

Követelmények formalizálása: Temporális logikák

dr. Majzik István

BME Méréstechnika és Információs Rendszerek Tanszék

Motiváció

Mire kellenek a temporális logikák?

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)

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

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

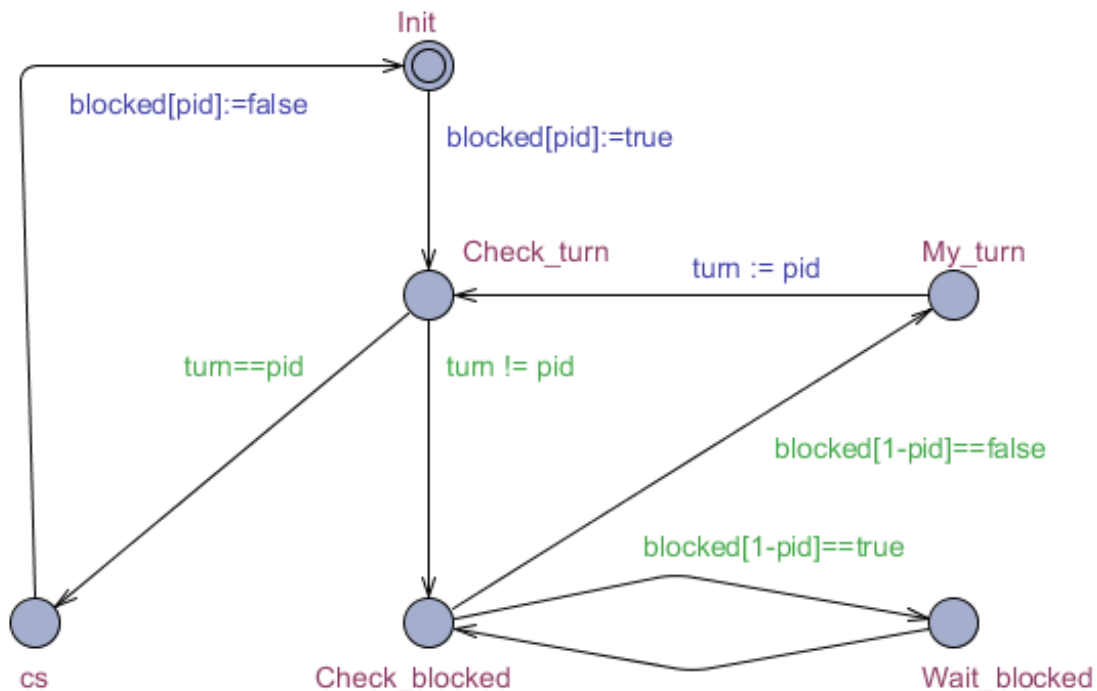
Helyes-e ez az algoritmus?

A modell UPPAAL-ban

Deklarációk:

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

A P automata (template),
pid paraméterrel:



Kihasztnált modellezési lehetőségek:

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

```
while (true) {                                P0  
  blocked0 = true;  
  while (turn!=0) {  
    while (blocked1==true) {  
      skip;  
    }  
    turn=0;  
  }  
  // Critical section  
  blocked0 = false;  
  // Do other things  
}
```

Ellenőrizendő követelmények

- Kölcsönös kizárás:
 - Egyszerre csak az egyik résztvevő lehet a kritikus szakaszban
- Lehetséges az elvárt viselkedés:
 - P0 egyáltalán **beléphet** a kritikus szakaszba
 - P1 egyáltalán **beléphet** a kritikus szakaszba
- Nincs kiéheztetés:
 - P0 **mindenképpen** be fog lépni a kritikus szakaszba
 - P1 **mindenképpen** be fog lépni a kritikus szakaszba
- Holtpontmentesség:
 - Nem alakul ki kölcsönös várakozás (leállás)

Hogyan ellenőrizhetjük a követelményeket?

