

Hennessy-Milner logika Lineáris temporális logika és modellellenőrzése

Majzik István

Budapesti Műszaki és Gazdaságtudományi Egyetem

Méréstechnika és Információs Rendszerek Tanszék

<http://www.mit.bme.hu/~majzik/>

Tartalomjegyzék

- **Temporális követelmények**
- A Hennessy-Milner logika
 - Modellellenőrzés a tabló módszerrel
- A PLTL lineáris temporális logika
 - Operátorok
 - Modellellenőrzés automata-elméleti alapon
 - A modellellenőrzés komplexitása

Motiváció: Mit szeretnénk elérni?

- Alacsony szintű formalizmusok (KS, LTS, KTS, TA)
- Magasabb szintű formalizmusok

Temporális logikák:
PLTL, CTL, CTL*

Rendszer modellje

Követelmény megadása

Automatikus
modell ellenőrző

OK

Ellenpélda

i

n

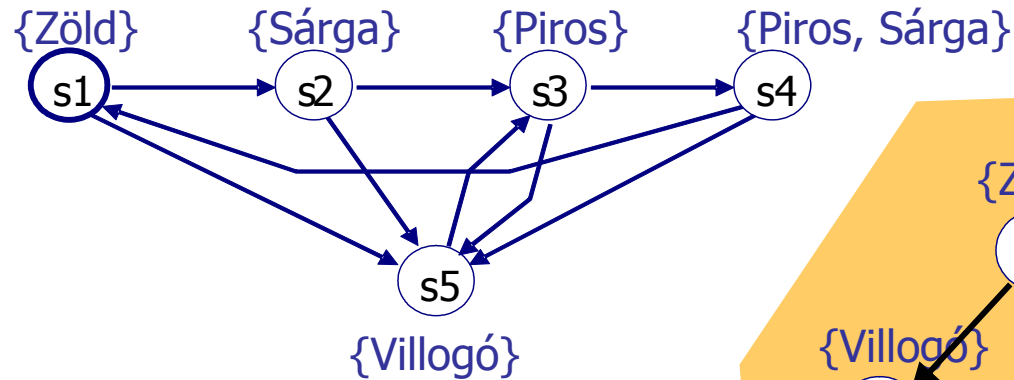
Tipikus követelmények

- **Biztonsági követelmények:**
 - Veszélyes, nemkívánatos helyzetek elkerülését fogalmazzák meg
 - Pl. holtpon, jogosulatlan hozzáférés, veszélyes állapot elkerülése
 - Invariáns jellegű tulajdonság, univerzális kvantor az elérhető állapotokon („mindig ...”)
 - Ha egy állapotsorozat nem teljesíti, akkor nem is egészíthető ki úgy, hogy teljesítse
- **Élőségi követelmények:**
 - Kívánatos helyzetek elérését írják elő
 - Pl. eredmény, válasz, kiszolgálás bekövetkezik
 - Egzisztenciális kvantor az állapotokon („létezik ...”)
 - Ha egy állapotsorozat nem teljesíti, akkor elvileg kiegészíthető úgy, hogy teljesítse

Követelmények leíró nyelve

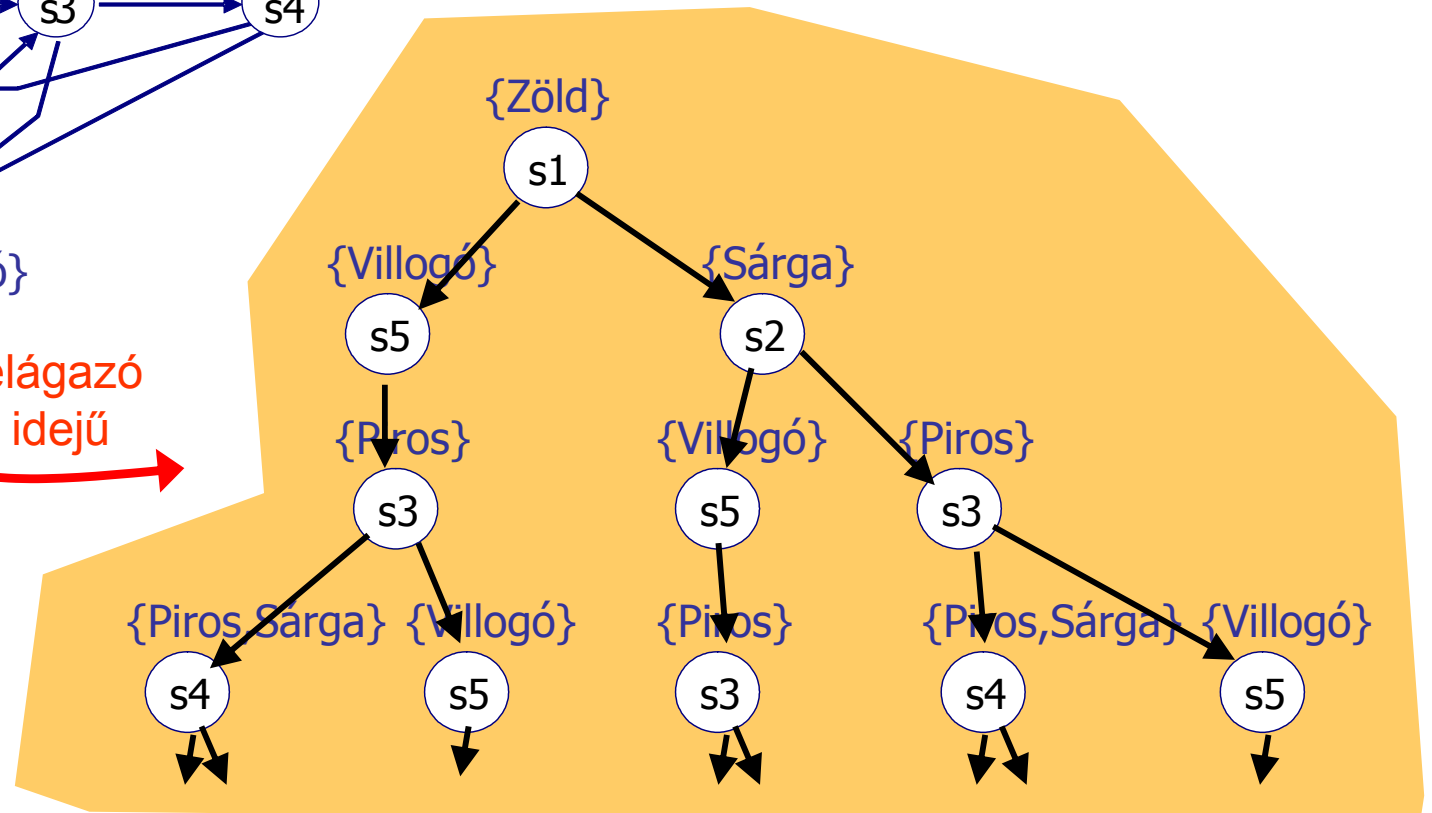
- **Elérhetőség:** Több állapot bekövetkezési sorrendjére vonatkozó követelmények
 - Megfeleltethető a **logikai időnek**:
 - Jelen időpillanat: Aktuális állapot
 - Következő időpillanat(ok): Rákövetkező állapot(ok)
 - **Temporális operátorok** (logikai időbeli, sorrendi operátorok) használhatók a követelmények kifejezésére
- **Temporális logikák:**
 - Formális rendszer arra, hogy **kijelentések igazságának logikai időbeli változását vizsgálhassuk**
 - Temporális operátorok: „mindig”, „valamikor”, „mielőtt”, „addig, amíg”, „azelőtt, hogy”, ...

