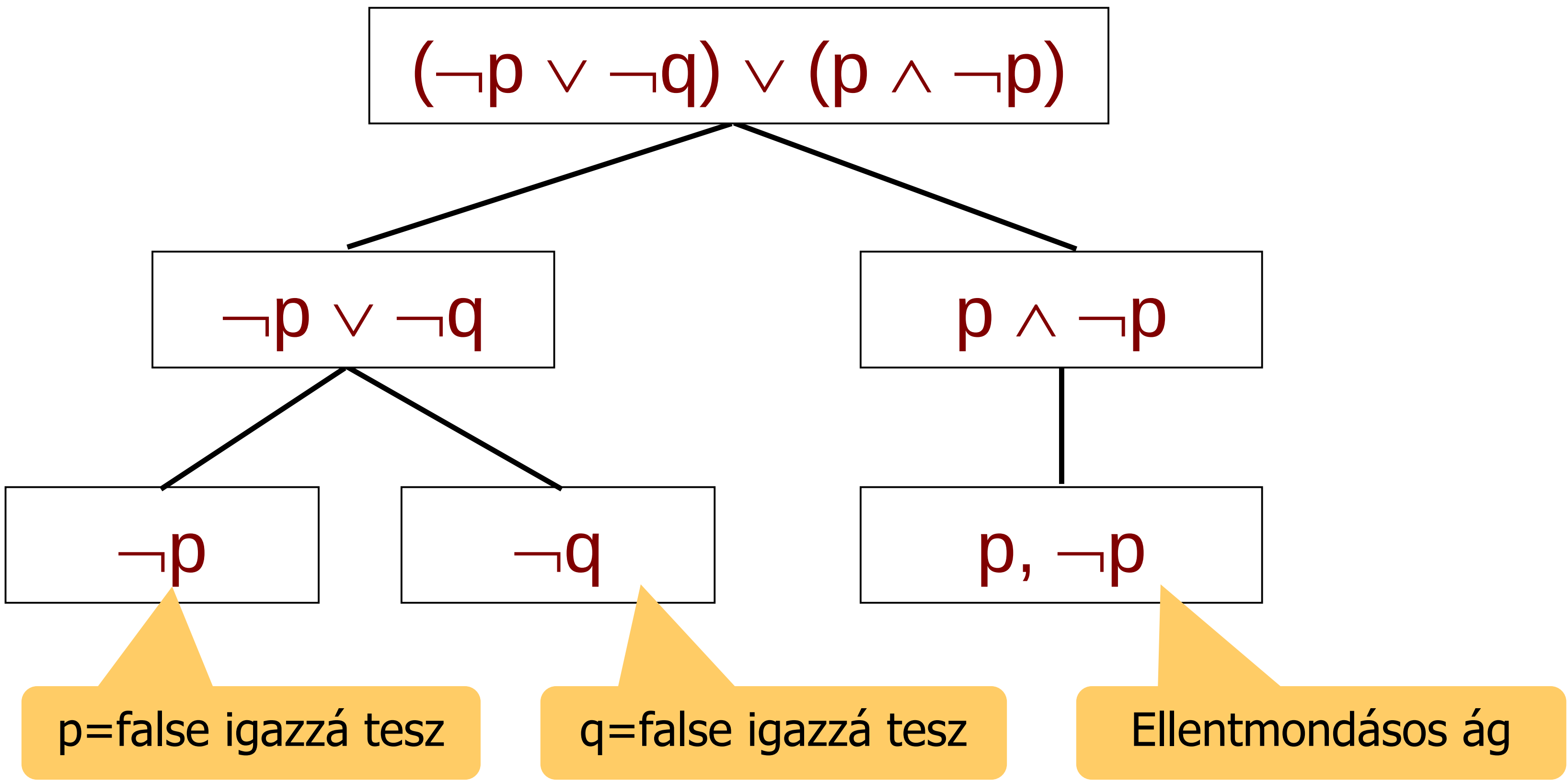
# Bevezető: Tabló kiértékelése Boole logika esetén

Meddig folytassuk a felbontást?

- Egy ág (felbontás) terminálása:
  - Operátor nem maradt, csak ponált vagy negált változók listája
  - A lista minden elemét igazzá kell tenni a változók behelyettesítésével (a ponált igaz, a negált hamis értéket kap)
- Egy-egy ág terminálása után:
  - Ellentmondásos ág: Ugyanaz a változó ponált és negált formában is előfordul a csomópontban; nincs konzisztens behelyettesítés
    - Pl. $p$, $\neg p$ lista ellentmondás, egyszerre nem lehet igaz a két kifejezés
  - Sikeres ág: Nincs ellentmondás; a csomópontban a lista minden eleme igazzá tehető behelyettesítéssel
    - Pl: $p$, $\neg q$ lista: $p$ igaz, $q$ hamis az a behelyettesítés, ami igazzá teszi
    - Az így adódó behelyettesítéssel a kezdeti kifejezés igazzá tehető
- A fa sikeres ágai alapján igazzá tehető a kifejezés

# Bevezető: Tabló konstruálása Boole logika esetén

- Eredeti kifejezés: $\neg(p \wedge q) \vee \neg(\neg p \vee p)$
- Negálás bevitele: $(\neg p \vee \neg q) \vee \ (p \wedge \neg p)$
- Tabló konstruálás:

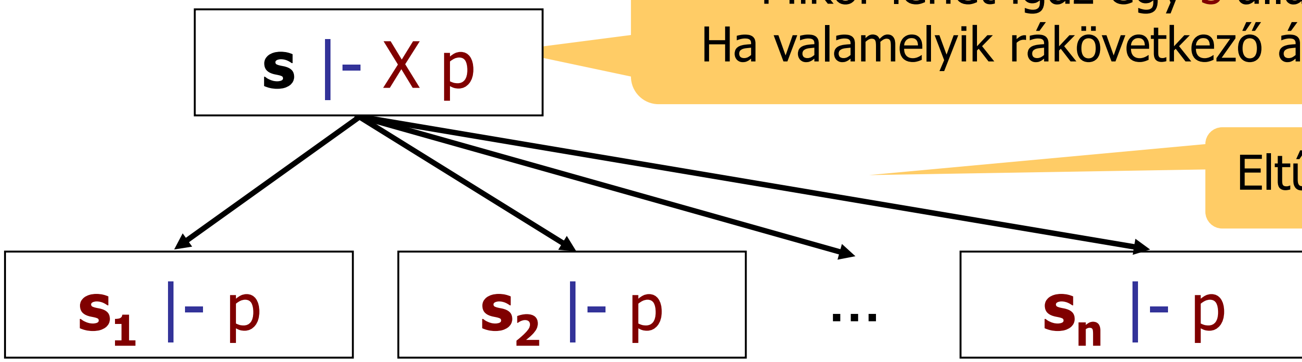


# A tabló kiterjesztése LTL-re

- Modellellenőrzés: Ellenpéldát keresünk adott kifejezéshez, tehát az eredeti LTL kifejezés negáltjából készül a tabló!
  - A negált kifejezésből kell negált normál formát képezni
  - Ha van sikeres (nem ellentmondásos) ág, az ellenpéldát ad
  - Ha minden ág ellentmondásos, akkor az eredeti kifejezés igaz
- Új felbontási szabályok kellenek a temporális operátorokhoz
  - Újdonság: A felbontás a modell alapján végezhető (állapotokon)
  - Jelölés: s |- p jelöli, hogy p teljesítését keressük s állapotból indulva
- Atomi kijelentések kezelése:
  - s |- P sikeres, ha $P \in L(s)$
  - s |- P ellentmondásos, ha $P \notin L(s)$
  - s |- $\neg$P sikeres, ha $P \notin L(s)$
- Temporális operátorok:
  - X és U felbontása elég (a többiek ezekkel kifejezhetők, ld. szintaxis)

