Követelmények formalizálása: Temporális logikák

dr. Majzik István
BME Méréstechnika és Információs Rendszerek Tanszék

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
}
```

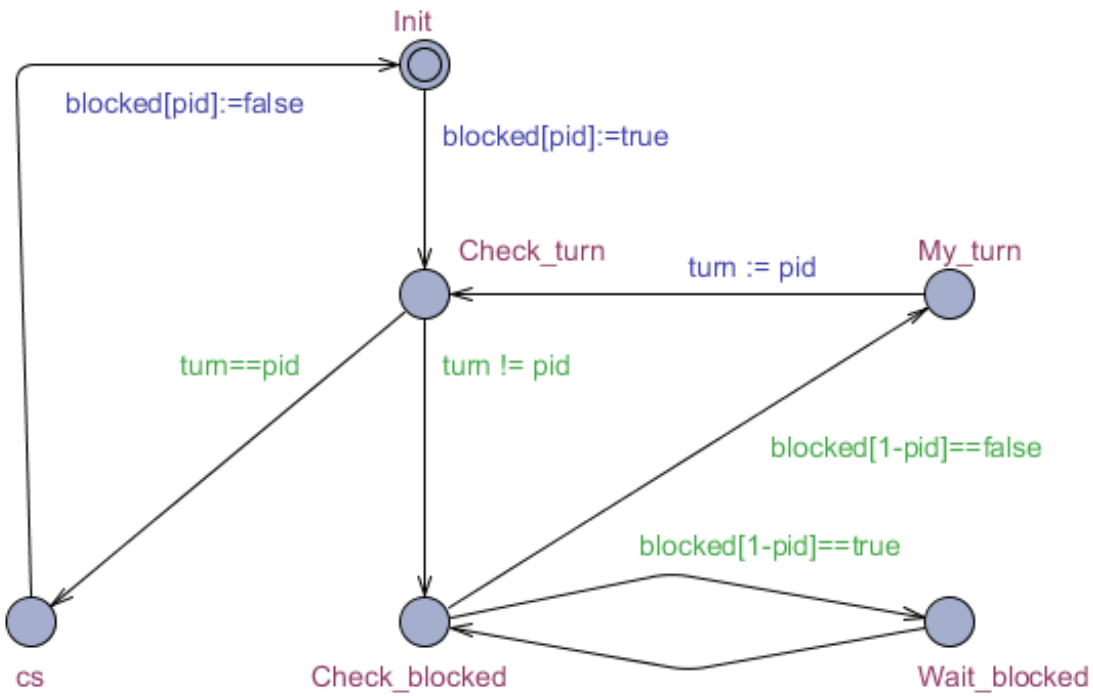
Helyes-e ez az algoritmus?

A modell UPPAAL-ban

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

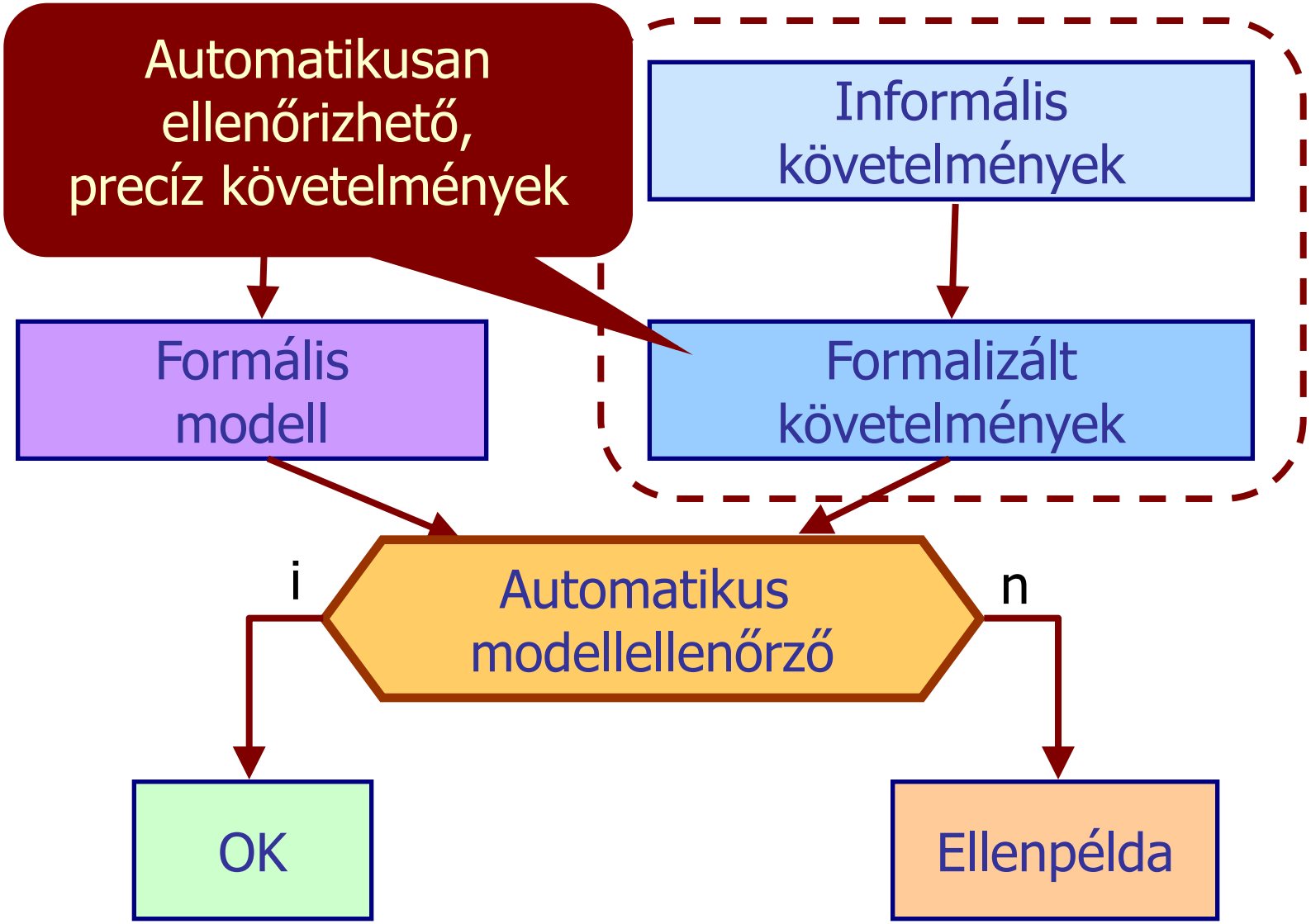
```
while (true) {
    blocked0 = true;
    while (turn!=0) {
        while (blocked1==true) {
            skip;
        }
        turn=0;
    }
    // Critical section
    blocked0 = false;
    // Do other things
}
```

Ellenőrizendő követelmények

- Kölcsönös kizárás:
  - Soha nem lehet egyszerre P0 és P1 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 formalizálhatjuk ezeket?