Lineáris és elágazó idejű temporális logikák

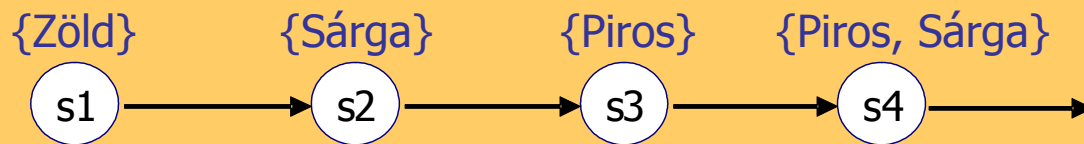


lineáris idejű

elágazó idejű



Logikai idő
elágazó idővonalak
mentén



Logikai idő mint idővonal (egy-egy lefutás)

Modell ellenőrzés

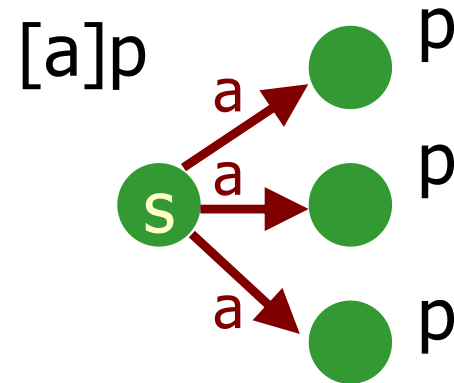
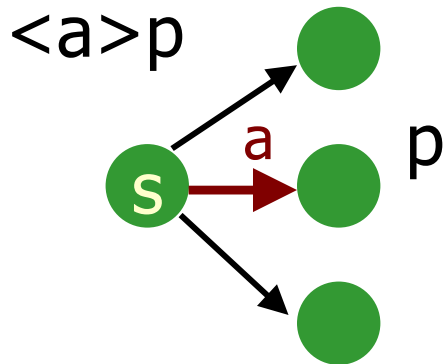
- Annak vizsgálata, hogy egy p temporális logikai kifejezés igaz-e az M modellen
- Fajtái:
 - **Lokális**: M adott kezdőállapotára
 - **Globális**: Minden kezdőállapot keresése, ami esetén p igaz lesz
- **Technikák**:
 - **Szemantikán alapuló**: Az operátorok szemantikája alapján „keresés” az állapot térben
 - **Automata-elméleti**: Visszavezetés az automata-elmélet eredményeire
 - **Tabló módszer**: Bizonyításkereséssel (a kifejezés felbontásával) kideríteni, hol lehet igaz a kifejezés

Tartalomjegyzék

- Temporális követelmények
- **A Hennessy-Milner logika**
 - Modellellenőrzés a tabló módszerrel
- A PLTL lineáris temporális logika
 - Operátorok
 - Modellellenőrzés automata-elméleti alapon
 - A modellellenőrzés komplexitása

A Hennessy-Milner logika

- LTS-en értelmezett egyszerű logika
- **Véges akciószekvenciák** (mint követelmények) megadására alkalmas (pl. teszt lefutása)
- Szintaxis:
 $HML ::= true \mid false \mid p \wedge q \mid p \vee q \mid [a]p \mid \langle a \rangle p$
- A temporális operátorok intuitív jelentése:



Hennesy-Milner logika szemantikája

$T=(S, \text{Act}, \rightarrow)$ LTS-en értelmezett

Jelölés: $T, s \models p$ jelentése: T LTS s állapota esetén igaz p

- **H1:** $T, s \models \text{true}$, $T, s \not\models \text{false}$
- **H2:** $T, s \models p \wedge q$ a.cs.a. $T, s \models p$ és $T, s \models q$
 $T, s \models p \vee q$ a.cs.a. $T, s \models p$ vagy $T, s \models q$
- **H3:** $T, s \models [a]p$ a.cs.a. $\forall s'$ ahol $s \xrightarrow{a} s'$: $s' \models p$
- **H4:** $T, s \models \langle a \rangle p$ a.cs.a. $\exists s'$: $s \xrightarrow{a} s'$ és $s' \models p$

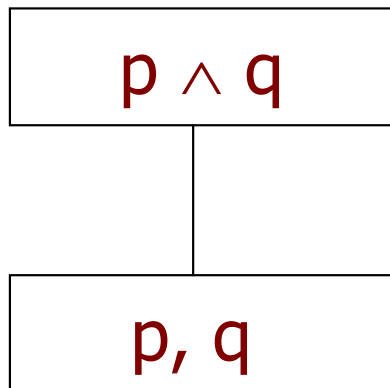
HML példák:

- $\langle a \rangle \text{true}$: igaz, ha létezik a -val címkézett kimenő átmenet
- $[a] \text{false}$: igaz, ha nincs a -val címkézett kimenő átmenet
- $\langle a \rangle \langle b \rangle \langle c \rangle \text{true}$: igaz, ha létezik az a, b, c szekvencia

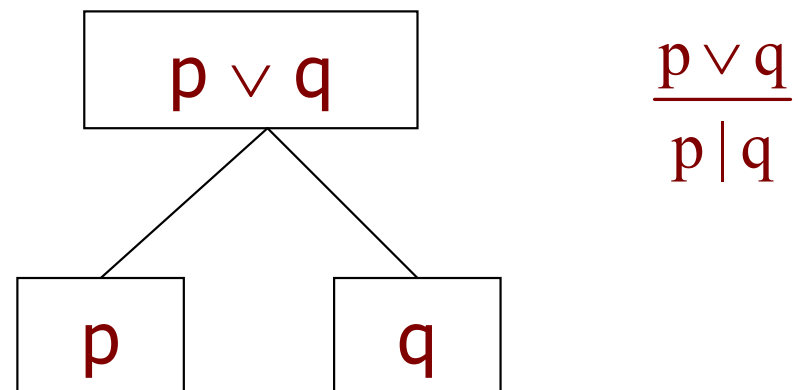
HML alapú modellellenőrzés: Tabló módszerrel

A tábló módszer bevezetése (Boole logika)

- A matematikai logika egy módszere
 - Arra a kérdésre válaszol, igazgá tehető-e egy adott kifejezés
- Tábló: Egy logikai kifejezés felbontása fa struktúrában
 - Csomópontok: (Rész-)kifejezések, amiket igazgá kell tenni
 - Élek: Rész-kifejezések viszonya (tábló konstrukciós szabály)
- A konstrukció előtt a kifejezést **negált normál formára** kell hozni (ld. de Morgan): Negálás csak változók előtt lehet
- Tábló konstrukciós szabályok Boole logika esetén:



$$\frac{p \wedge q}{p, q}$$



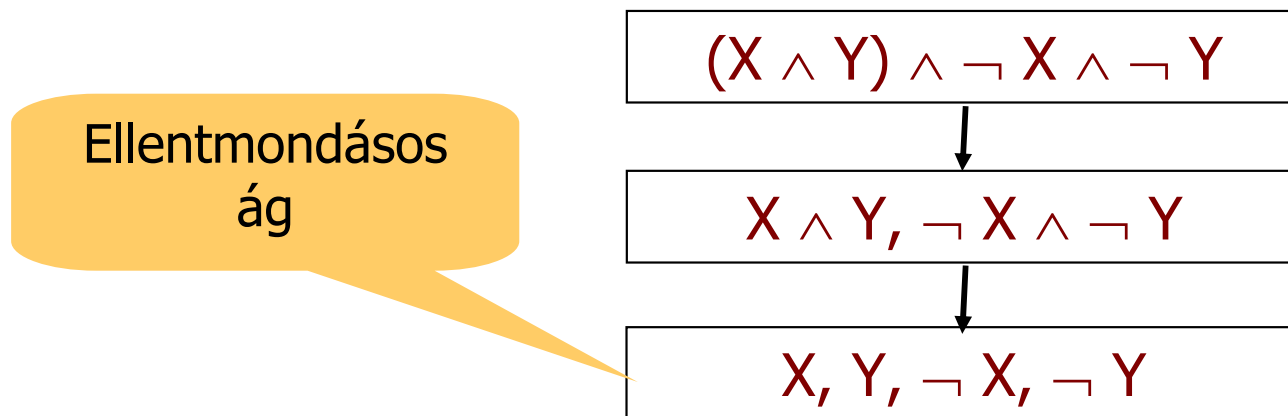
$$\frac{p \vee q}{p | q}$$

A tabló kiértékelése

- Egy ág (felbontás) terminálása: Csak (ponált vagy negált) változók maradhatnak a csomópontban
 - Ezek igaz vagy hamis értéket kaphatnak
- Egy ág terminálása után:
 - „Ellentmondásos” ág: Ugyanazon változó ponált és negált formában is előfordul, így nem lehet érvényes behelyettesítés
 - Pl. $p, \neg p, q$ esetén ellentmondás
 - „Sikeres” ág: Nincs ellentmondás, a kezdeti kifejezés az ág végén található behelyettesítéssel igazzá tehető
 - Pl: $p, \neg q$ esetén: p igaz, q hamis a behelyettesítés
- A fa nem ellentmondásos ágai jelölik ki, hol igaz a kifejezés
 - Modell ellenőrzésben: Az a kérdés, igaz-e a kifejezés
 - Ezért a negált kifejezést bontják fel
 - Ha minden ág ellentmondásos, akkor az eredeti kifejezés igaz!
 - Sikeres ág ellenpéldát ad