- A megvalósítás tesztelésével
 - Létre tudunk-e hozni minden lehetséges végrehajtást lefedő teszt eseteket?
 - A problémás esetek figyeléséhez külön ellenőrző kell
 - A hiba már csak az implementáció után derülhet ki, drágán javítható
- Modellezéssel és szimulációval
 - Tudunk-e szimulálni minden lehetséges végrehajtást?
 - A problémás esetek detektálása nagy odafigyelést igényel
 - A hibák viszont olcsóbban javíthatók modell szinten
- Modellezéssel és az állapottér teljes ellenőrzésével
 - Szisztematikus algoritmus kell az állapottér teljes felvételéhez
 - Legyen automatikus a követelmények teljesülésének figyelése is
 - Ehhez egy olyan nyelv szükséges, amikkel a követelmények leírhatók
 - Általános módszer adható a követelmények modell alapú ellenőrzésére

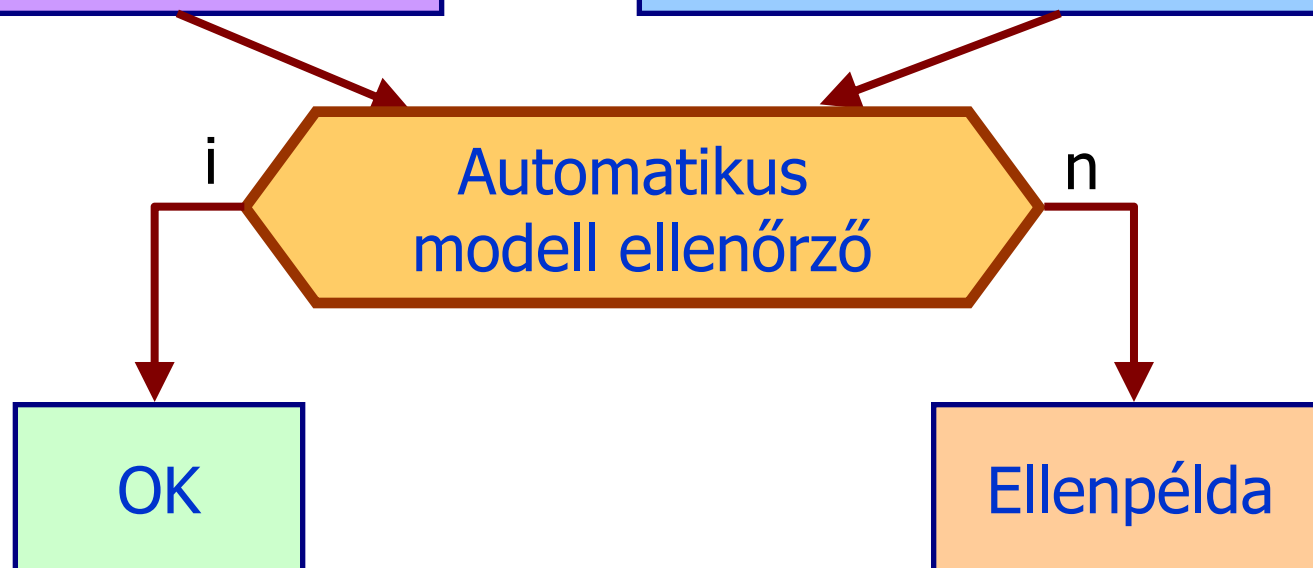
Mit szeretnénk elérni?

- Alacsony szintű, vagy
- magasabb szintű, vagy
- mérnöki modellek

Automatikusan
ellenőrizhető,
precíz követelmények

Rendszer modellje

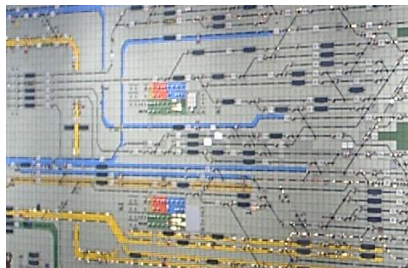
Követelmény megadása



Követelmények formalizálása

Milyen jellegű követelményeket formalizálunk?

- Verifikáció: Modell \leftrightarrow Sokféle követelmény
 - Funkcionális: Logikailag helyes a működés <- Most ez a célunk
 - Extra-funkcionális: Teljesítmény, megbízhatóság, ...
- Célkitűzés: **Állapotok elérhetőségének ellenőrzése**
 - **Állapotok bekövetkezését és sorrendjét vizsgáljuk**
 - Eljuthatunk-e kedvező állapotba? -> Élő jellegű követelmények
 - Elkerüljük-e a kedvezőtlen állapotokat? -> Biztonsági követelményekEzek az állapottér teljes felderítésével ellenőrizhetők!
 - **Állapotok lokális tulajdonságára hivatkozunk**
 - Állapot neve, állapotváltozók értéke
- Fontos állapot alapú, eseményvezérelt rendszerekben