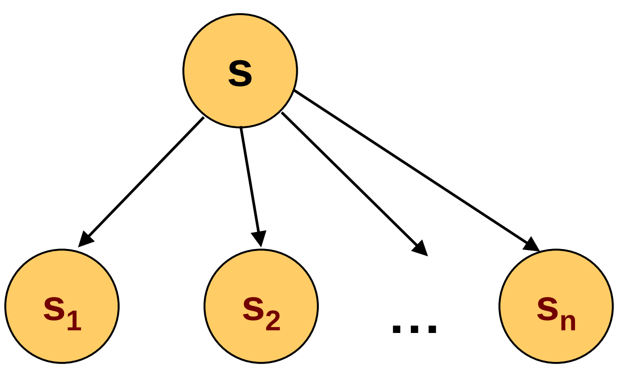
# Felbontás az X operátor esetén

Mikor lehet igaz egy s állapotban X p ?
Ha valamelyik rákövetkező állapotban igaz p




Eltűnik az X operátor

## amennyiben a modellben:




### Ellentmondásra jutunk:

- Az s-nek nincs rákövetkező állapota
- Atomi kijelentés p mint lokális tulajdonság nem teljesül

### Sikeres ágak:

- Atomi kijelentés p mint lokális tulajdonság teljesül

# Felbontás az U operátor esetén

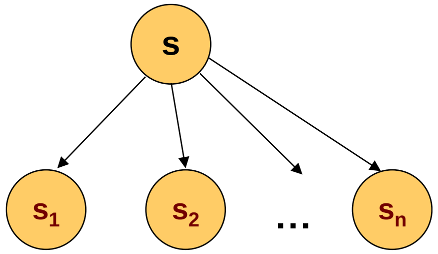
- Felhasználjuk: $p \, U \, q = \underbrace{q}_{} \vee \underbrace{(p \wedge X(p \, U \, q))}_{}$

Külön figyelni kell: véges útvonal, végtelen p útvonal

$s \models p \, U \, q$

$s \models q$ | $s \models p, \quad s \models X(p \, U \, q)$

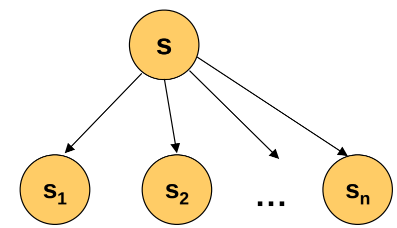
$s \models p, \quad s_1 \models p \, U \, q$ … $s \models p, \quad s_n \models p \, U \, q$

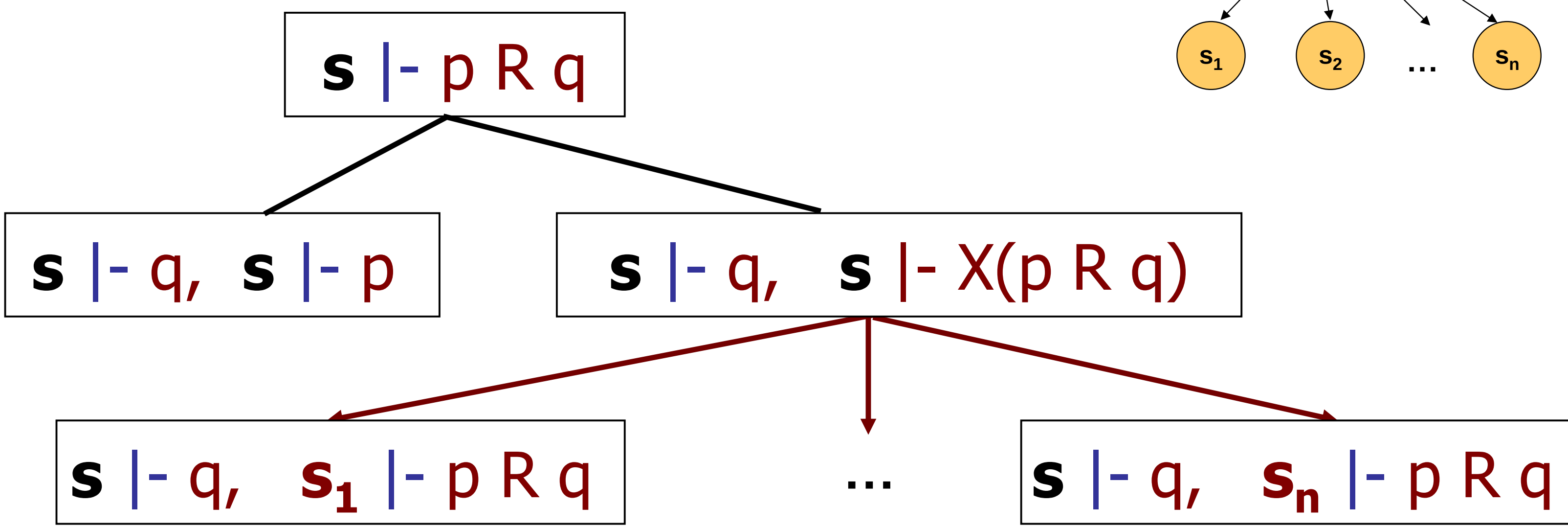



- A felbontás meddig folytatódik?
  - Ellentmondásra jutunk:
    - Atomi kijelentés mint lokális tulajdonság nem teljesül
    - X operátor van, de az útvonal véget ér q teljesülése nélkül
    - Ciklus alakul ki p teljesülésével, de q teljesülése nélkül
  - Sikeres ágak:
    - Atomi kijelentések (lista) mint lokális tulajdonságok teljesülnek
    - Ciklus alakul ki, amiben nincs ellentmondás

# Egy speciális operátor: R

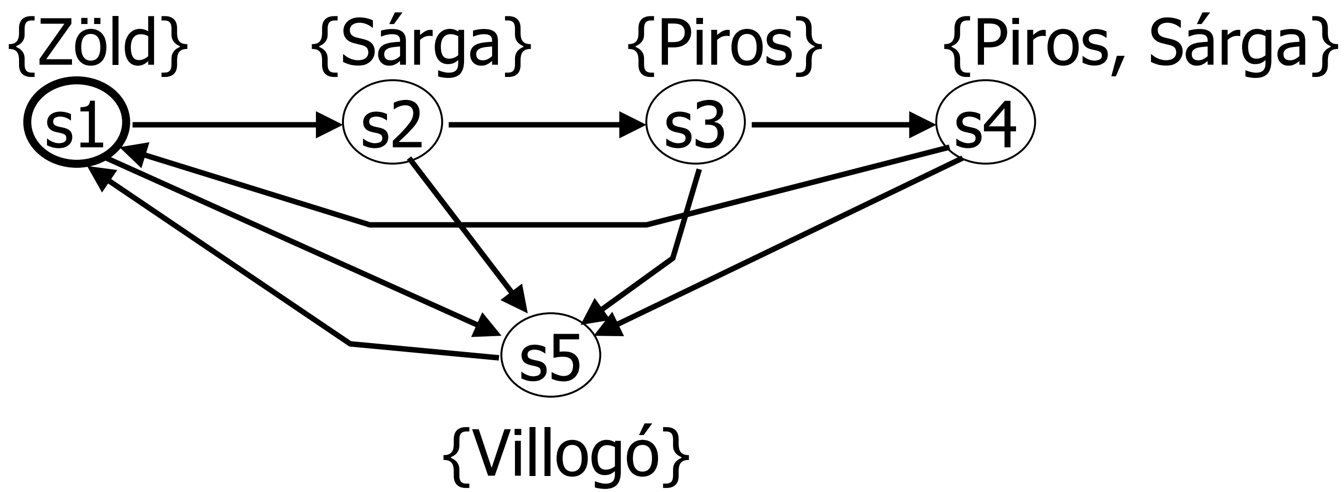
- Negált normál formára hozás az U operátor esetén:
  $\neg(p\ U\ q) = ?$
  - Bevezethető az U operátor „duálisa”, az R (Release)
    $$\neg(p\ U\ q) = (\neg p)\ R\ (\neg q)$$
  - Felírható: $p\ R\ q = q \wedge (p \vee X\ (p\ R\ q))$
- Az R operátor tablója:



# Egy példa: A modell és a LTL kifejezés

- A közlekedési lámpa vezérlő egy modellje (KS)
- Igaz-e, hogy ha a lámpa a kezdeti állapotban Zöld, akkor előbb-utóbb Piros lesz?
  - Az ellenőrizendő LTL kifejezés: Zöld $\Rightarrow$ F Piros

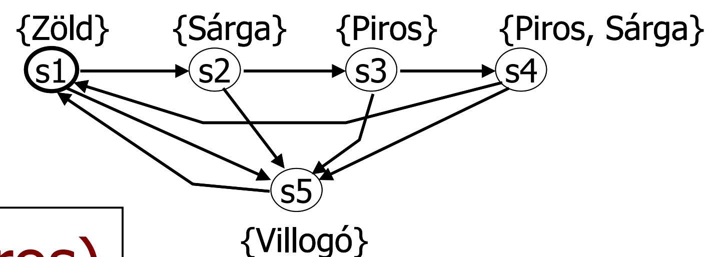



A modell alapján „kézzel” tudunk-e ellenpéldát adni?

# Egy példa: A kifejezés tablója (1)

- A kifejezés negálása: s1 |– ¬(Zöld ⇒ F Piros)
- Negált normál forma (P ⇒ Q = ¬P ∨ Q alapján):

  ¬(Zöld ⇒ F Piros) = Zöld ∧ ¬F Piros = Zöld ∧ G (¬Piros)
- A tabló konstruálása:




s1 állapotban van Zöld címke, OK

s1 |- Zöld ∧ G(¬Piros)

s1 |- Zöld, s1 |- G(¬Piros)

Egyszerűsített jelölés (s1 |- Zöld teljesült, kimaradhat a listából)

s1 |- G(¬Piros)

Folytatódik a következő dián!

# Egy példa: A kifejezés tablója (2)

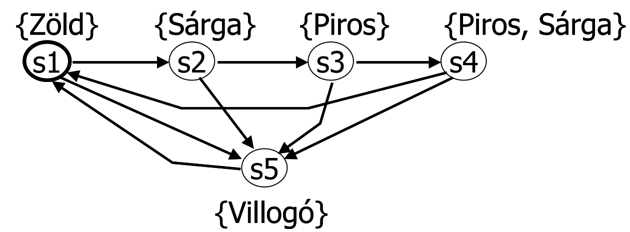
s1 állapotban nincs Piros

s1 |- G(¬Piros)

s1 |- ¬Piros, XG(¬Piros)

s1 állapot után s2 vagy s5 jöhet

s1 |- XG(¬Piros)




s2 állapotban nincs Piros

s2 |- G(¬Piros)

s2 |- ¬Piros, XG(¬Piros)

s2 után s3 vagy s5

s2 |- XG(¬Piros)

s5 |- G(¬Piros)

s5 |- ¬Piros, XG(¬Piros)

s5 |- XG(¬Piros)

Ellent-mondásos ág, s3-ban van Piros

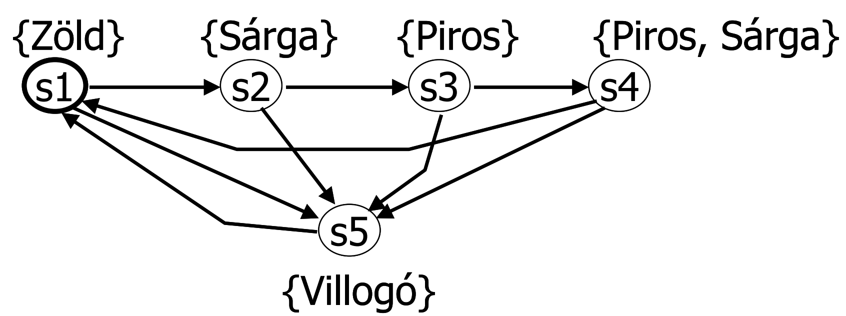
s3 |- G(¬Piros)

s3 |- ¬Piros, XG(¬Piros)

Ellentmondás nélküli ciklusok:
s1, s5, …
s1, s2, s5, …

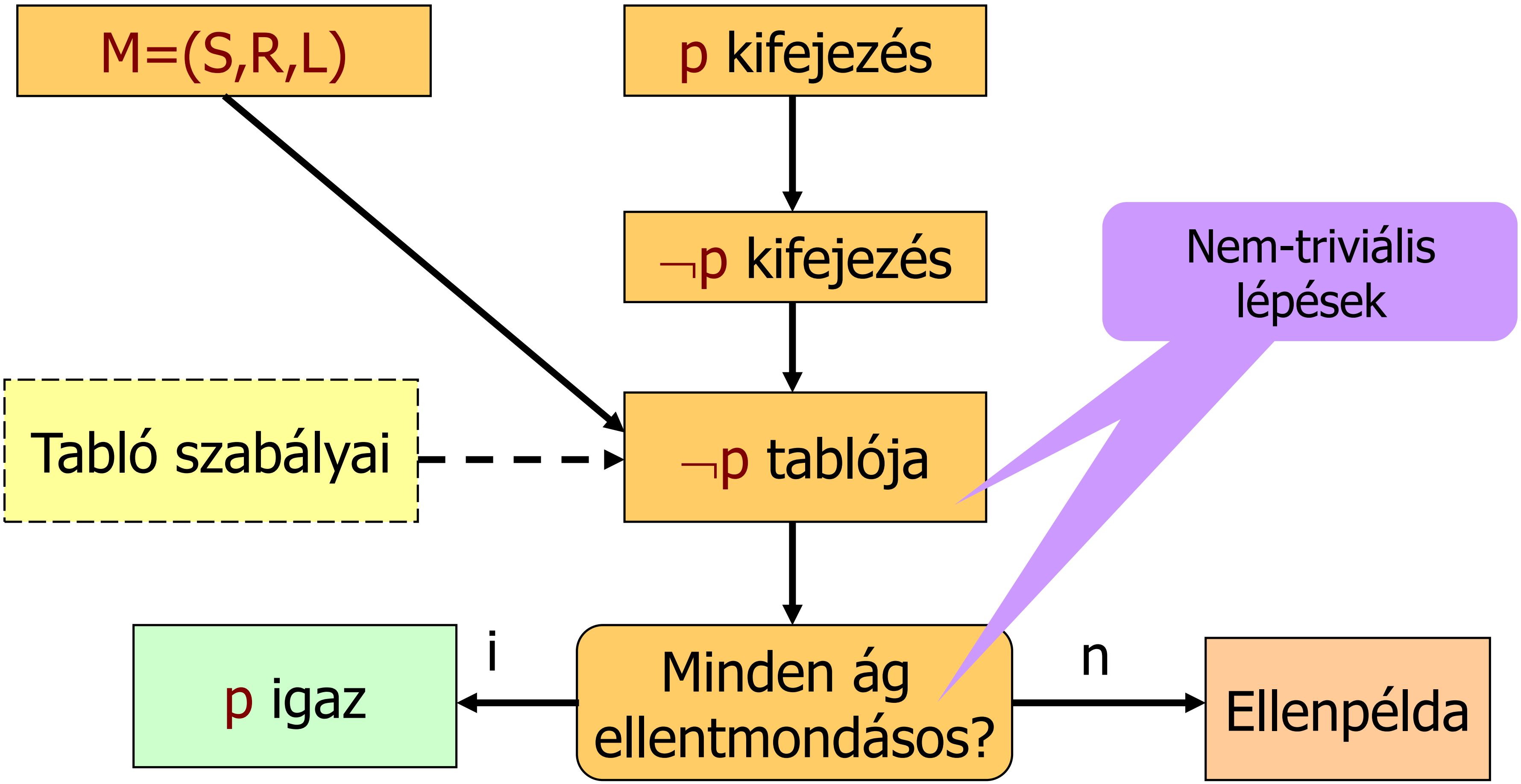
# Egy példa: A modellellenőrzés eredménye