Mit szeretnénk elérni?



Követelmények rögzítése és formalizálása

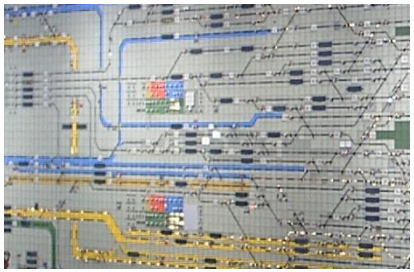
Mit tudunk a jellegzetes követelményekről (kritikus rendszerekben)?

Mit érdemes formalizálni?

Milyen jellegű követelményeket formalizálunk?

- Verifikáció: Modell <-> Sokféle követelmény
  - Funkcionális: Logikailag helyes a működés <- Most ez a célunk
  - Extra-funkcionális: Teljesítmény, megbízhatóság, … <- Később
- Célkitűzés: Állapotok elérhetőségének ellenőrzése
  - Rendszer(modell): Állapotok lokális tulajdonságait ismerjük
    - Állapot neve, állapotváltozók értéke, működési mód, …
  - Követelmények: Állapotok bekövetkezési lehetőségeit vizsgáljuk
    - Elkerüljük-e a kedvezőtlen állapotokat?
    - Eljuthatunk-e kedvező állapotba?

    Ezek az állapottér teljes felderítésével ellenőrizhetők!
- Fontos állapot alapú, eseményvezérelt rendszerekben



Szöveges követelmények kezelése

- Tipikus követelmény megadás: Szöveges forma

Ha az `alarm` be van kapcsolva és `alert` történik, akkor a `safety` kimenet legyen igaz, amíg az `alarm` be van kapcsolva.

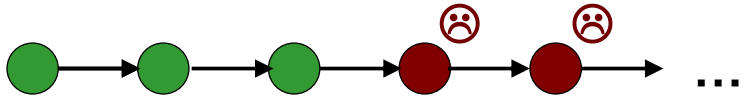
If the `switch` is turned to `AUTO`, and the `light intensity` is `LOW` then the `headlights` should stay or turn immediately `ON`, afterwards the `headlights` should continue to stay `ON` in `AUTO` as long as the `light intensity` is not `HIGH`.

  - Egyértelmű a szöveges leírás?
  - Nehezen áttekinthető a struktúra (feltétel, következmény, kimenet, időzítési viszonyok, …)
- Mit lehet formalizálni?

Jellegzetes követelmény kategóriák

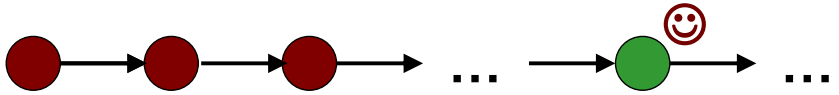
Biztonsági követelmények

- Veszélyes helyzetek elkerülését írják elő:
  - „Minden állapotban kisebb a nyomás a kritikusnál.”
  - „Nincs több processz a kritikus szakaszban.”