„Biztonsági” jellegű követelmények

- **Veszélyes helyzetek elkerülését írják elő:**
 - „Minden állapotban kisebb a nyomás a kritikusnál.”
 - „A prégép csak lecsukott ajtó mellett üzemelhet.”
- **Univerzális kvantor az elérhető állapotokon:**
 - „Minden elérhető állapotban igaz, hogy ...”
 - Invariánst fogalmazznak meg
- **Ha egy állapotsorozat nem teljesíti:
nem is egészíthető ki úgy, hogy teljesítse**
- **Informatikai példák:**
 - Holtpontmentesség
 - Kölcsönös kizárás
 - Adatbiztonság (egyik állapotban sincs jogosulatlan hozzáférés)

„Élő” jellegű követelmények

- **Kívánatos helyzetek elérését írják elő**
 - „Az indítás után a présgép kiadja az elkészült terméket.”
 - „A zavarás után a folyamat visszakerül stabil állapotba.”
- **Egzisztenciális kvantor az elérhető állapotokon**
 - „Létezik (elérhető) olyan állapot, hogy ...”
- **Ha egy állapotsorozat nem teljesíti:
elvileg kiegészíthető úgy, hogy teljesítse**
- **Informatikai példák:**
 - Adott kérés kiszolgálása megtörténik
 - Elküldött üzenet megérkezik
 - A processz kiszámítja a várt értéket

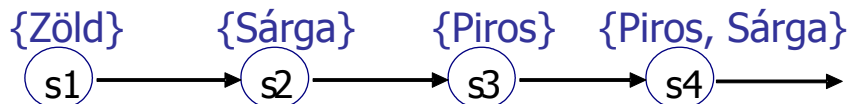
Az elérhetőségi követelmények leíró nyelve

- **Elérhetőség:**
Állapotok **bekövetkezési sorrendjére vonatkozik**
 - **Bekövetkezési sorrend:** Megfeleltethető a **logikai időnek**
 - Jelen időpillanat: Aktuális állapot
 - Következő időpillanat(ok): Rákövetkező állapot(ok)
 - **Temporális (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”, ...

Temporális logikák osztályozása

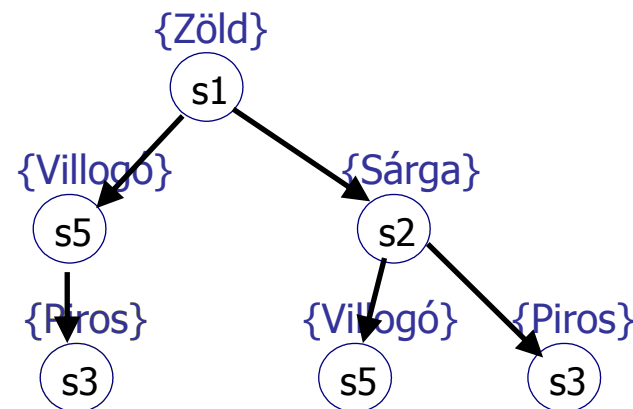
- **Lineáris:**

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



- **Elágazó:**

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



Temporális logikák

Hol értelmezhetjük a temporális logikákat?

- Célkitűzés: Állapottér vizsgálata

Matematikai modell: Kripke-struktúra

- Állapotok lokális tulajdonságait címkézéssel vezettük be

$KS = (S, R, L)$ és AP , ahol

$AP = \{P, Q, R, \dots\}$ atomi kijelentések halmaza (domén-specifikus)