- A tabló eredménye a negált kifejezésre:
  - Egy ellentmondásos ág (a negált kifejezés nem teljesül)
  - Két ellentmondás nélküli ciklus (a negált kifejezés teljesül)
- Következtetés:
  - Vannak olyan lefutások, ahol a negált kifejezés teljesül:
    Ciklus 1: s1, s2, s5, …
    Ciklus 2: s1, s5, …
  - Ezek a (még nem negált) eredeti kifejezés ellenpéldái

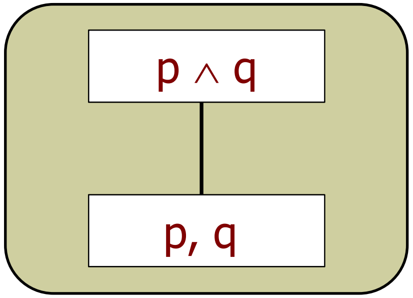


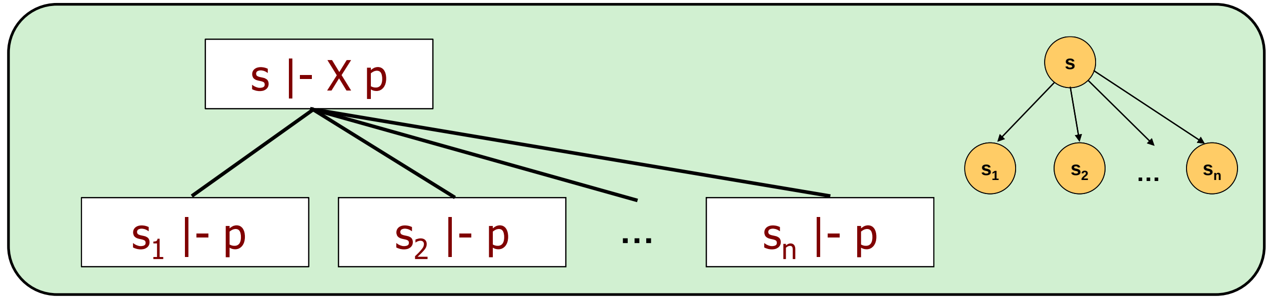

- Az eredeti kifejezés Zöld $\Rightarrow$ F Piros tehát nem igaz
  - Ellenpéldák adhatók

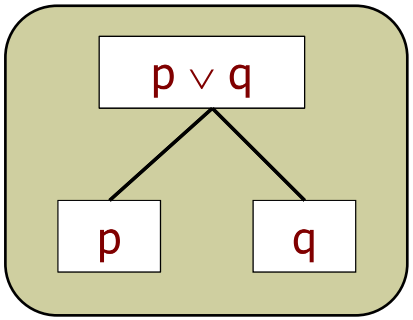
# A tabló módszer (összefoglalás)

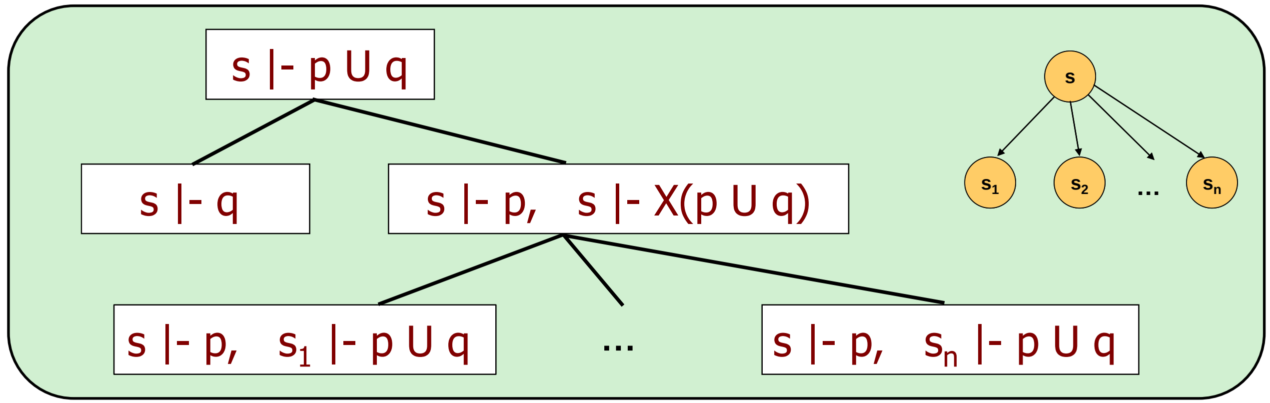


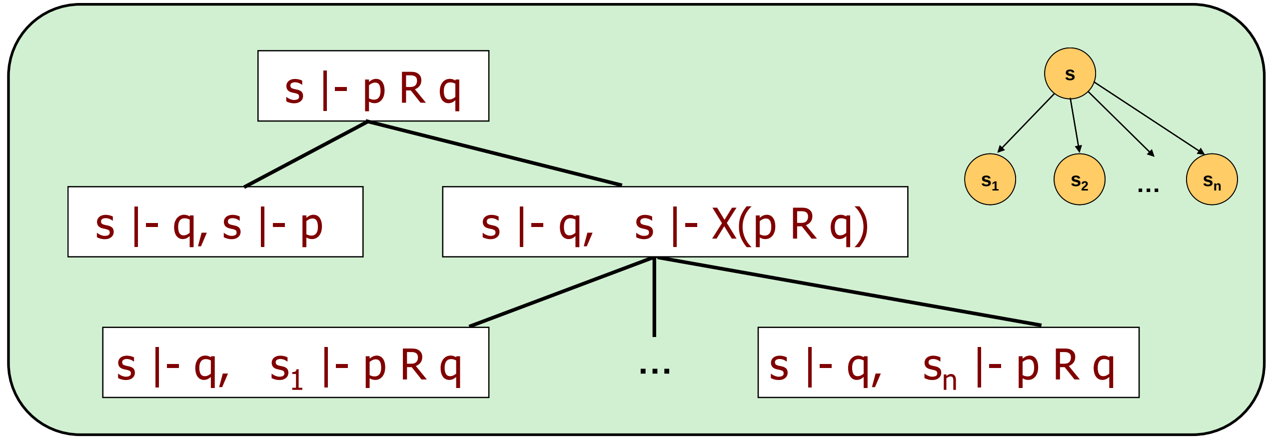
# Tabló felbontási szabályok (összefoglalás)