Egy példa (Boole logika)

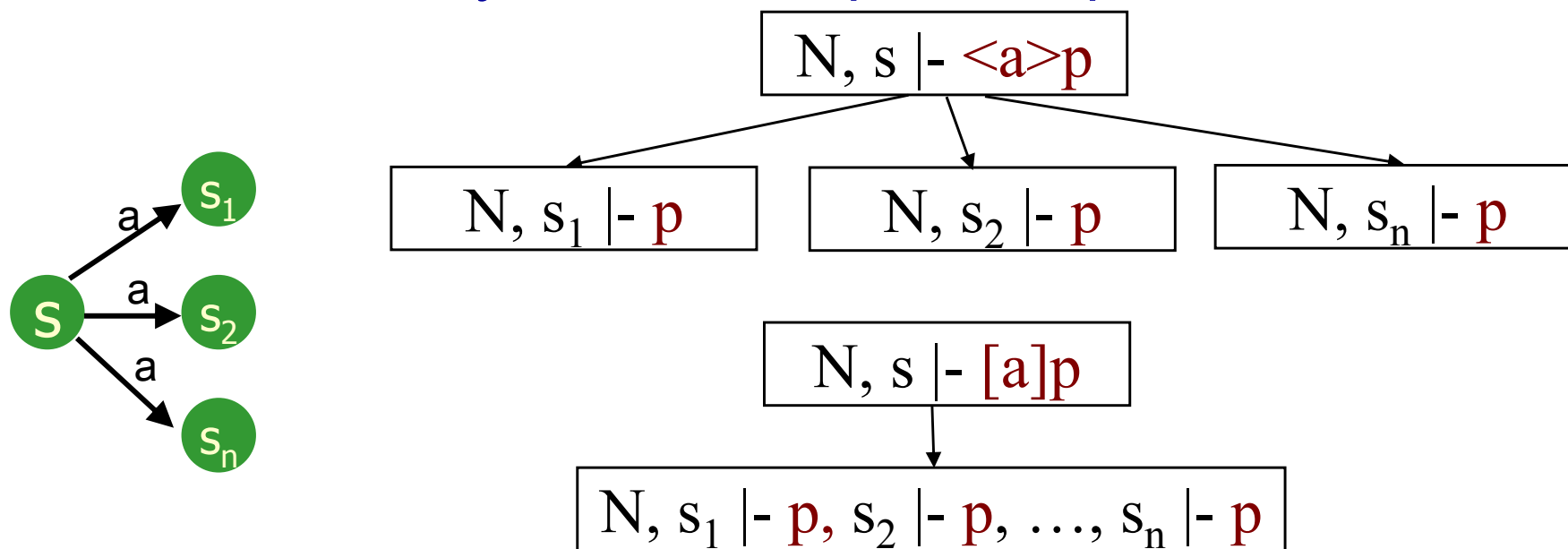
- Vizsgálandó kifejezés: $\neg ((X \wedge Y) \rightarrow (X \vee Y))$
- Az implikáció kifejtése: $\neg (\neg(X \wedge Y) \vee (X \vee Y))$
- Negált normál forma:
 $(X \wedge Y) \wedge \neg(X \vee Y)$
 $(X \wedge Y) \wedge (\neg X \wedge \neg Y)$
- Tabló konstruálás:



- Itt minden ág ellentmondásos
 - A kiindulásként használt kifejezés nem lehet igaz

Modell ellenőrzés HML esetén

- Felbontási szabályok HML temporális operátorok esetén:



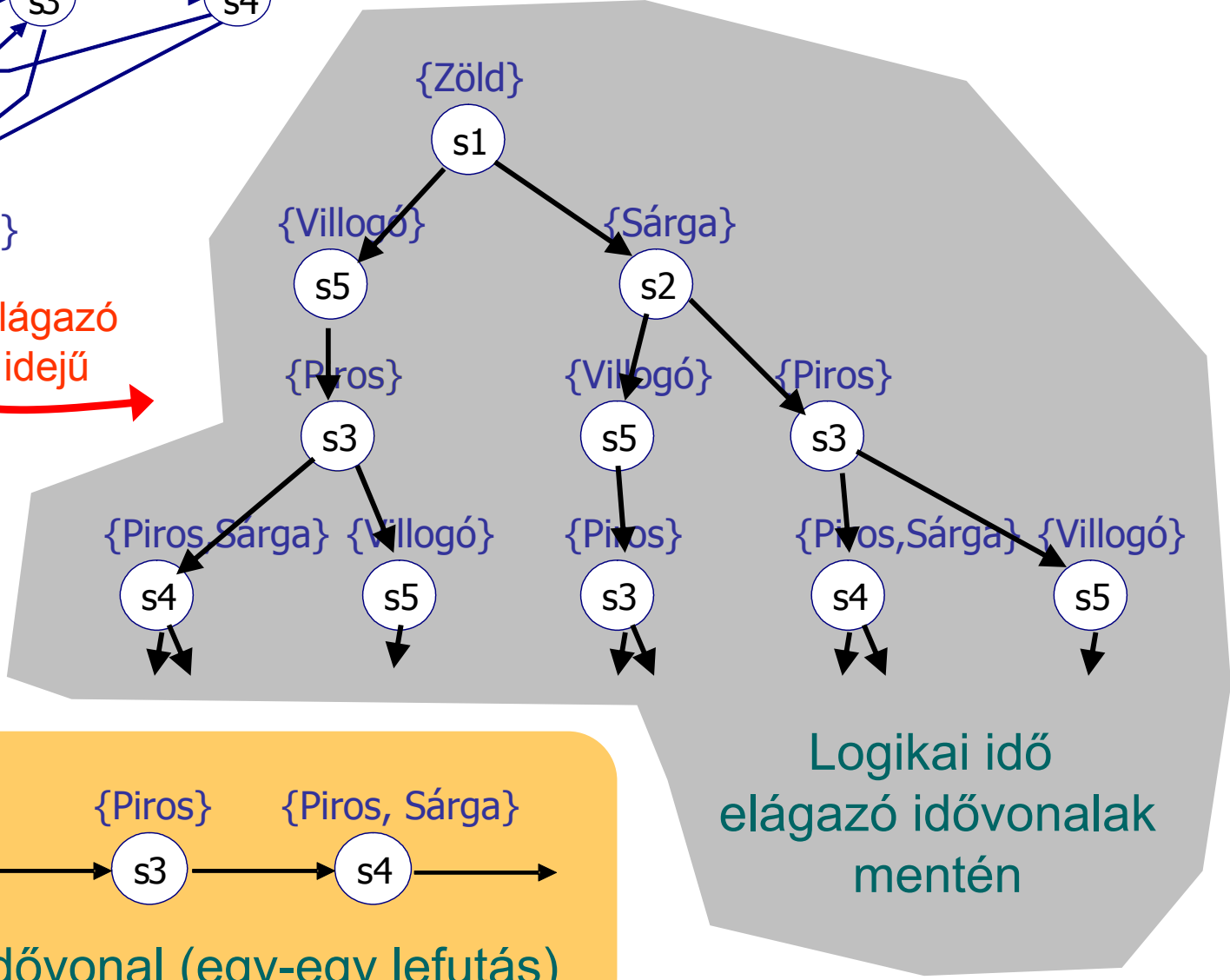
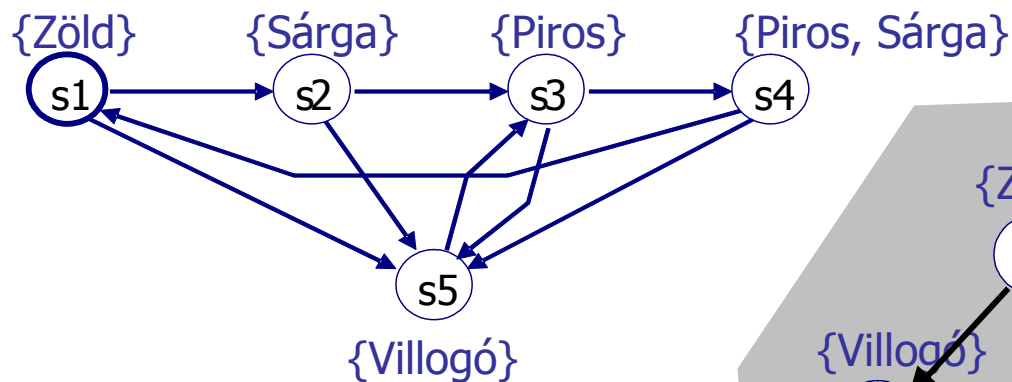
ahol $\{s_1, s_2, \dots, s_n\} = \{s' \mid s \xrightarrow{a} s'\}$