- Univerzális tulajdonság az elérhető állapotokon:
  - „Minden elérhető állapotban igaz, hogy …”
  - Invariáns tulajdonság
  - Ha egy állapotsorozat nem teljesíti: Nem is egészíthető ki úgy, hogy teljesítse

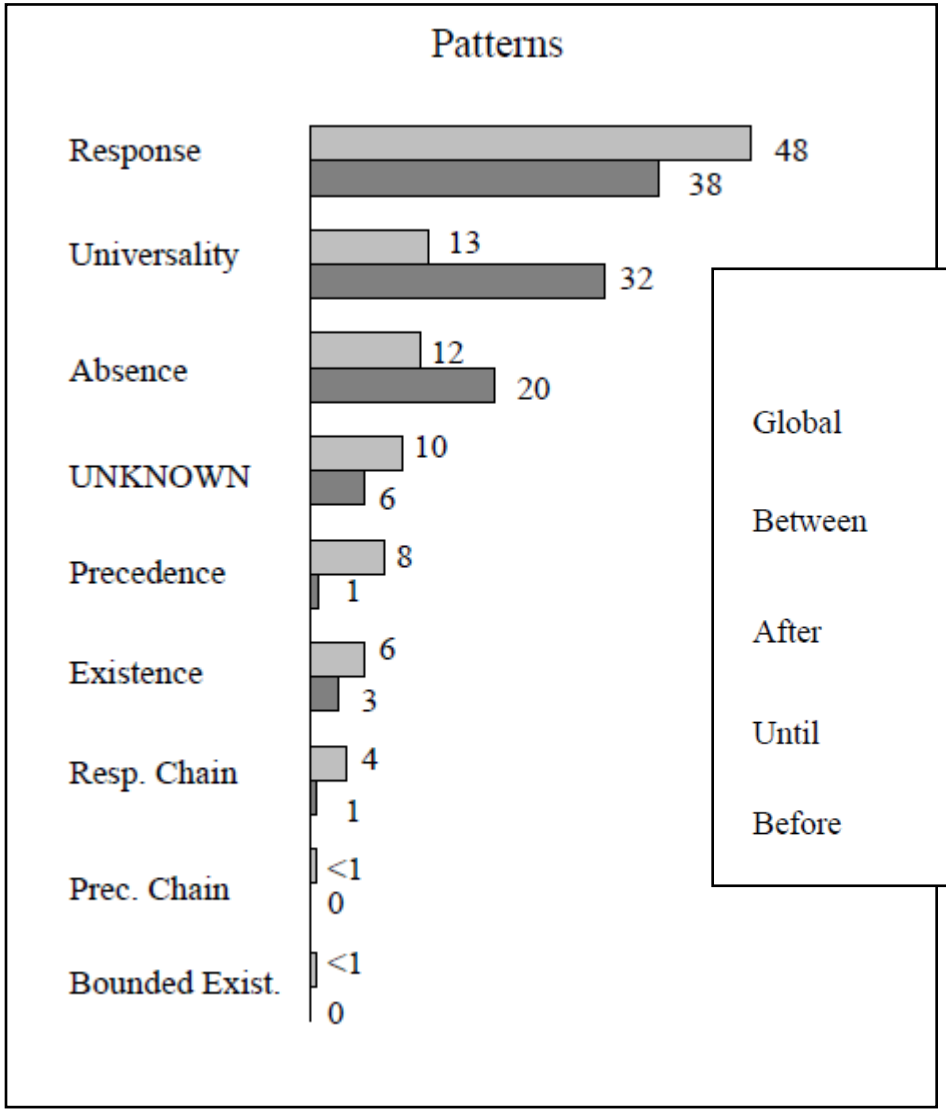


Élő jellegű követelmények

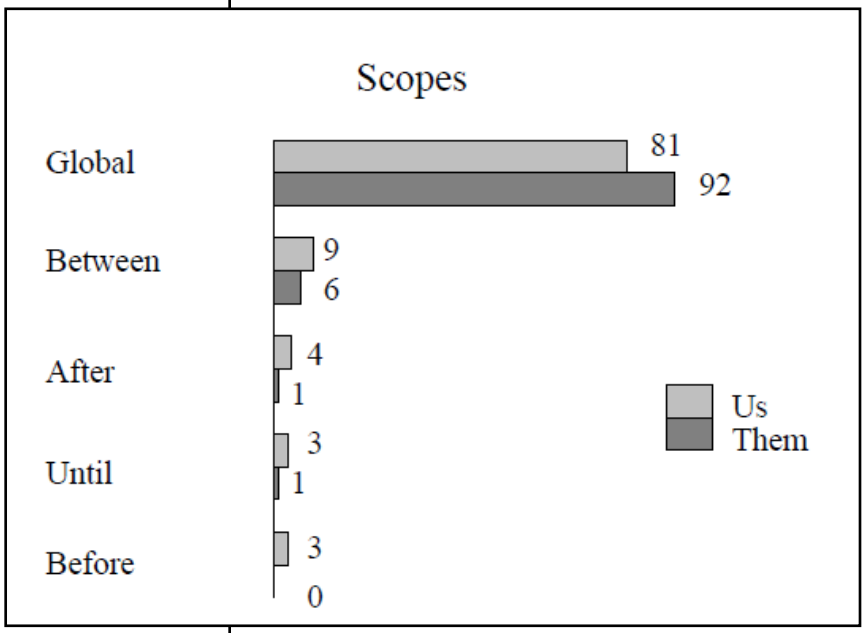
- Kívánatos helyzetek elérését írják elő
  - „Az indítás után a présgép kiadja a kész terméket.”
  - „A kérés kiszolgálása előbb-utóbb megtörténik.”