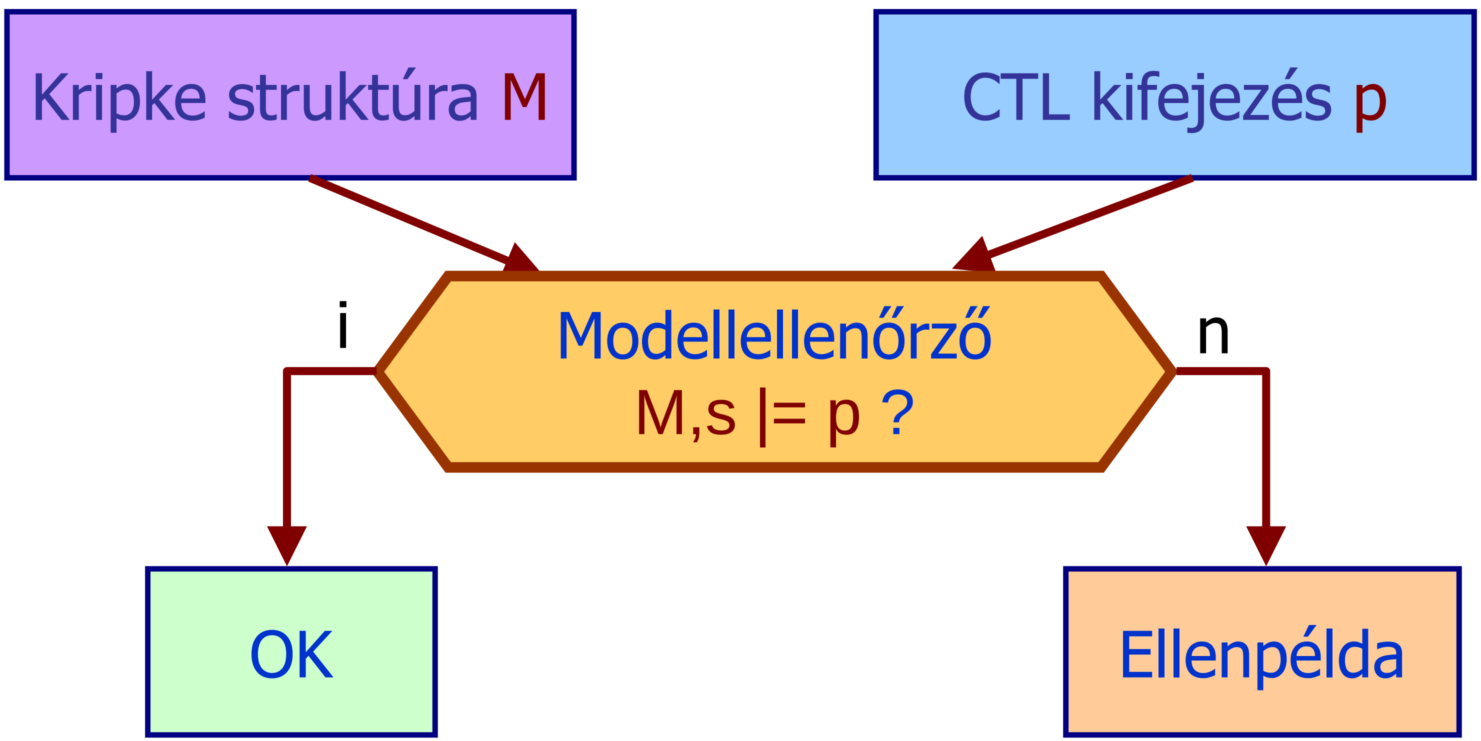




A negált kifejezés felbontásaként kapott sikeres ágak az ellenpéldák

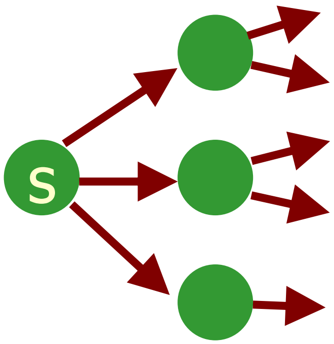
# CTL modellellenőrzés szemantika alapon

# A modellellenőrzés feladata

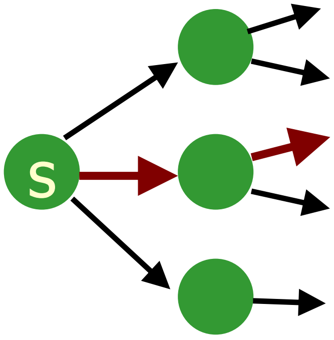


# Ismétlés: Elágazó idejű temporális logikák

- Útvonal kvantorok:
  - A: „for All paths”,
    minden lehetséges útra az
    adott állapotból kiindulva



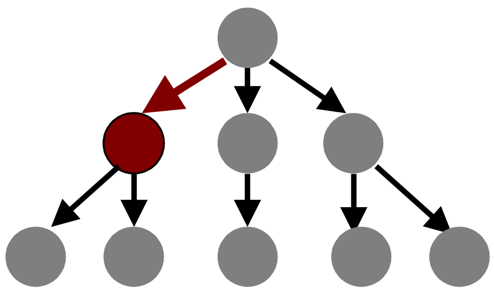

  - E: „Exists path”, „for some path”,
    legalább egy útra az
    adott állapotból kiindulva



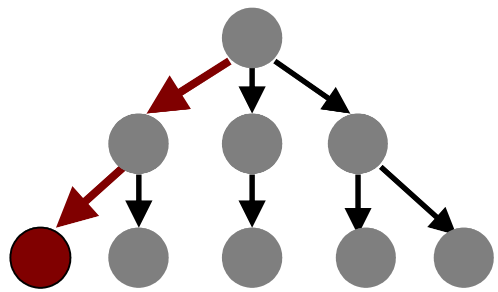

- Útvonalakon kiértékelhető operátorok (mint LTL):
  - X p, F p, G p, p U q

# Ismétlés: Elágazó idejű temporális logika: CTL

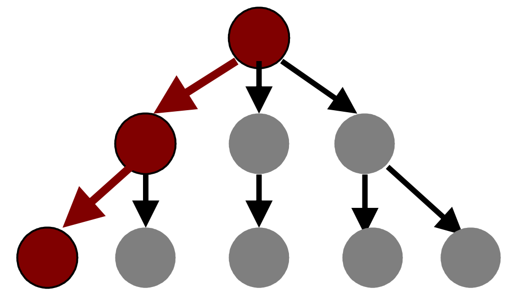
## CTL: Állapotokon kiértékelhető összetett operátorok



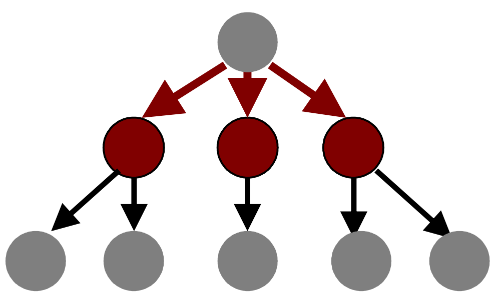
EX P



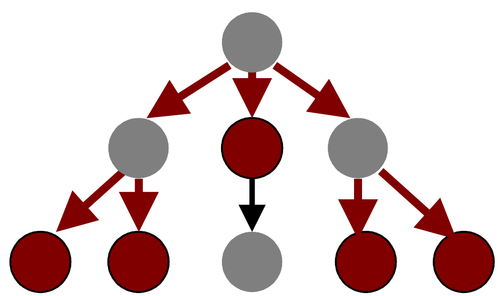
EF P



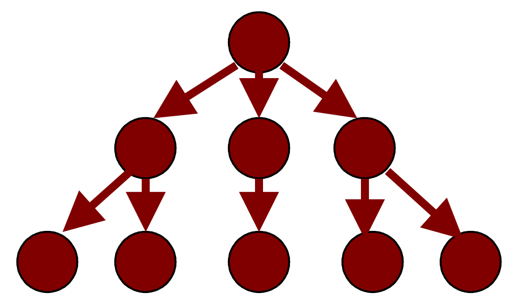
EG P



AX P



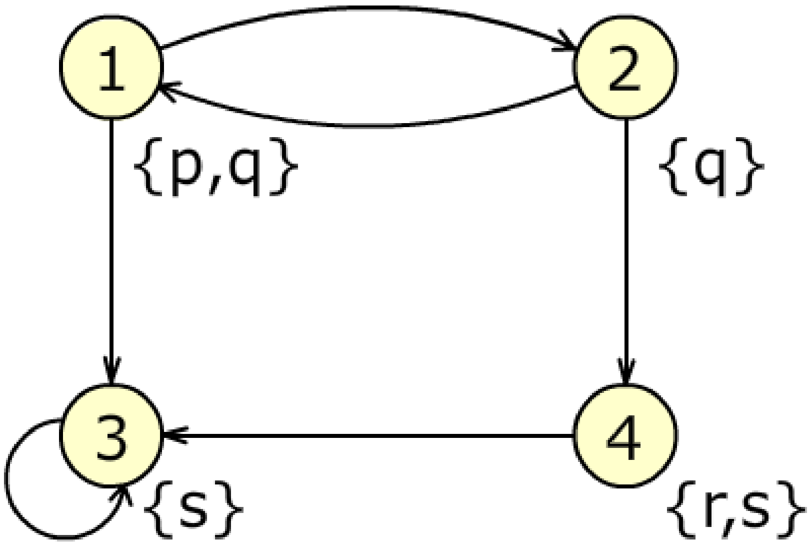
AF P



AG P

valamint E(p U q) illetve A(p U q)