- A modell állapottere alapján történik a felbontás!
- Sikeres ág: $s \mid - \text{true}$ vagy $s \mid - [a]p$, ahol nincs a átmenet
- Igaz a kifejezés a modellen: Van sikeres ág

Tartalomjegyzék

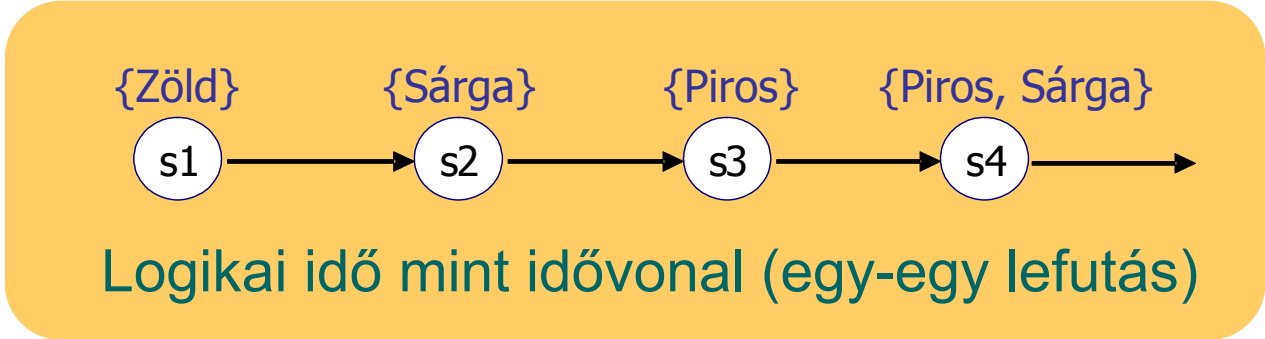
- Temporális követelmények
- A Hennessy-Milner logika
 - Modellellenőrzés a tabló módszerrel
- **A PLTL lineáris temporális logika**
 - Operátorok
 - Modellellenőrzés automata-elméleti alapon
 - A modellellenőrzés komplexitása

Lineáris és elágazó idejű temporális logikák



elágazó idejű

lineáris idejű



Logikai idő
elágazó idővonalak
mentén

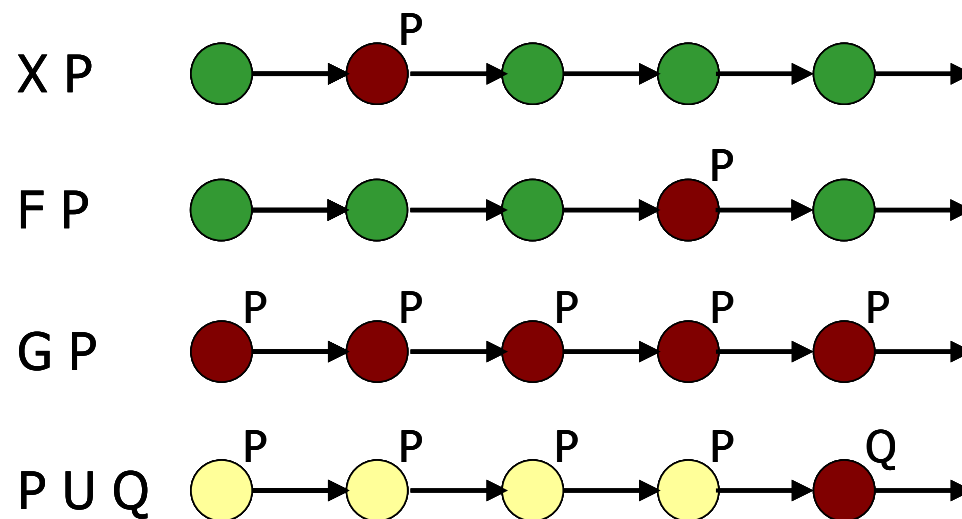
PLTL: Egy lineáris idejű temporális logika

PLTL (Propositional Linear Time Temporal Logic)

p, q, r, \dots kifejezések konstruálása:

- Atomi kijelentések (AP elemei): P, Q, \dots
- Boole logikai operátorok: $\wedge, \vee, \neg, \Rightarrow$
 \wedge : ÉS, \vee : VAGY, \neg : Negálás, \Rightarrow : Implikáció
- Temporális operátorok: F, G, X, U

informálisan:



PLTL formális szintaxis

Az érvényes PLTL kifejezések halmaza a következő szabályokkal képezhető:

- **L1:** Minden P atomi kijelentés egy kifejezés.
- **L2:** Ha p és q egy-egy kifejezés, akkor $p \wedge q$ illetve $\neg p$ is
- **L3:** Ha p és q egy-egy kifejezés, akkor $p U q$ illetve $X p$ is

BNF: $PLTL ::= P \mid \neg p \mid p \wedge p \mid X p \mid p U p$

Operátorok precedenciája növekvő sorrendben:

$\equiv, \Rightarrow, \vee, \wedge, \neg, (X, U)$

„Kifejezhető” temporális operátorok:

$F p$ jelentése $true U p$

$G p$ jelentése $\neg F(\neg p)$

$p WB q$ jelentése $\neg((\neg p) U q)$

$p B q$ jelentése $\neg((\neg p) U q) \wedge F q$

PLTL szemantika

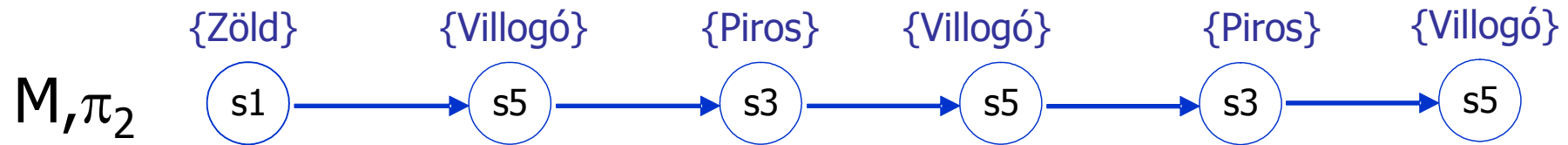
Jelölések:

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

Formális szemantika a szintaxis alapján képzett kifejezésekhez:

- **L1:** $M, \pi \models P$ a.cs.a. $P \in L(s_0)$
- **L2:** $M, \pi \models p \wedge q$ a.cs.a. $M, \pi \models p$ és $M, \pi \models q$
 $M, \pi \models \neg q$ a.cs.a. $M, \pi \models q$ nem igaz.
- **L3:** $M, \pi \models (p \cup q)$ a.cs.a.
 $\exists j \geq 0: (\pi^j \models q$ valamint $\forall 0 \leq k < j: \pi^k \models p)$
 $M, \pi \models X p$ a.cs.a. $\pi^1 \models p$

Egy példa



- $M, \pi_2 \models F (\text{Villogó} \Rightarrow X \text{Piros})$,
mert van olyan szuffix, hogy $\text{Villogó} \Rightarrow X \text{Piros}$ igaz



Tartalomjegyzék

- Temporális követelmények
- A Hennessy-Milner logika
 - Modellellenőrzés a tabló módszerrel
- **A PLTL lineáris temporális logika**
 - Operátorok
 - **Modellellenőrzés automata-elméleti alapon**
 - A modellellenőrzés komplexitása

A modellellenőrzés feladata

Ha nincs útvonal megadva, akkor a kezdőállapotból induló minden útra ellenőriz

Kripke struktúra M

PLTL kifejezés p

Modellellenőrző
 $M, \pi \models p ?$

OK

Ellenpélda

i

n

Az automata alapú megközelítés

- Betűk a 2^{AP} ábécéből
 - A KS egy s állapotához $L(s)$ betű tartozik
 - Pl. {Piros, Sárga} az ábécé egy betűje a példában
- 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.

Modell ellenőrzés az automatákkal

- Kérdés: $L(A_M) \subseteq L(A_p)$, vagyis a „modell” nyelv része-e a „tulajdonság” nyelvnek?