- Egzisztenciális tulajdonság az elérhető állapotokon
  - „Létezik (elérhető) olyan állapot, hogy …”
  - Bekövetkezést ír elő
  - Ha egy állapotsorozat nem teljesíti: elvileg kiegészíthető úgy, hogy teljesítse



Egy felmérés eredménye: Követelmény minták



A követelmények jelentős hányada meghatározott mintákra illeszkedik

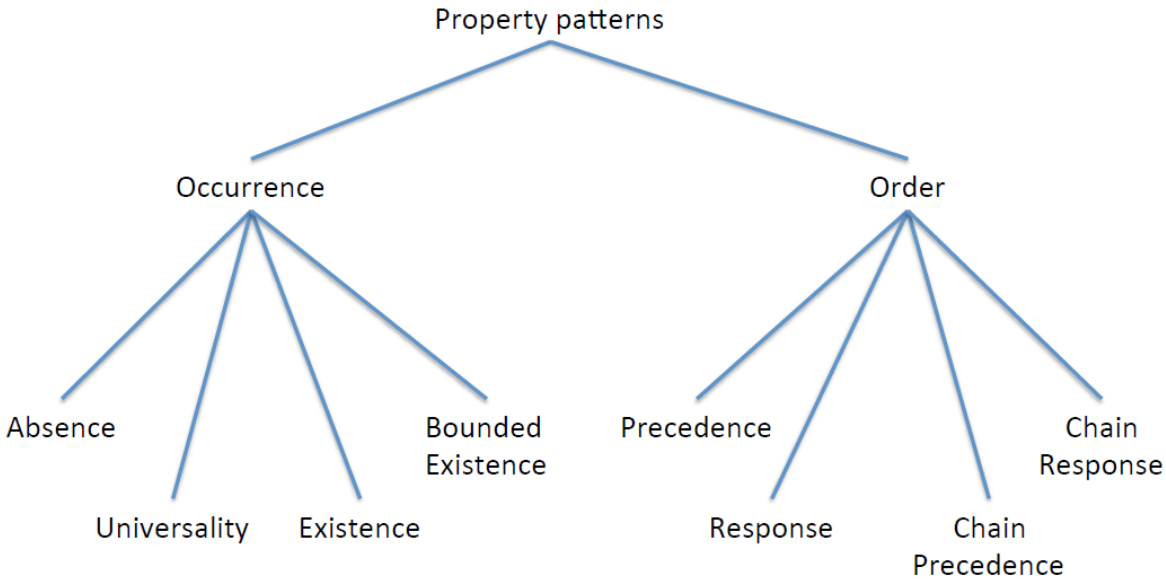


- Minták százalékos megoszlása
- 2 fejlesztői csoport által felvett követelmények

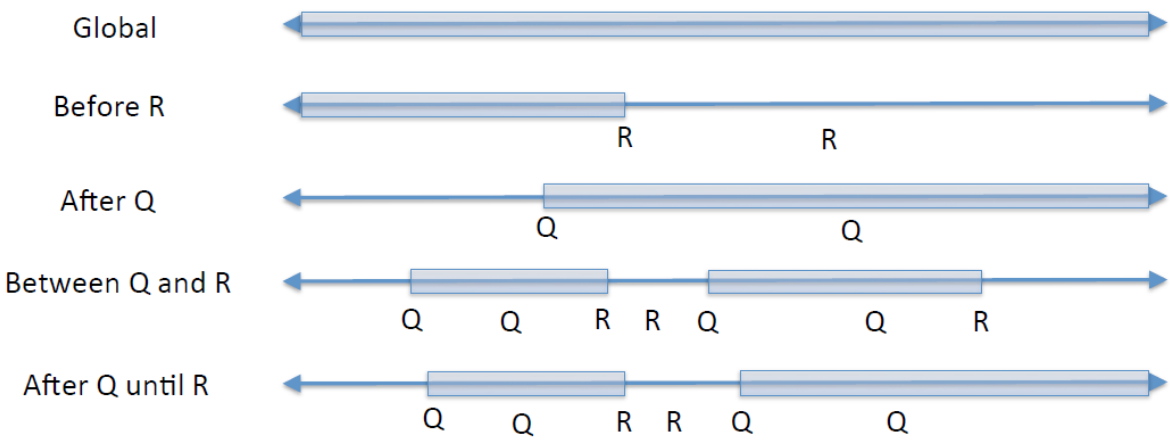
http://patterns.projects.cis.ksu.edu/documentation/patterns.shtml

Követelmény minták csoportosítása

Minták: Sorrendi előírások



Hatókörök: További eseményekhez (adott tulajdonságú állapotokhoz) képest



A minták jelentése (magyarázat)

Occurrence: Adott tulajdonságú állapotok előfordulása

- Absence: A hivatkozott állapot/esemény nem fordul elő.
- Universality: A hivatkozott állapot/esemény folyamatosan előfordul.
- Existence: A hivatkozott állapot/esemény egyszer biztosan előfordul.
- Bounded existence: A hivatkozott állapot/esemény biztosan előfordul „k” alkalommal (létezik „legalább k” és „legfeljebb k” változat is.)

Order: Adott tulajdonságú állapotok sorrendezése

- Precedence: Az egyik állapot/esemény mindenképpen meg kell, hogy előzze a másik állapotot/eseményt.
- Response: Az egyik állapotot/eseményt mindenképpen kell, hogy kövesse egy másik meghatározott állapot/esemény.
- Chain precedence: A Precedence minta általánosítása állapot / esemény sorozatokra.
- Chain response: A Response minta általánosítása állapot / esemény sorozatokra.

Példák a minták megjelenésére

- Response minta Global hatókörben:

A végrehajtás során bármikor,
ha Request kérés következik be, akkor vagy egy Reply
vagy egy Reject válasznak kell következnie.

- Precedence minta After hatókörben:

A NormalMode állapot bekövetkezése után
a ResourceGranted állapot csak akkor következhet be,
ha ezt megelőzi ResourceRequest állapot.