# A CTL temporális logika használata




Mely állapotok esetén igaz:

- AX s

  3, 4
- E (q U r)

  1, 2, 4
- EG s

  3, 4
- EF EG s

  1, 2, 3, 4

  Vizsgálat:
  - Hol igaz az EG s
  - Honnan lehet eljutni olyan állapotba, ahol EG s igaz

Az EG s címkeként felrakható ezekre az állapotokra

# Alapötlet: Állapotok címkézése

- Globális modellellenőrzés p kifejezésre:
  - Címkézzük fel p-vel a modell összes olyan állapotát, ahol a p kifejezés igaz
    - A modellen igaz p, ha a kezdőállapoton szerepel a p címke
- A címkézés az eredeti kifejezés egyszerűbb rész-kifejezéseivel kezdve történik
  - Első lépés: Első részkifejezések az atomi kijelentések, ezek már szerepelnek címkeként
  - Következő lépések: Összetett részkifejezések p-ben, amik egy operátor használatával az eddig már címkeként megjelent részkifejezésekből adódnak
  - A címkézés vége: Az eredeti p kifejezés lesz a címke

# CTL modellellenőrzés állapot címkézéssel

- Állapotok címkézése: ahol igaz egy adott (rész)kifejezés
- Összetett kifejezés esetén hogyan történik a címkézés?
  - Kifejezések felbontása azok szintaktika struktúrája alapján, „belülről kifelé” haladva, egy-egy újabb operátort alkalmazva:

$$AF\ (\ P \wedge E\ (Q\ U\ R))$$

Első lépés: P, Q, R

Következő: E(Q U R)

Következő: P∧E(Q U R)

Utolsó: AF(P∧E(Q U R))

- Algoritmus az összetett kifejezés felbontása alapján:
  - Kiindulás: A modell (KS) címkézve van atomi kijelentésekkel
  - Továbblépés: Címkézés az egyre összetettebb részkifejezésekkel
  - Szabályok: Ha p illetve q címkék már vannak, akkor megadható, hol lehet $\neg p$, $p \wedge q$, EX p, AX p, E(p U q), A(p U q) címke
    - Így haladunk egy összetett kifejezésben „belülről kifelé”

# Szabályok: Atomi kijelentések és Boole operátorok

- P atomi kijelentés azokban az s állapotokban igaz, ahol $P \in L(s)$
  - A modellben P szerepel s címkéi között
- $\neg P$ azokban az s állapotokban igaz, ahol $P \notin L(s)$
  - Ahol $P \notin L(s)$, az állapot $\neg P$ kifejezéssel címkézhető
- $p \wedge q$ azokban az s állapotokban igaz, ahol p és q is igaz
  - Egy állapot címkézése lehet $p \wedge q$, ha címkéi között már szerepel p és q

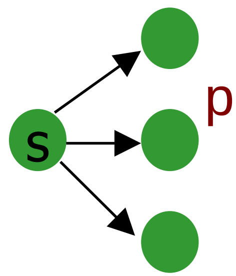
Temporális operátorok: EX, AX, E( U ), A( U )

- Bonyolultabb szabályokat igényel a címkézés!

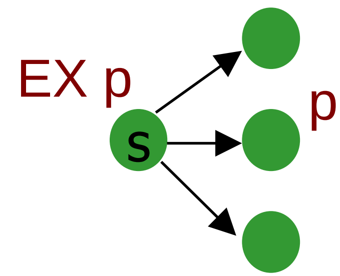
# Szabályok: Az AX, EX alakú kifejezések

- Szemantika: EX p azokban az s állapotokban igaz, amelyeknek van olyan rákövetkező állapota, ahol p igaz
  - Egy állapot címkézése lehet EX p, ha van olyan rákövetkező állapota, ami p-vel címkézett

Ha

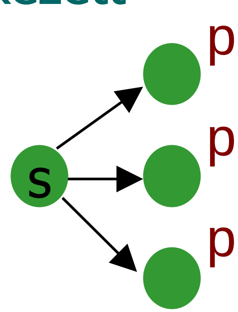



akkor

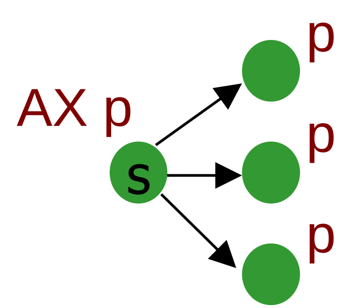



- Szemantika: AX p azokban az s állapotokban igaz, amelyeknek minden rákövetkező állapotában p igaz
  - Egy állapot címkézése lehet AX p, ha minden rákövetkező állapota p-vel címkézett

Ha




akkor



# Szabályok: Az E(p U q) kifejezések

- Hol igaz E(p U q)?
  - Szemantika alapján: $E(p\ U\ q) = q \vee (p \wedge EX\ E(p\ U\ q))$
  - Megadja: Mi kell a teljesítéshez lokálisan ill. a következő állapotban
- Tehát mely s állapotok címkézhetők E(p U q)-val?
  - Ha s címkézett q-val, vagy
  - ha s címkézett p-vel, és legalább egy rákövetkezője (ld. EX) már címkézett E(p U q)-val
- Iteráció adódik a címkézésre:
  - Először a q-val már címkézett állapotok adják azokat az állapotokat, ahol megjelenik az E(p U q) címke
  - Ezek megelőző állapotait kell végignézni: Ha szerepel ott a p címke, akkor ráteheti az E(p U q) címke is (hiszen p teljesül és van olyan rákövetkező, ahol E(p U q) teljesül)

  Így visszafelé járjuk be azokat az útvonalakat, amik p-vel címkézett állapotokon keresztül visznek q-val címkézett állapotba

# Az E(p U q) modellellenőrzés alapötlete

$$E(p\ U\ q) = q \vee (p \wedge EX\ E(p\ U\ q))$$

Hová rakható E (p U q) címke?

Ahol q címke már van …

vagy

ahol p címke már van …

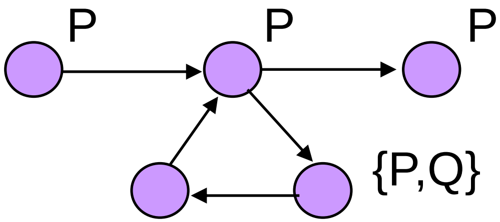
és

van legalább egy rákövetkezője, ahol már van E(p U q) címke

Címke ráhelyezés első lépése

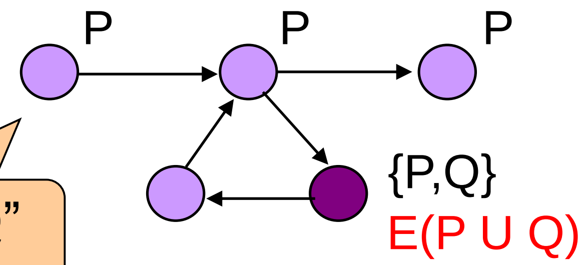
Iteratívan bővíthető a címkehalmaz (amíg bővül)

# Példa: Az E(P U Q) címkézés iterációja



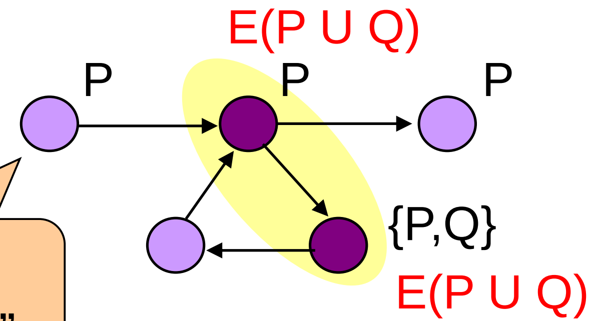

Kripke struktúra a kezdő címkézéssel

Első lépés: „Q”