- Ha igen, akkor $M, \pi \models p$

- A kérdés átalakítása:

- Nyelvek metszetének üressége: $L(A_M) \cap L(A_p)^c = \emptyset$,
itt $L(A_p)^c$ a komplementer nyelv

- A „modell automata” és a „komplementer tulajdonság automata”

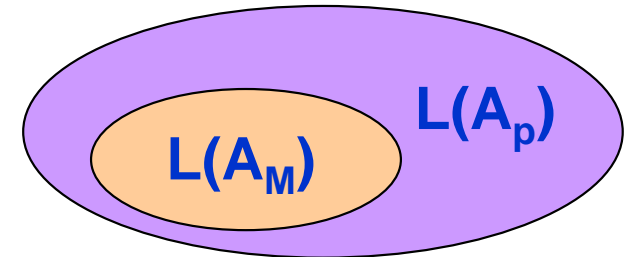
$A_M \times A_p^c$ szinkron szorzatát képezve, az általa elfogadott nyelv üres-e?

- Ha üres, akkor $M, \pi \models p$ teljesül.
- Az elfogadott nyelv üres, ha nincs elérhető elfogadó állapot.
- A_p^c konstruálása: teljesen definiált és determinisztikus esetben az elfogadó és nem elfogadó állapotok felcserélése

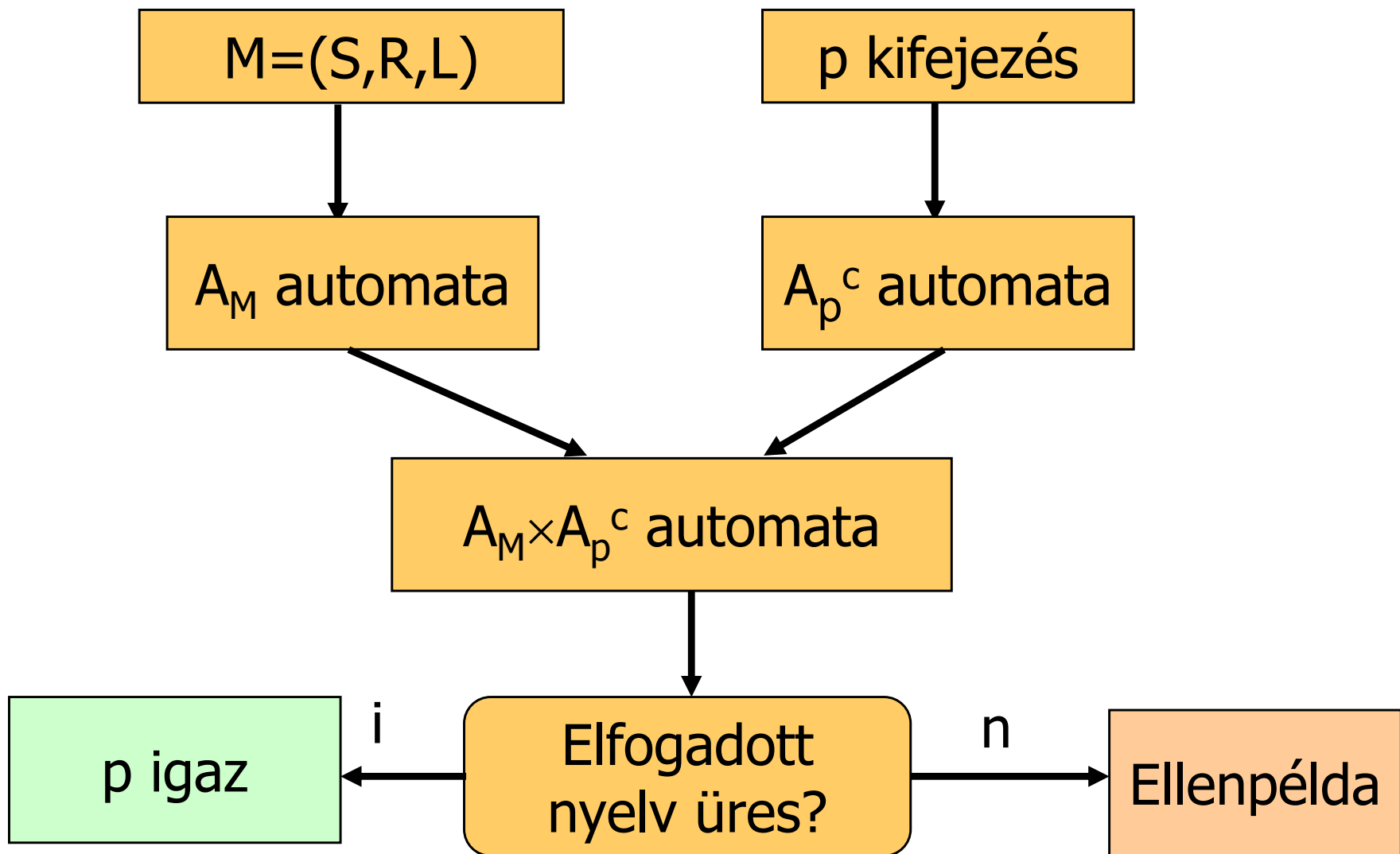
- Folyamatosan működő rendszerek:

- Automaták végtelen hosszúságú szavakon

- Büchi elfogadási kritérium: gyakorlatilag cikluskeresésre vezet

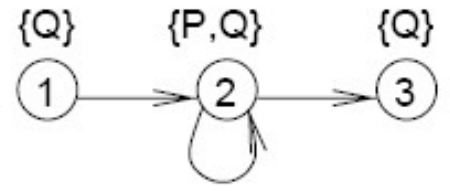


Az automata alapú modell ellenőrzés

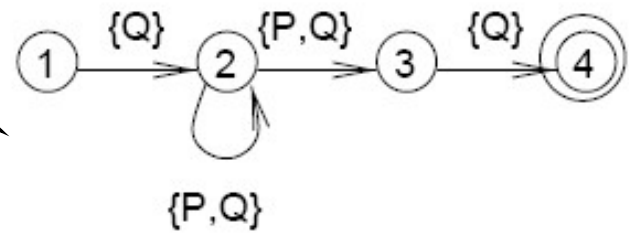


Példa: $F P \wedge G Q$ ellenőrzése

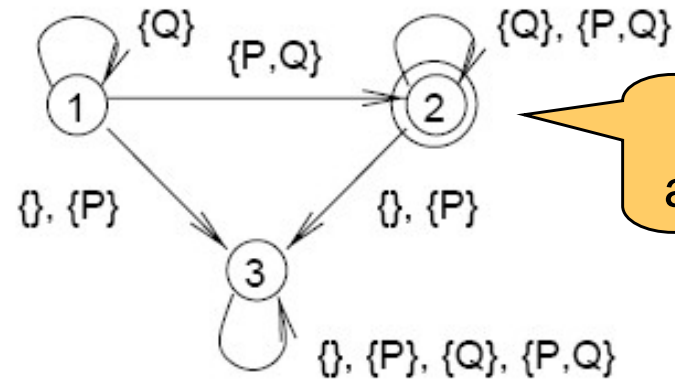
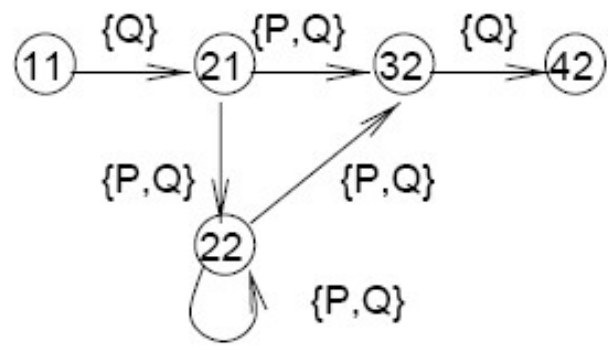
M modell



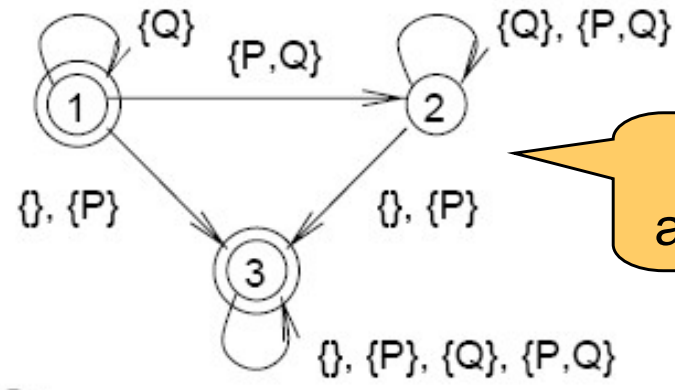
A_M automata



A_M és A_p^c automaták szinkron szorzata



A_p automata



A_p^c automata

Nincs elfogadó állapot:
Nincs ellenpélda

„On-the-fly” modell ellenőrzés

- 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 konkurens, magasabb szintű modellből való származtatás esetén)
- Modell és tulajdonság automata szimultán előállítás