Egy jellegzetes megoldás

- Szöveges követelményminták használata*
  - Kitölthető, paraméterezhető minták komponálása
  - Struktúrát áttekinthetőbbé teszi
  - A mintákhoz formális szemantika rendelhető

When …, Ensure …

…, after … occurs

… = …

… within …

When alarm = ON , after alert occurs

Ensure safety within 0.1 [sec]

* Darvas Dániel által készített példák a STIMULUS eszköz alapján

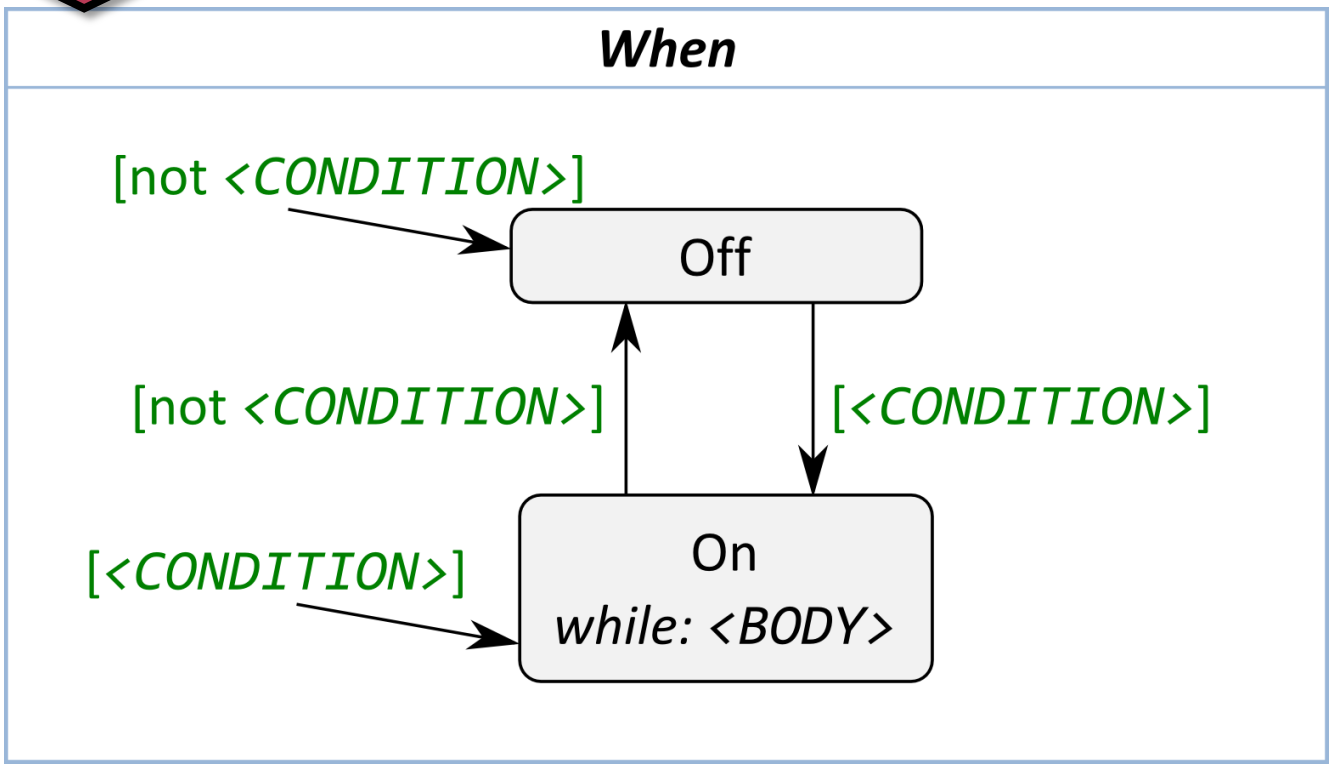
Példa: A követelményminták szemantikája

When *<CONDITION>*, Ensure *<BODY>*

=

When *<CONDITION>* becomes true until it becomes false, let have *<BODY>*

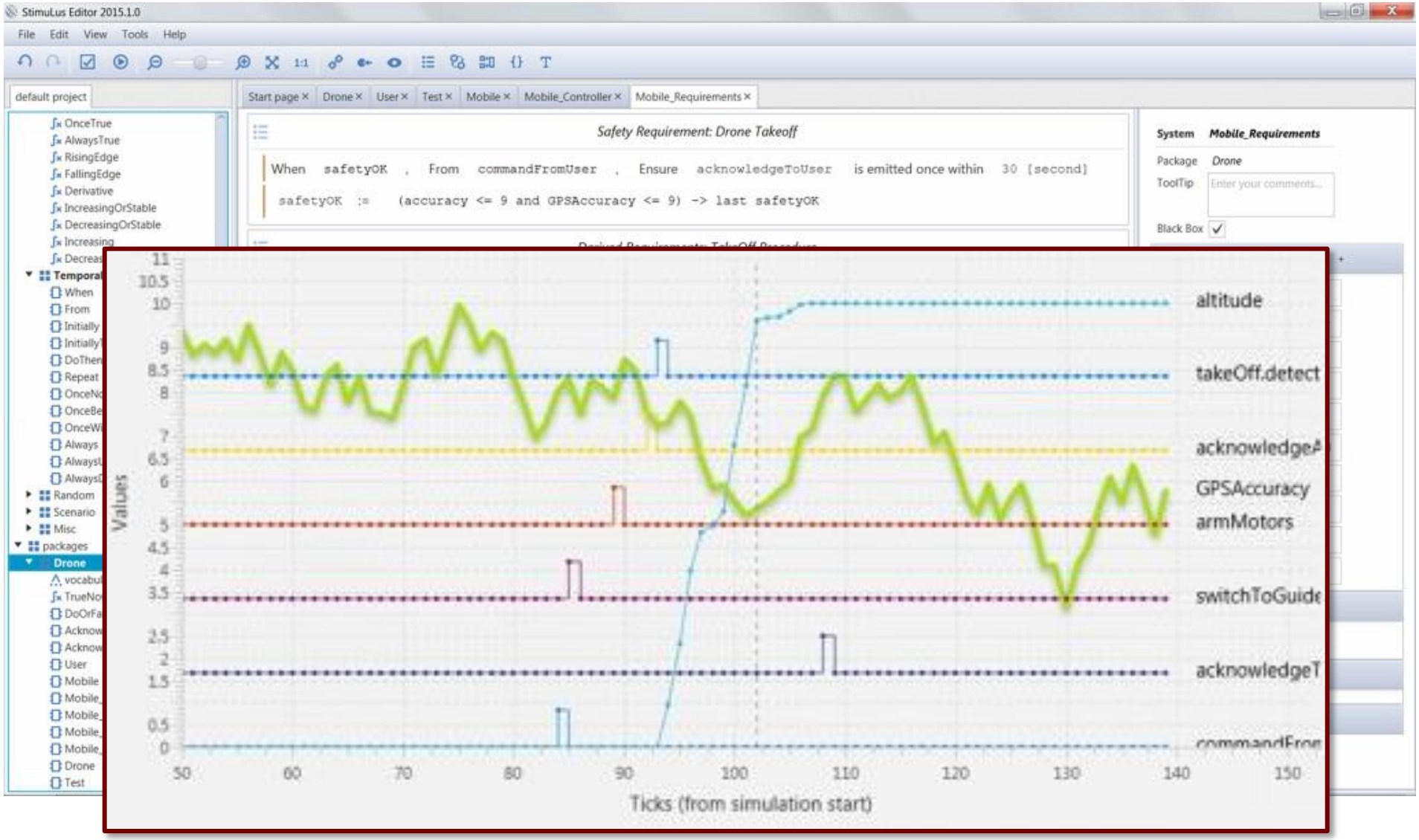
- A blokkok szemantikáját itt egy-egy állapotgép írja le
- Más lehetőség: logikai leírás



A formalizált követelmények felhasználása

- Követelmények validálása
  - A követelményeket kielégítő lefutások generálhatók
    - Tényleg azt rögzítettük, amire gondoltunk?