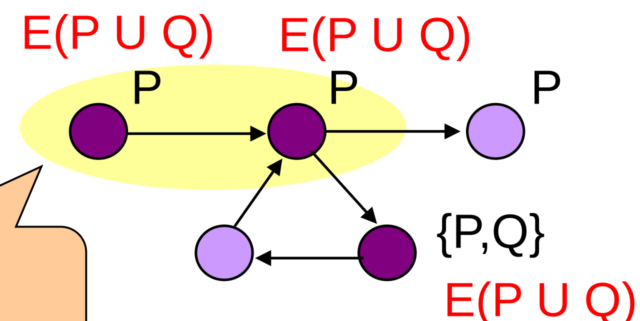

E(P U Q) =
$Q \vee (P \wedge EX\ E(P\ U\ Q))$

Második lépés: „P∧ EX”




Az iteráció addig tart, míg nő az állapothalmaz (fixpontot érünk el)

Harmadik lépés: „P∧ EX”



# Szabályok: Az A(p U q) kifejezések

- Hol igaz A(p U q)?
  - Szemantika alapján: $A(p\ U\ q) = q \vee (p \wedge AX\ A(p\ U\ q))$
  - Megadja: Mi kell a teljesítéshez lokálisan ill. a következő állapotban
- Tehát mely s állapotok címkézhetők A(p U q)-val?
  - Ha s címkézett q-val, vagy
  - ha s címkézett p-vel, és minden rákövetkezője (ld. AX) már címkézett A(p U q)-val
- Iteráció adódik:
  - Először a q-val már címkézett állapotok adják azokat az állapotokat, ahol megjelenik az A(p U q) címke
  - Ezek megelőző állapotait kell végignézni:
    Ha szerepel ott a p címke, és minden rákövetkező állapotukon szerepel az A(p U q) címke,
    akkor ezekre is ráteheтő az A(p U q) címke

Ezzel a formális szintaxisban használt operátorokat lefedtük!

# Iteráció halmazműveletekkel: Definíciók

- Ismétlés: Címkézési szabályok
  - Mely s állapotok címkézhetők E(p U q)-val?
    - Ha s címkézett q-val, vagy
    - ha s címkézett p-vel,
      és legalább egy rákövetkezője már címkézett E(p U q)-val
  - Mely s állapotok címkézhetők A(p U q)-val?
    - Ha s címkézett q-val, vagy
    - ha s címkézett p-vel,
      és minden rákövetkezője már címkézett A(p U q)-val
- Hogyan definiálhatók ezek az állapothalmazok?
  - Már címkézett Z állapothalmaz alapján:

$\mathrm{pre}_E$(címkézett)

$\mathrm{pre}_A$(címkézett)

$\mathrm{pre}_E(Z) = \{s \in S \mid$ létezik olyan s', hogy $(s,s') \in R$ és $s' \in Z\}$

$\mathrm{pre}_A(Z) = \{s \in S \mid$ minden s'-re, ahol $(s,s') \in R$: $s' \in Z\}$

Legalább egy rákövetkezője címkézett (Z-ben)

Minden rákövetkezője címkézett (Z-ben)

# Iteráció halmazműveletekkel: Algoritmus

- A címkézés bővítése halmazműveletekkel történik
- Definiált jelölések:
  - $pre_E(Z)$: olyan állapotok, amelyek legalább egy rákövetkezője címkézett (Z-ben)
  - $pre_A(Z)$: olyan állapotok, amelyek minden rákövetkezője címkézett (Z-ben)
- E(p U q) címkézési szabály: s állapotok címkézhetők E(p U q)-val
  - Ha s címkézett q-val, vagy
  - ha s címkézett p-vel, és legalább egy rákövetkezője már címkézett E(p U q)-val
- E(p U q) címkézési algoritmus halmazműveletekkel:
  - Kezdőhalmaz: $Z_0 = \{s \mid q \in L(s)\}$, azaz q-val címkézettek
  - Címkézés bővítése: $Z_{i+1} = Z_i \cup (\{s \mid p \in L(s)\} \cap pre_E(Z_i))$

    Eddig címkézettek uniója …

    ... P-vel címkézettek és …

    … legalább egy rákövetkezője már címkézett
  - Iteráció vége: Ha $Z_{i+1} = Z_i$, azaz már nem bővül a halmaz

# CTL modellellenőrzés: Összefoglalás

- Globális modellellenőrzés:
  - Állapotok címkézése azokkal a (rész)kifejezésekkel, amelyek igazak az adott állapotban
  - Címkézés egyre összetettebb kifejezésekkel („belülről kifelé”), az atomi kijelentésekből indítva az összetettebb kifejezések felé
- Címkézés egy részkifejezéssel:
  - Az előző lépésben adott címkézés felhasználása az operátorok szemantikája alapján képzett szabályok szerint
  - EX, AX esetén: Megelőző állapot vizsgálata és címkézése
  - E(p U q), A(p U q) esetén: Inkrementális címkézés
    - Kezdőhalmaz: A belső kifejezések (p, q) által meghatározott állapothalmazok alapján
    - Iteráció: A szemantika alapján, megelőző állapotokat címkézve
    - Iteráció vége: Nem nő a címkézett állapotok halmaza

# A bevezető példa kifejtése

- Kifejezések felbontása azok struktúrája alapján, és „belülről kifelé” címkézés:

$AF\ (P \wedge E\ (Q\ U\ R))$

Q és R címkék a KS-ban

Inkrementális címkézés: E(. U .)
Az iteráció végén megjelenik az E(Q U R) címke

Itt P-vel és E(Q U R)-val címkézett állapothalmazok metszete, építve a már meglévő E(Q U R) címkékre:
Megjelenik a P∧E(Q U R) címke

Inkrementális címkézés: AF alapján, építve a már meglévő P∧E(Q U R) címkékre:
Megjelenik az AF(P∧E(Q U R)) címke.
Ez a kezdőállapotra ellenőrizhető.

# Gyakorló feladat

- Egy speciális jelzőlámpa 3 égőt tartalmaz: piros, sárga és zöld.
  - A jelzőlámpa alaphelyzetében mindhárom égő kikapcsolt.
  - Bekapcsolás után rögtön a piros égő világít.
  - Innen két választási lehetőség adódik a továbblépésre: egyik esetben a lámpa piros-sárgára (mindkettő világít), másik esetben zöldre vált.
  - A választástól függően a piros-sárga után következik a zöld, míg a zöld után újra a piros, majd ezekből az ismert állapotokból folytatódik tovább a működés.
- A jelzőlámpa alaphelyzetéből kiindulva teljesül-e:

$$E((\neg \text{ piros})\ U\ (EX \text{ zöld}))$$

# Összefoglalás

- LTL modellellenőrzés
  - Tabló konstruálása
    - Boole-logikai bevezetés: Ellentmondásos és sikeres ágak
    - Tabló LTL esetén: Ellenpélda keresés (negált követelményre)
- CTL modellellenőrzés
  - Szemantika alapú modellellenőrzés
    - Inkrementális címkézés bővülő részkifejezésekkel (globális modellellenőrzés)
    - Halmazműveletekkel történik

Hogyan tehető hatékonnyá ez az algoritmus?