$S = \{s_1, s_2, s_3, \dots, s_n\}$ állapotok halmaza

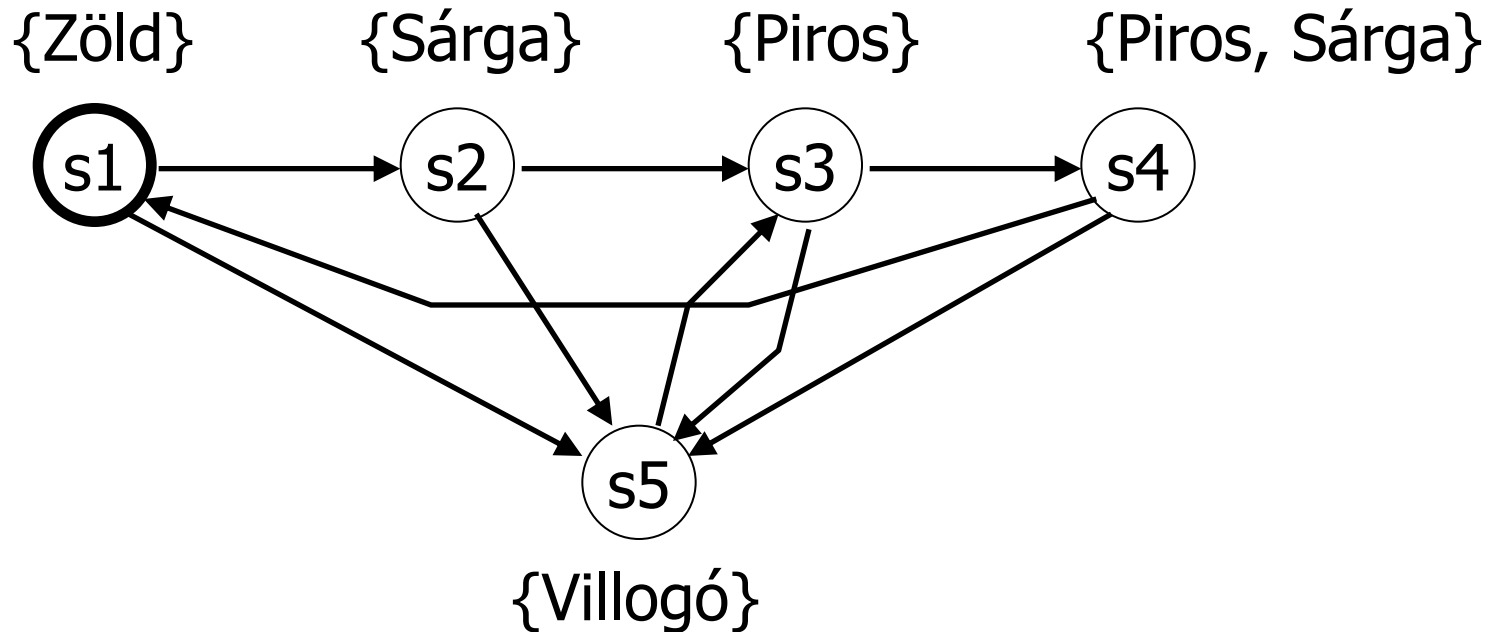
$R \subseteq S \times S$: állapotátmeneti reláció

$L: S \rightarrow 2^{AP}$ állapotok címkézése atomi kijelentésekkel

Kripke-struktúra példa

Közlekedési lámpa vezérlője

- $AP = \{\text{Zöld}, \text{Sárga}, \text{Piros}, \text{Villogó}\}$
- $S = \{s1, s2, s3, s4, s5\}$

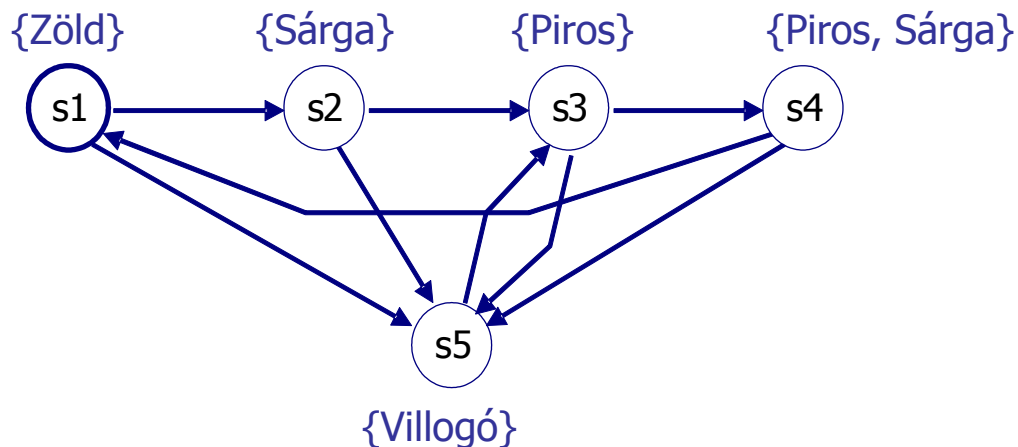


Lineáris idejű temporális logika: PLTL

Lineáris idejű temporális logikák

- A Kripke-struktúra egy-egy útvonalán értelmezhetők
 - Egy-egy „lefutás” (pl. egy konkrét bemenet hatására)

A modell (KS):