  - Minta lefutások kiértékelhetők a követelmények szerint
    - Teljes a követelmények halmaza?
  - Ellentmondások felderíthetők
- Formális verifikáció
  - Tervek (modellek) ellenőrzése
- Teszt kiértékelő (test oracle) generálása
  - Implementáció ellenőrzése (teszt környezetben)
- Követelmények dokumentációja
  - Olvasható, de formális háttérrel rendelkező, validált

Példa: Argosim Stimulus eszköz



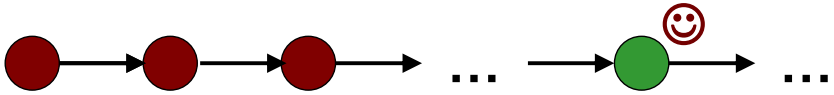
http://argosim.com/

Tanulságok

- Követelmények többsége jellegzetes mintákra illeszkedik
  - Ha … akkor …, Amíg … addig …, Ha …, azután …
  - Előfordulás, sorrendezés eseményekre (állapotokra)
- Az összetett követelmények egyszerűbb mintákból komponálhatók
  - Paraméterezés: Egy-egy esemény, állapot jellemzői
  - Egymásba ágyazás
- A minták formalizálása sokat segít
  - Követelmények vizsgálata: Validáció, teljesség, konzisztencia
  - Tervek (modellek) ellenőrzése: Lefutások kimerítő vizsgálata
  - Teszt kiértékelés, futásidőbeli monitorozás komponensei generálhatók

Formalizálás: Milyen leíró nyelvre van szükségünk?

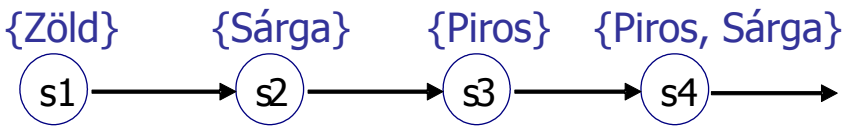
- Elérhetőség: Állapotok bekövetkezésére, állapotok sorrendjére vonatkozik
  - Bekövetkezési sorrend: Megfeleltethető a logikai időnek
    - Jelen időpillanat: Aktuális állapot
    - Következő időpillanat(ok): Rákövetkező állapot(ok)



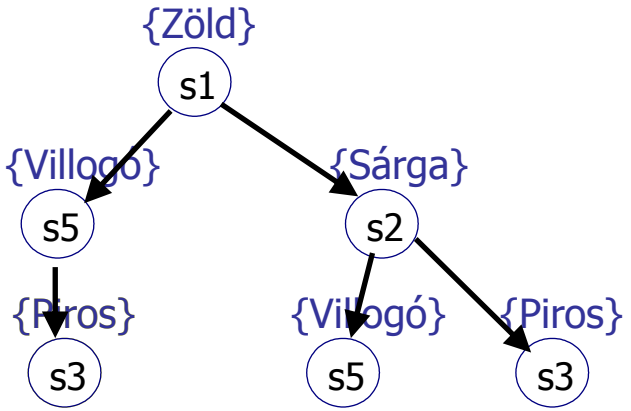