A_M konstruálása M alapján

- Címkék áthelyezése a kimenő élekre
- Elfogadó állapot konstruálása és behúzása a végállapotoktól (ahonnan nincs kimenő él)
- Az automata:

$$A_M = (2^{AP}, S \cup \{s_f\}, \{s_0\}, \rho, \{s_f\})$$

ahol az állapotátmeneti reláció:

$$\rho = \{ (s, L(s), t) \mid (s, t) \in R \} \cup \\ \{ (s, L(s), s_f) \mid \text{nincs } t, \text{ hogy } (s, t) \in R \}$$

A_p konstruálása (1)

- A_p automata: csak olyan utak, ahol p igaz
- Ötlet: A **tabló módszerrel** kifejtetni a kifejezéseket
 - Meghatározni, milyen címkézés kell legyen egy **adott állapotban** (felbontás csomópontjai alapján)
 - Meghatározni, mi vonatkozik a **következő állapotból** induló szuffixre (temporális operátorok alapján élek és rákövetkező állapotok létrehozása), és ezt onnan tovább bontani
- **Negált normál formára hozás:**
 - Boole operátorokra: Id. de Morgan szabályok
 - Temporális operátorokra:
 - $\neg(X p) = X (\neg p)$
 - $\neg(p U q) = (\neg p) R (\neg q)$ ahol R a „release” operátor:
 - $p R q = q \wedge (p \vee X (p R q))$

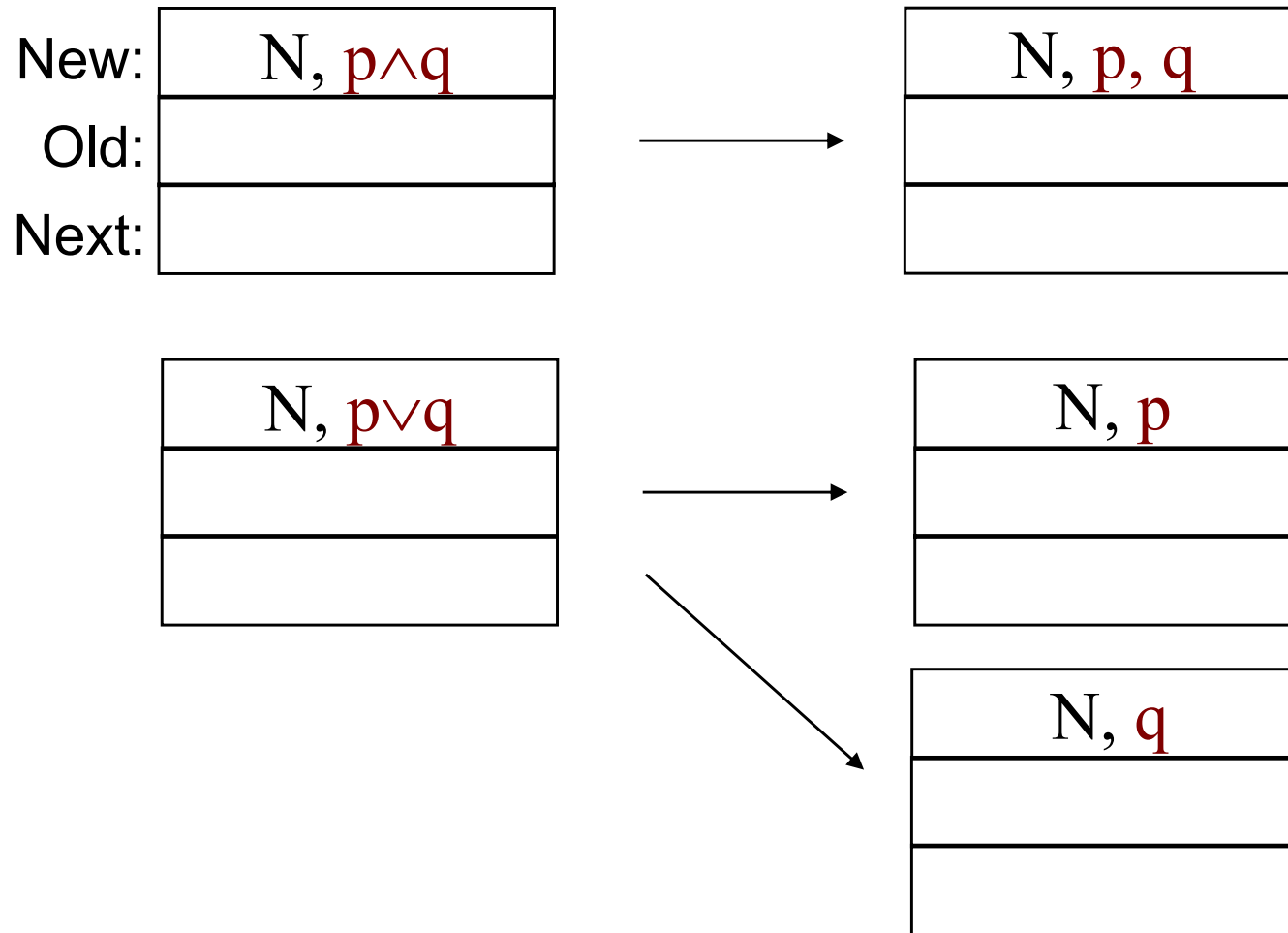
A_p konstruálása (2)

- A táblához használt adatstruktúra (rekord):
 - **New**: felbontandó kifejezés (kifejezéslista)
 - **Old**: felbontott kifejezés (állapot címkét határozza meg)
 - **Next**: következő állapotra vonatkozó kifejezés (éleket határozza meg)

New:	N, p
Old:	
Next:	

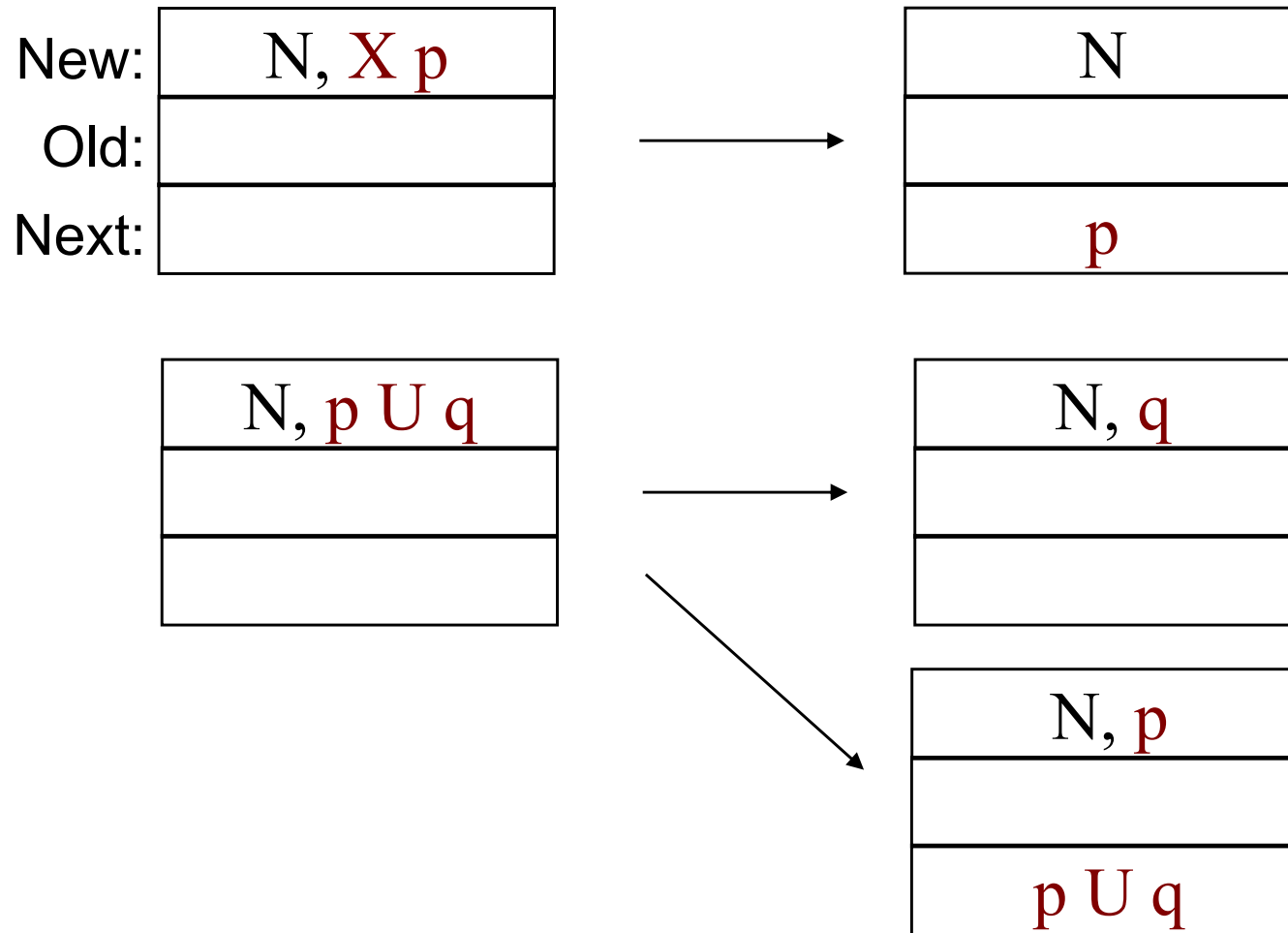
A_p konstruálása (3)