Egy útvonal (állapotsorozat):



PLTL: Egy lineáris idejű temporális logika

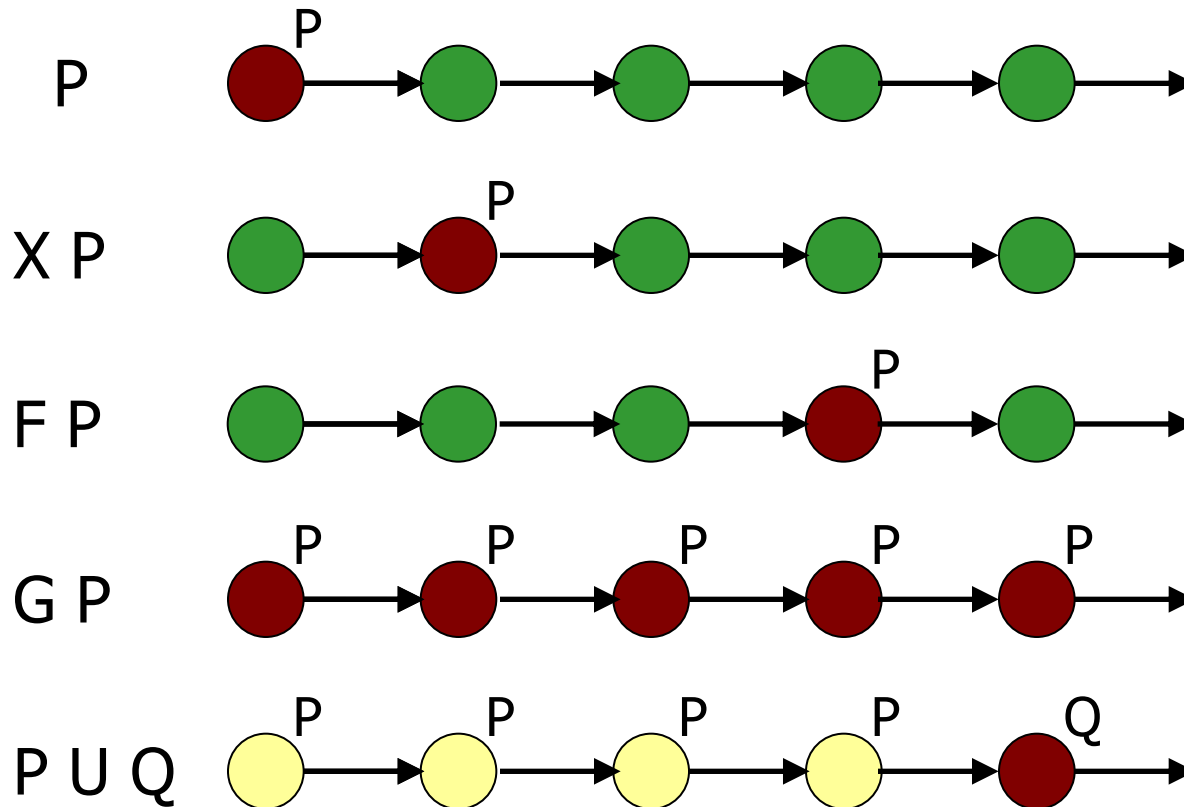
PLTL (Propositional Linear Time Temporal Logic)

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:
 - $F p$: „Future p ”, egy elérhető állapotban igaz lesz p
 - $G p$: „Globally p ”, minden elérhető állapotban igaz lesz p
 - $X p$: „neXt p ”, a következő állapotban igaz lesz p
 - $p U q$: „ p Until q ”, egy elérhető állapotban igaz lesz q , és addig minden állapotban igaz p

PLTL temporális operátorok

Kripke-struktúra egy útvonalán (idővonalán):



PLTL példák I.

- $p \Rightarrow Fq$

Ha a kiindulási állapotra p igaz, akkor valamikor (egy későbbi állapotra) q is igaz lesz.

- $G(p \Rightarrow Fq)$

Minden állapotra fennáll,
hogy ha p igaz, akkor valamikor q is igaz lesz.
Pl. bármikor kiadott p kérésre q válasz érkezik.

- $p \text{ U } (q \vee r)$

Itt p igaz, amíg q vagy r igaz nem lesz.
Pl. az első állapotból kiadott folyamatos p kérést
 q válasz vagy r elutasítás követ.