  - Temporális (logikai időbeli, sorrendi) operátorok használhatók a követelmények kifejezésére
- Temporális logikák:
  - Formális rendszer arra, hogy kijelentések igazságának logikai időbeli változását vizsgálhassuk
  - Temporális operátorok: „mindig”, „valamikor”, „mielőtt”, „addig, amíg”, „azelőtt, hogy”, … (megfelelnek a jellegzetes követelmény-mintáknak)

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 értelmezése

- Célkitűzés: Állapottér vizsgálata
- Legegyszerűbb matematikai modell: Kripke-struktúra
  - Állapotok lokális tulajdonságait címkézéssel vezettük be

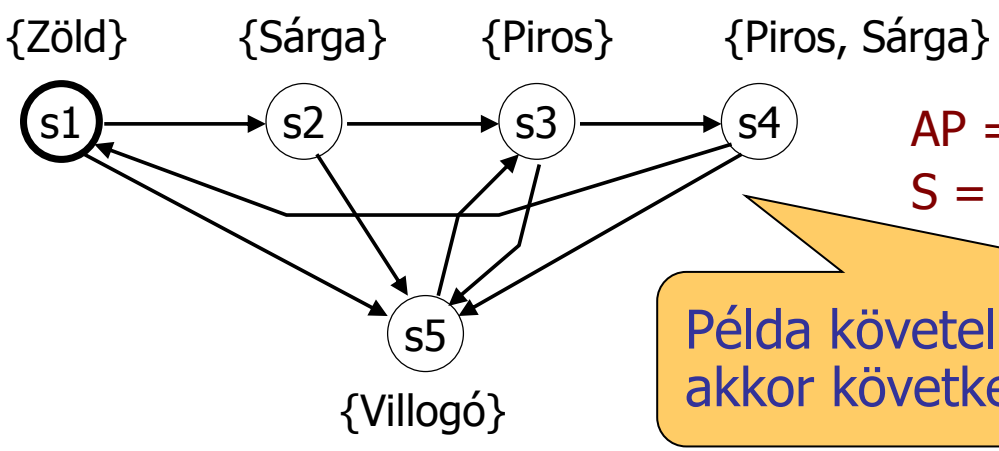
$KS = (S, R, L)$ és AP, ahol

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

$\mathrm{S} = \{\mathrm{s}_1, \mathrm{s}_2, \mathrm{s}_3, ...\mathrm{s}_n\}$ állapotok halmaza

$\mathrm{R} \subseteq \mathrm{S} \times \mathrm{S}$: állapotátmeneti reláció

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



AP = {Zöld, Sárga, Piros, Villogó}
S = {s1, s2, s3, s4, s5}

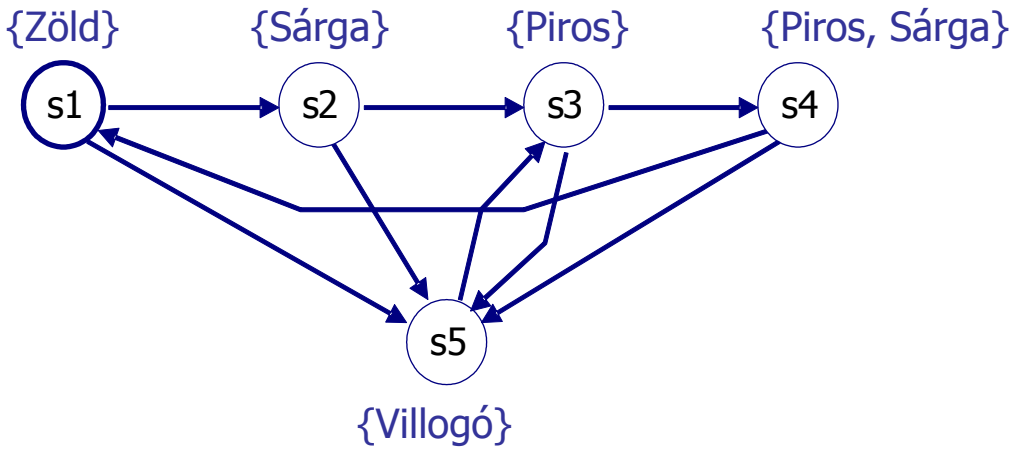
Példa követelmény: A Zöld után a Piros csak akkor következhet be, ha előtte Sárga volt.

Lineáris idejű temporális logika: PLTL

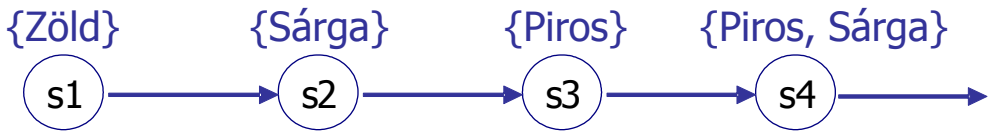
Lineáris idejű temporális logikák

- 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

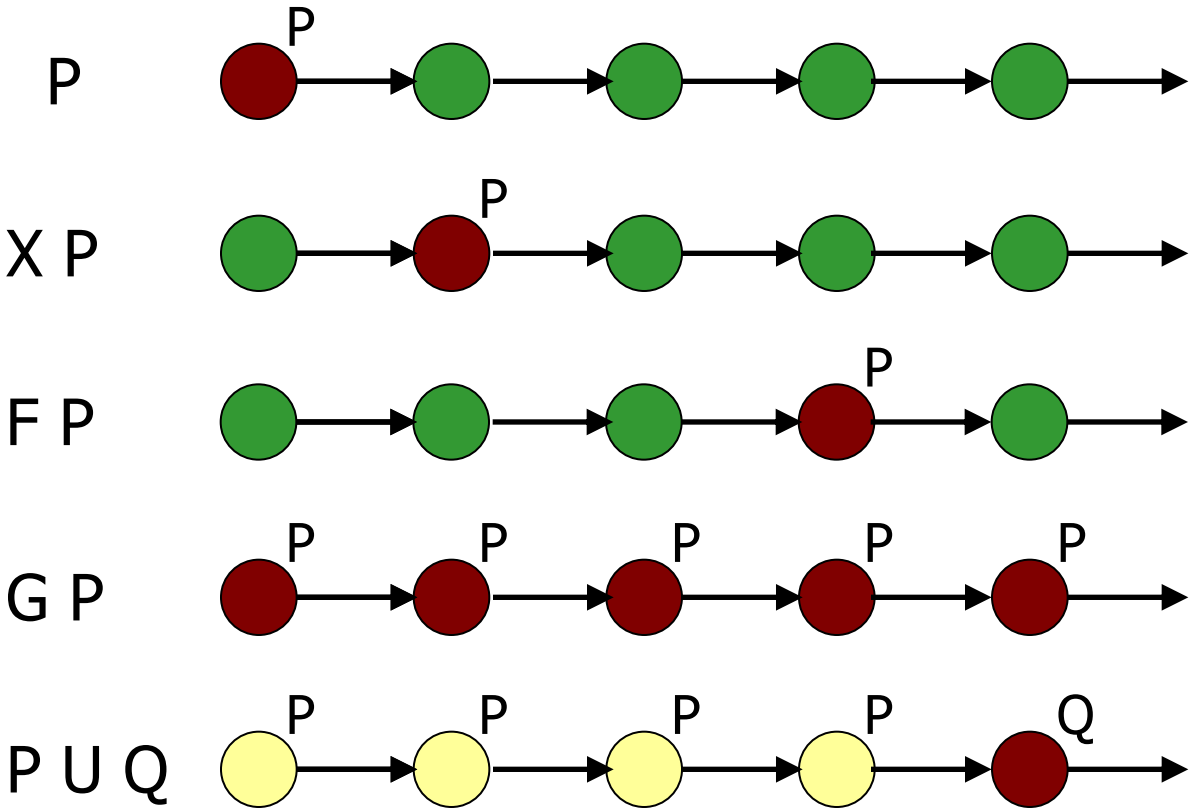
PLTL (Propositional Linear Time Temporal Logic)

p, q, r, ... PLTL kifejezések konstruálása:

- Atomi kijelentések (AP elemei): P, Q, ...