- A felbontási szabályok \wedge és \vee esetén:



A_p konstruálása (4)

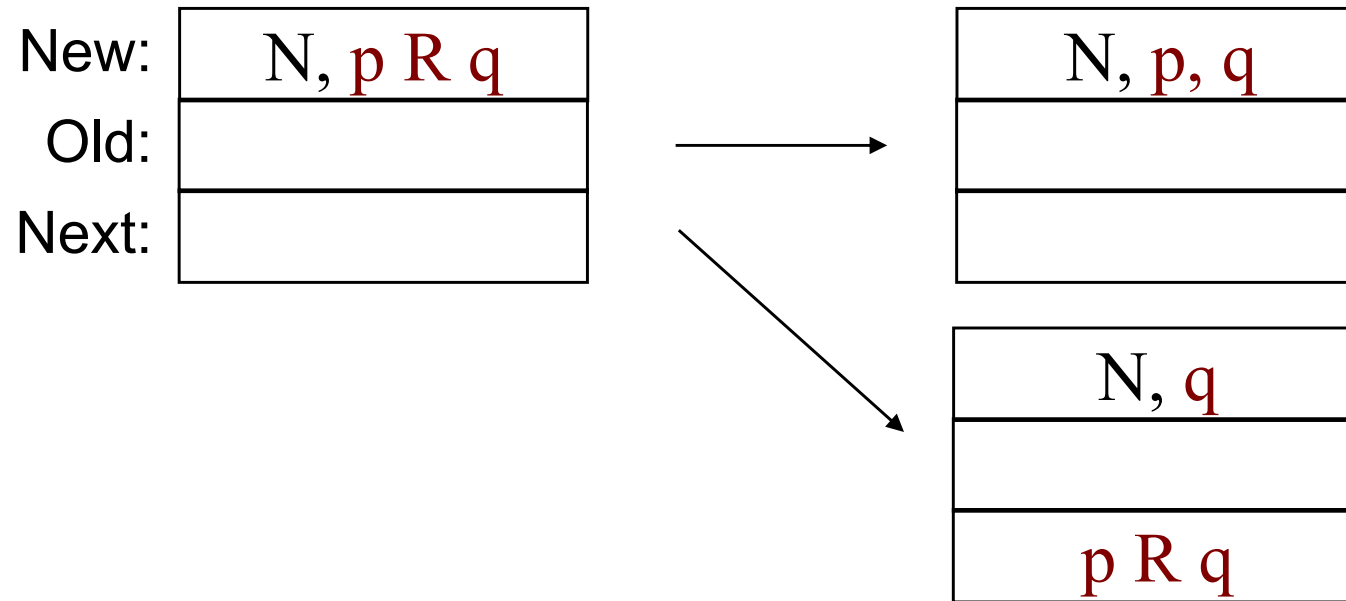
- A felbontási szabályok X és U esetén:



mivel $p \cup q = q \vee (p \wedge X(p \cup q))$

A_p konstruálása (5)

- A felbontási szabályok **R** esetén:

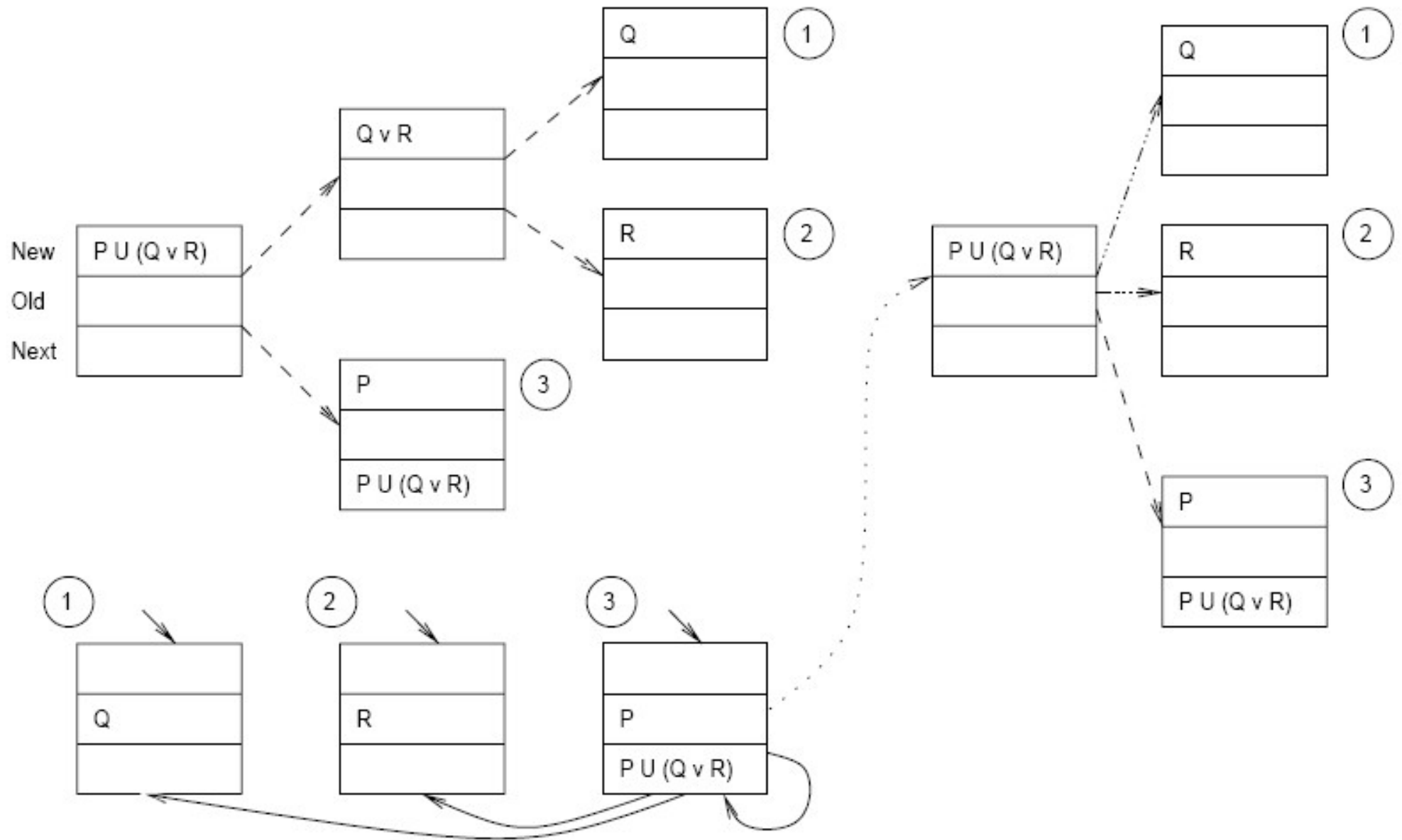


mivel $p R q = q \wedge (p \vee X(p R q))$

A_p konstruálása (6)