# LTL modellellenőrzés: Automata alapú megközelítés

(Kitekintés, kiegészítő anyag)

# Automaták véges hosszúságú szavakon

- $A=(\Sigma, S, S_0, \rho, F)$ ahol
  - $\Sigma$ az ábécé, S állapotok, $S_0$ kezdőállapotok
  - $\rho$ az állapotátmeneti reláció, $\rho: S \times \Sigma \to 2^S$
  - F az elfogadó állapotok halmaza
- Az automata futása:
  - Egy beérkező $w=(a_0, a_1, a_2, \ldots a_n)$ betűsorozat hatására egy $r=(s_0, s_1, s_2, \ldots s_n)$ állapotsorozat
  - r elfogadó futás, ha $s_n \in F$
  - Egy w szót elfogad az automata, ha létezik rá elfogadó futás
- $L(A)=\{ w \in \Sigma^* \mid w \text{ elfogadott } \}$ az automata által elfogadott nyelv

# Automaták végtelen hosszúságú szavakon

- Alkalmazás: Folyamatosan működő rendszerek
  - Nem ellenőrizhető, hogy a végállapot elfogadó-e
- Büchi elfogadási kritérium:
  - Egy beérkező $w=(a_0, a_1, a_2, \ldots)$ betűsorozat hatására egy $r=(s_0, s_1, s_2, \ldots)$ állapotsorozat
  - lim(r) = {s | s előfordul végtelenül sokszor, azaz nincs olyan j, hogy $\forall k>j$: $s \neq s_k$}
  - Elfogadó a futás, ha $\lim(r) \cap F \neq 0$
  - Egy w szót elfogad az automata, ha létezik rá elfogadó futás (azaz végtelen sokszor érint elfogadó állapotot)
- $L(A)=\{w \in \Sigma^* \mid w \text{ elfogadott}\}$ az automata által elfogadott nyelv

# Az automata alapú módszer

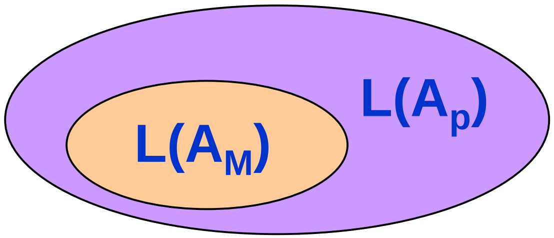
- A KS egy s állapotához: L(s) a betű a $2^{AP}$ ábécéből
  - Pl. {Piros, Sárga} az ábécé egy betűje
- A $\pi=(s_0, s_1, s_2, \dots s_n)$ útvonal egy szót azonosít:

  $$(L(s_0), L(s_1), L(s_2), \dots L(s_n))$$

- Két automatát kell konstruálni:
  - M=(S,R,L) alapján egy $A_M$ automata konstruálható, amely azokat és csakis azokat a szavakat fogadja el, amelyek megfelelnek M útjainak.
  - p kifejezés alapján egy $A_p$ automata konstruálható, amely azokat és csakis azokat a szavakat fogadja el, amelyek olyan utakhoz tartoznak, ahol p igaz
    - Itt felhasználhatók a tabló képzés szabályai: mi kell teljesüljön egy adott állapotban, és mi a rákövetkezőben (ld. X után)

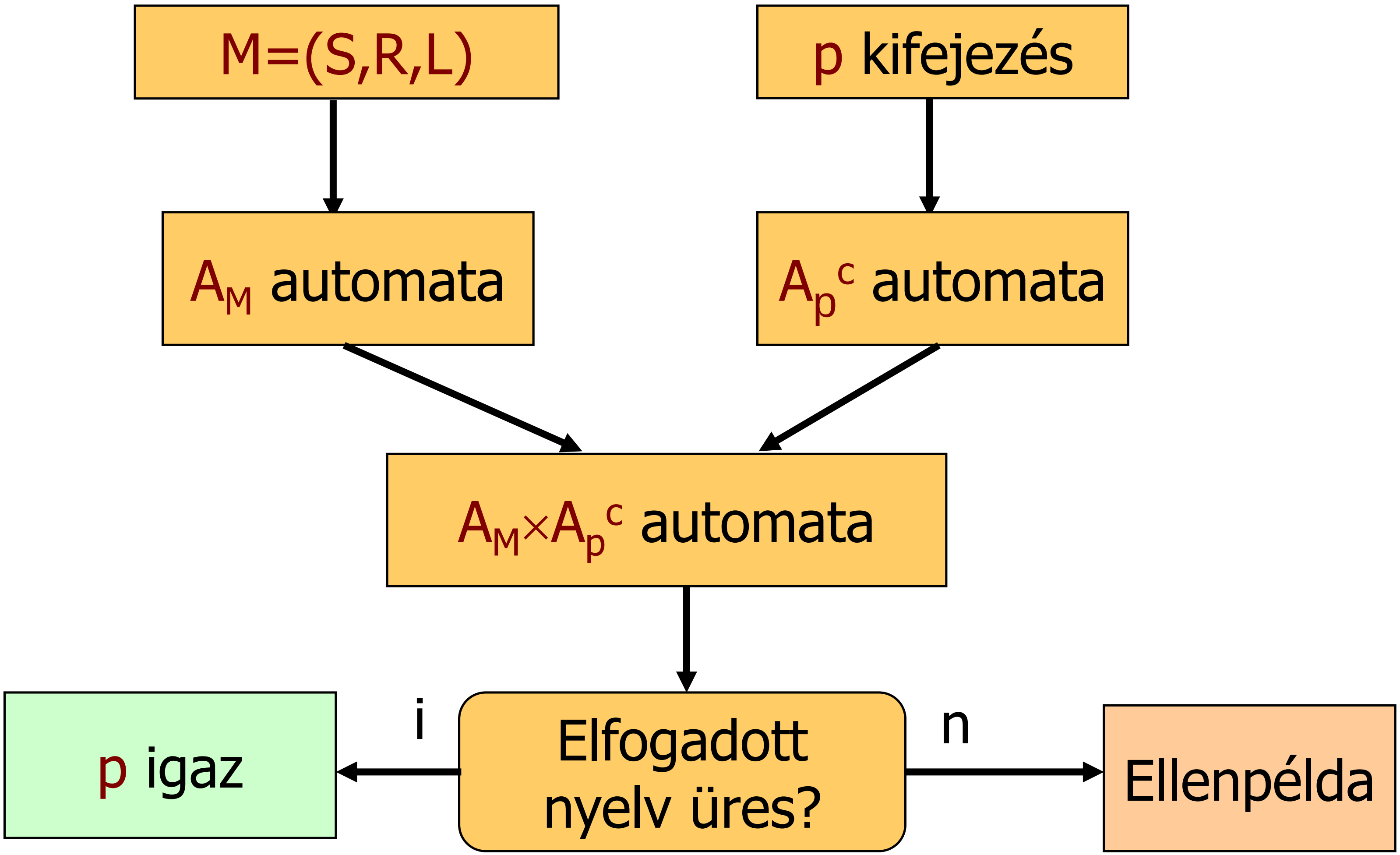
# Modellellenőrzés az automatákkal

- Modellellenőrzési feladat: $L(A_M) \subseteq L(A_p)$, vagyis a „modell” nyelv része-e a „tulajdonság” nyelvnek?
  - Ha igen, akkor $M \models p$




- A kérdés átalakítása:
  - Nyelvek metszetének ürességét kell vizsgálni:
    $L(A_M) \cap L(A_p)^c = 0$, itt $L(A_p)^c$ a komplementer nyelv
  - Az $A_M$ „modell automata” és az $A_p^c$ „komplementer tulajdonság automata” $A_M \times A_p^c$ szinkron szorzatát képezve, az általa elfogadott nyelv üres-e?
    - Ha üres, akkor $M, \pi \models p$ teljesül
    - Az elfogadott nyelv üres, ha nincs elérhető elfogadó állapot
- Folyamatosan működő rendszerek:
  - Automaták végtelen hosszúságú szavakon;
    Büchi elfogadási kritérium: cikluskeresésre vezet

# Az automata alapú modellellenőrzés



# „On-the-fly" modellellenőrzés

- Alapötlet:
  - Az $A_p$ automata generálása közben lehet elvégezni a szinkron szorzat automata konstruálását is
- Szinkron szorzat automata konstruálása:
  - Az ellenőrizendő kifejezés által vezérelten történik: ahogyan az $A_p$ automata új állapota előáll, úgy kell $A_M$ állapotait „előkeresni"
  - Nem szükséges hozzá a modell állapottér teljes generálása
    - Pl. egy magasabb szintű modellből való származtatás esetén