- $(p \wedge G(p \Rightarrow Xp)) \Rightarrow Gp$

A matematikai indukció leírása: Mindig teljesül

PLTL példák II.

- GF p

Minden állapotra igaz, hogy ebből indulva valamikor p igaz lesz.

- Nem találunk olyan állapotot, ami után p tulajdonságú állapot ne lenne elérhető.
- Pl. minden állapotból a p tulajdonságú kezdőállapotba vihető a rendszer.

- FG p

Valamikor olyan állapotba kerül a rendszer, hogy azontúl p folyamatosan igaz lesz.

- Pl. kezdeti tranziens után a p tulajdonságú üzemi állapotokba kerül a rendszer.

PLTL nyelv formális kezelése

- Az eddigiek csak informális bevezetést adtak
Kérdések vetődhetnek fel:
 - $F p$ igaz-e, ha p rögtön az első állapotban igaz?
 - $p U q$ igaz-e, ha q az első állapotban igaz?
- Az automatikus ellenőrzést is lehetővé tevő precíz megadáshoz szükséges:
 - Formális szintaxis szabályok:
Mik az érvényes PLTL kifejezések?
 - Formális szemantika szabályok:
Adott modellen mikor igaz egy PLTL kifejezés?

PLTL formális szintaxis

Az érvényes PLTL kifejezések halmaza a következő szabályokkal képezhető:

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

Operátorok precedenciája növekvő sorrendben:

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

„Kimaradt” operátorok

- **true** minden állapotra igaz („beépített”)
false egy állapotra sem igaz

- $p \vee q$ jelentése $\neg((\neg p) \wedge (\neg q))$

$p \Rightarrow q$ jelentése $(\neg p) \vee q$

$p \equiv q$ jelentése $(p \Rightarrow q) \wedge (q \Rightarrow p)$

- $F p$ jelentése $\text{true} \cup p$

$G p$ jelentése $\neg F(\neg p)$

- „Mielőtt” operátor (before):

$p \text{ WB } q = \neg((\neg p) \cup q)$ (weak before)

$p \text{ B } q = \neg((\neg p) \cup q) \wedge F q$ (strong before)

Informálisan:

Nem igaz, hogy nem fordul elő p a q előtt

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
- $M, \pi \models p$ jelöli:
az M modellben a π útvonalon igaz p

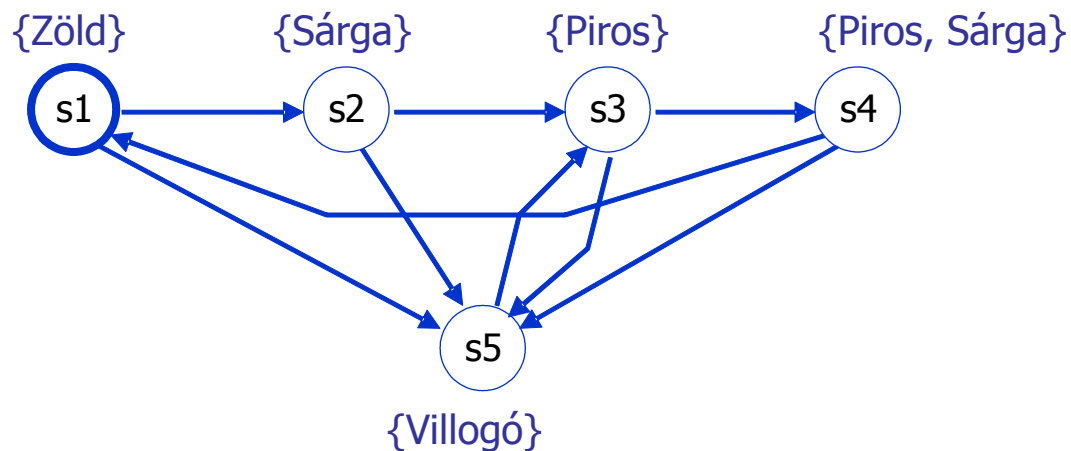
PLTL szemantika

A szintaxis szabályok alapján képzett kifejezésekhez megadható a formális szemantika:

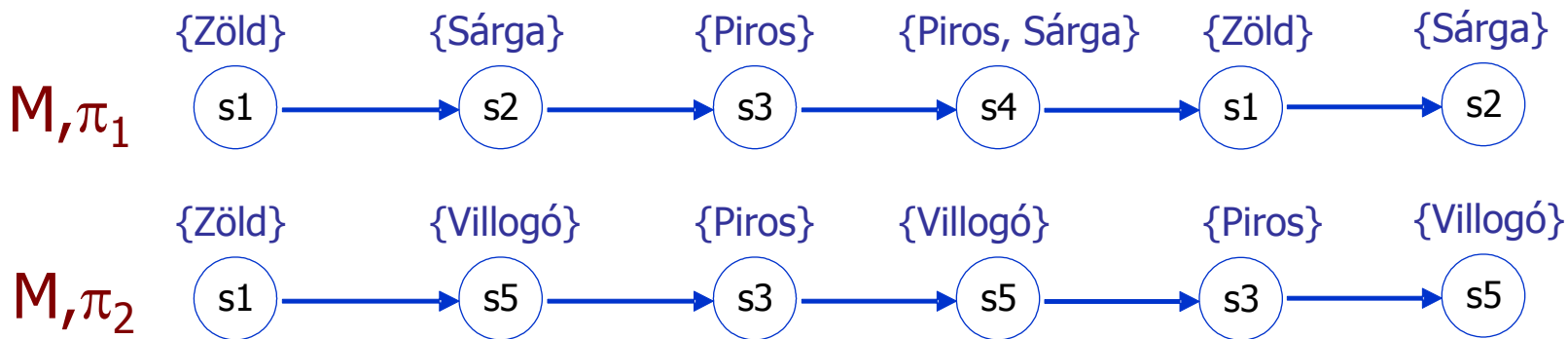
- **L1:** $M, \pi \models P$ a.cs.a. $P \in L(s_0)$
- **L2:** $M, \pi \models p \wedge q$ a.cs.a. $M, \pi \models p$ és $M, \pi \models q$
 $M, \pi \models \neg q$ a.cs.a. $M, \pi \models q$ nem igaz.
- **L3:** $M, \pi \models (p \cup q)$ a.cs.a.
 $\exists j \geq 0 : (\pi^j \models q \text{ valamint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

PLTL kifejezések értelmezése: Példák

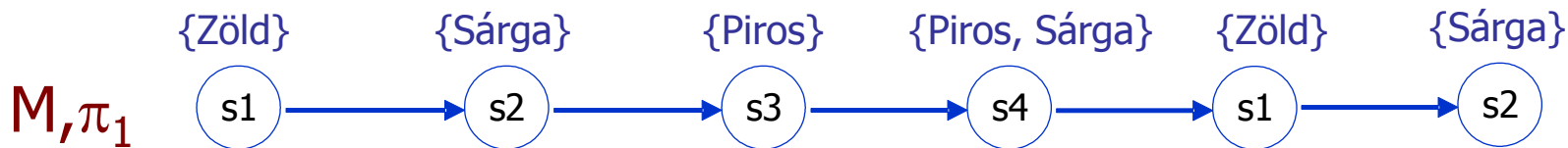
- **M** Kripke-struktúra:



- Útvonalak:

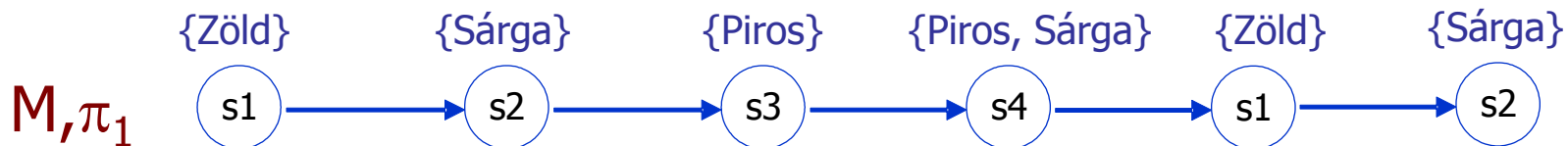


Példák (folytatás)

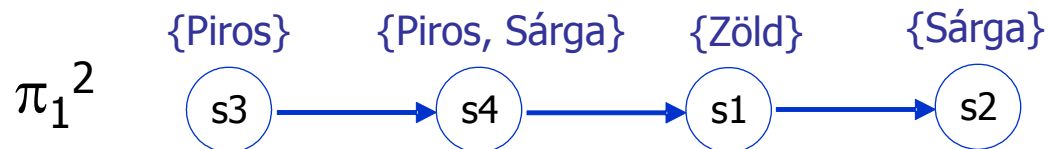


- $M, \pi_1 \models \text{Zöld}$, mert Zöld a kezdőállapot címkeje
- $M, \pi_1 \not\models \text{Piros}$ nem teljesül M, π_1 esetén, mert nem Piros a kezdőállapot címkeje
- $\text{Zöld} \cup \text{Piros}$ nem igaz M, π_1 esetén, mert a Sárga közbeszól
- $M, \pi_1 \models \text{F Piros}$, mert elérhető olyan állapot, ahol Piros a címke, Pontosabban: Van olyan szuffix, amely esetén Piros a kezdőállapot címkeje.