- A tábló egy csomópontjából az A_p automata egy állapota lesz, ha:
 - a csomópont New mezőjében csak **atomi kijelentések maradnak** (ezeket átírjuk az Old mezőbe a címkézés meghatározásához)
 - A_p -nek még nincs olyan állapota, aminek Old mezőjében ugyanilyen kijelentések már vannak (egyébként ugyanazt az állapotot kaptuk)
- A_p egy **s** állapotának létrehozásakor:
 - A Next mezőben lévő **q** kifejezés alapján **új felbontás indítása**, a Next mező New-ba való átírásával (**következő állapotra** vonatkozott!)
 - Az így **q**-ból létrejövő **új állapotokhoz** élet kell húzni **s**-ből
- Összegezve:
 - A_p állapotok létrejönnek, ha New tovább nem bontható
 - A_p él létrejön, ha a Next mező alapján bontunk fel kifejezést (következő állapotot határozza meg)

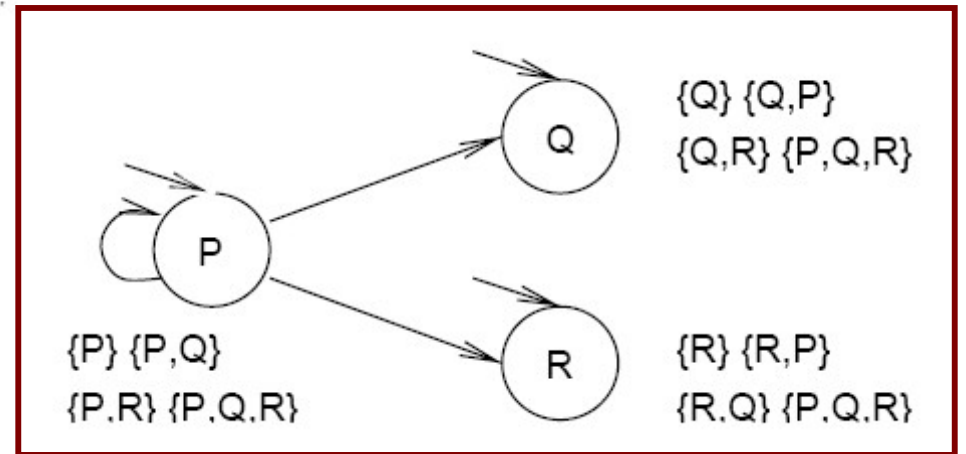
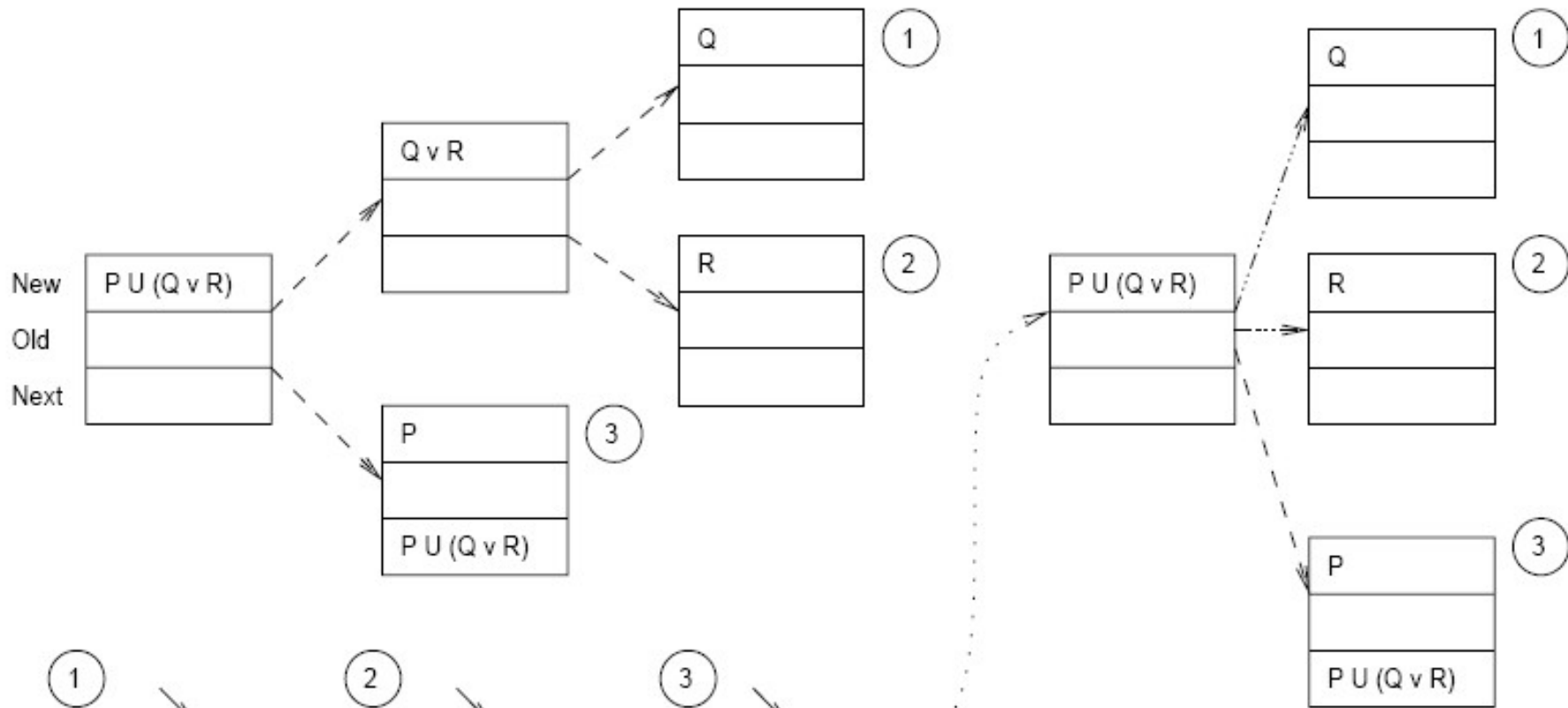
Példa: $P \cup (Q \vee R)$



A_p konstruálása (7)

- További elemek A_p -ben:
 - Kezdőállapot(ok):
 - Az első felbontás alapján
 - Elfogadó állapotok:
 - Ahol a Next mező üres volt (nincs további feltétel)
 - Állapotok címkézése: Az AP olyan részhalmazai, amelyek **kompatibilisek** az Old mezőben lévő kijelentés-halmazzal:
 - minden atomi kijelentés benne, ami az Old-ban is
 - nincs olyan atomi kijelentés benne, aminek negáltja van Old-ban
- mivel minden **megengedett** viselkedést le kell írni!

Példa: $P \cup (Q \vee R)$



Tartalomjegyzék

- Temporális követelmények
- A Hennessy-Milner logika
 - Modellellenőrzés a tabló módszerrel
- **A PLTL lineáris temporális logika**
 - Operátorok
 - Modellellenőrzés automata-elméleti alapon
 - **A modellellenőrzés komplexitása**

A PLTL modell ellenőrzés komplexitása

- $M=(S,R,L)$ struktúrán a p kifejezés modell ellenőrzésének worst case időkomplexitása:
 $O(|S|^{2 \times 2^{|p|}})$
 - Itt $|S|$ az állapotok száma, $|p|$ az operátorok száma
 - $|S|^2$ adódik a **modell automata** átmeneteinek számából (max. négyzetes, átlagos esetben csak lineáris)
 - $2^{|p|}$ adódik a **tulajdonság automata** átmeneteinek számából: a tabló során ennyi a rész-kifejezések (mint lehetséges felbontandó kifejezések) maximális száma
 - Az exponenciális komplexitás riasztó, de
 - általában a PLTL kifejezés rövid,
 - a komplexitás S mérete miatt lesz nagy
 - $O(|S|^{2 \times 2^{|p|}})$ a **szinkron szorzat automata** állapotterének méretéből adódik (elfogadó állapot keresése)

SPIN modell ellenőrző

The screenshot shows the SPIN Linear Time Temporal Logic Formulae editor. The title bar reads "Linear Time Temporal Logic Formulae". The "Formula:" field contains the expression $\langle \rangle [] \text{oneLeader}$. The "Operators:" field shows buttons for $[]$, $\langle \rangle$, U , \rightarrow , and , or , and no . The "Property hold" section has two radio buttons: "All Executions (desired behavior)" (selected) and "No Executions (error behavior)". The "Notes" section is partially visible. The "Symbols" section contains the following definitions:

```
#define elected          (nr_leaders > 0)
#define noLeader      (nr_leaders == 0)
#define oneLeader     (nr_leaders == 1)
|
```

Three callout boxes provide additional information:

- PLTL operátorok jelölése:**
F megfelelője $\langle \rangle$
G megfelelője $[]$
(X operátor nem található)
- Útvonalak kezelése**
- Címkék a modell állapotváltozói segítségével definiálhatók**