Példák (folytatás)



- $M, \pi_1 \models F(\text{Piros} \cup \text{Zöld})$ igaz,
mert van olyan szuffix,
amire teljesül a $(\text{Piros} \cup \text{Zöld})$:



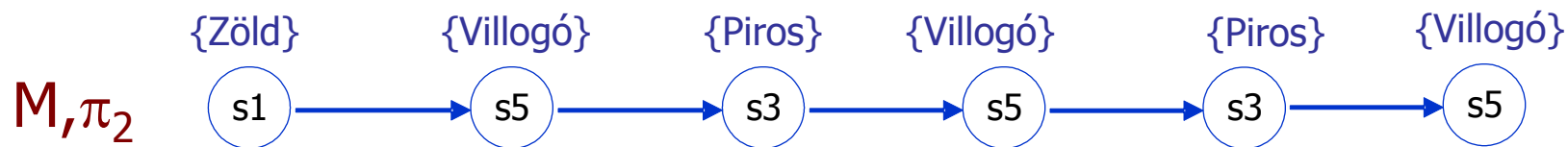
Példák (folytatás)



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



Példák (folytatás)



- $M, \pi_2 \models X F (XX \text{ Piros})$, mert az első szuffixe:



$F (XX \text{ Piros})$ teljesül, mert π_2^1 -nek van olyan szuffixe, ahol $XX \text{ Piros}$ igaz:



Követelmények formalizálása: Példa

Adott egy klímaberendezés, aminek a következő üzemmódokat kell biztosítania:

$AP = \{ \text{Kikapcsolva, Bekapcsolva, Elromlott, GyengénHűt, ErősenHűt, Fűt, Szellőztet} \}$

- Egy-egy állapothoz több címke tartozhat!
- A követelmény formalizálás fázisában a teljes viselkedést még nem ismerjük (azt a követelményeknek megfelelően kell majd megterveznünk)

Példa (folytatás)

$AP = \{\text{Kikapcsolva, Bekapcsolva, Elromlott, GyengénHűt, ErősenHűt, Fűt, Szellőztet}\}$

- A klímát be fogják kapcsolni:
 F (Bekapcsolva)
- A klíma előbb-utóbb mindig elromlik:
 $G F$ (Elromlott)
- Ha a klíma elromlik, mindig megjavítják:
 $G (\text{Elromlott} \Rightarrow F (\neg \text{Elromlott}))$
- Ha a klíma elromlott, nem fűthet:
 $G (\neg (\text{Elromlott} \wedge \text{Fűt}))$

Példa (folytatás)

AP={Kikapcsolva, Bekapcsolva, Elromlott, GyengénHűt, ErősenHűt, Fűt, Szellőztet}

- A klíma csak úgy romolhat el, ha be volt kapcsolva:

$$G (X \text{ Elromlott} \Rightarrow \text{Bekapcsolva})$$

- A fűtési fázis befejezésekor szellőztetni kell:

$$G ((\text{Fűt} \wedge X(\neg \text{Fűt})) \Rightarrow X (\text{Szellőztet}))$$

és ha elromlott?

$$G ((\text{Fűt} \wedge X(\neg \text{Fűt})) \Rightarrow X (\text{Szellőztet} \vee \text{Elromlott}))$$

- Szellőztetés után mindaddig nem hűthet erősen, míg egy gyenge hűtéssel nem próbálkozott:

$$G ((\text{Szellőztet} \wedge X(\neg \text{Szellőztet})) \Rightarrow X(\neg \text{ErősenHűt} \cup \text{GyengénHűt}))$$

PLTL összefoglalás

- Követelmények megfogalmazása
- Temporális logikák
- Lineáris idejű temporális logikák
- PLTL
 - Operátorok (intuitív)
 - Formális szintaxis
 - Formális szemantika
- PLTL kifejezések értelmezése
- Követelmények formalizálása