  Lokális tulajdonságok (címkék) állapotokhoz
- Boole logikai operátorok: $\wedge$, $\vee$, $\neg$, $\Rightarrow$
  $\wedge$: És, $\vee$: Vagy, $\neg$: Negálás , $\Rightarrow$: Implikáció
- Temporális operátorok: X, F, G, U informálisan:
  - X p: „neXt p”, a következő állapotban igaz lesz p
  - F p: „Future p”, egy elérhető állapotban igaz lesz p
  - G p: „Globally p”, minden elérhető 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.

- $p \Rightarrow F\ q$

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

  - Példa: Start $\Rightarrow$ F End

- $G(p \Rightarrow F\ q)$

  Minden állapotra fennáll,
  hogy ha p igaz, akkor valamikor q is igaz lesz.

  - Példa: G (Request $\Rightarrow$ F Reply)
    bármikor kiadott kérésre válasz érkezik

- $p\ U\ (q \vee r)$

  A kezdőállapotból p fennáll, amíg q vagy r igaz nem lesz.

  - Példa: Requested U (Accept $\vee$ Refuse)
    folyamatos kérést válasz vagy elutasítás követ

- $(p \wedge G(p \Rightarrow X\ p)) \Rightarrow G\ p$

  A matematikai indukció leírása (mindig teljesül)

PLTL példák II.

- GF p

  Minden állapotra igaz, hogy abból tekintve a további futást valamikor p igaz lesz.

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

- FG p

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

  - Példa: FG Normal
    a kezdeti tranziens után a Normal tulajdonságú üzemi állapotokba kerül a rendszer

Követelmények formalizálása: Példa

Adott egy klímaberendezés, aminek a következő üzemmódokat kell biztosítania:

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

- Egy-egy állapothoz több címke tartozhat!
  - Pl. {Bekapcsolva, Szellőztet}
- A követelmény formalizálás fázisában a teljes viselkedést (modellt) még nem feltétlenül ismerjük
  - Csak címkékkel ellátott állapotokat tételezünk fel

Példa (folytatás)

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

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

Példa (folytatás)

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

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

  $G\ (\ X\ \text{Elromlott} \Rightarrow \text{Bekapcsolva})$
- A fűtési fázis befejezésekor szellőztetni kell:

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

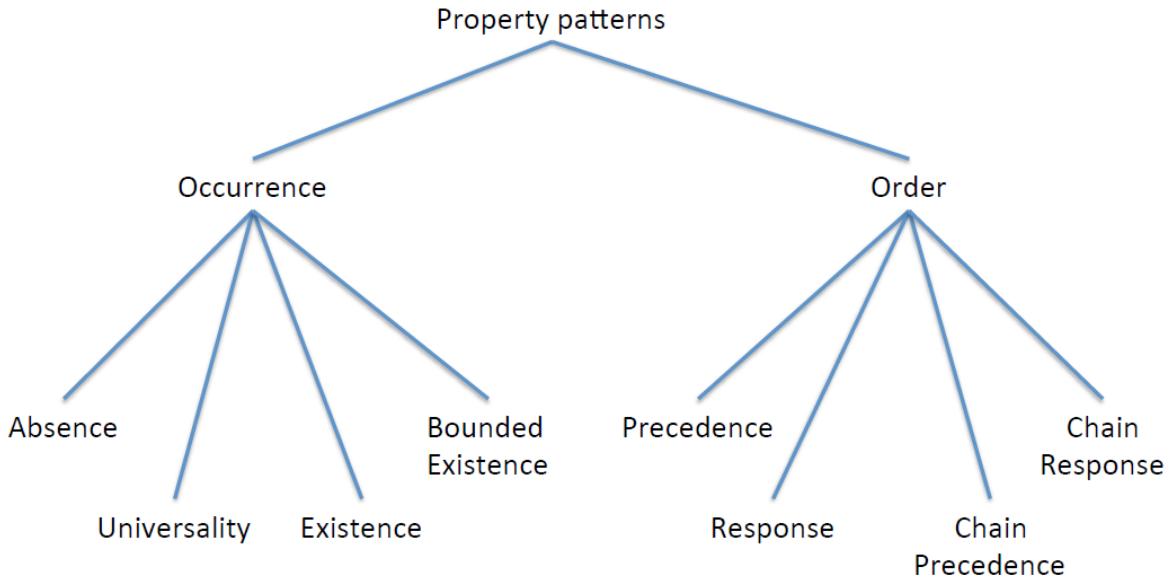
  de el is romolhat:

  $G\ ((\text{Fűt} \wedge X(\neg\text{Fűt})) \Rightarrow X\ (\text{Szellőztet} \vee \text{Elromlott}))$
- Szellőztetés után mindaddig nem hűthet erősen, míg egy gyenge hűtéssel nem próbálkozott:

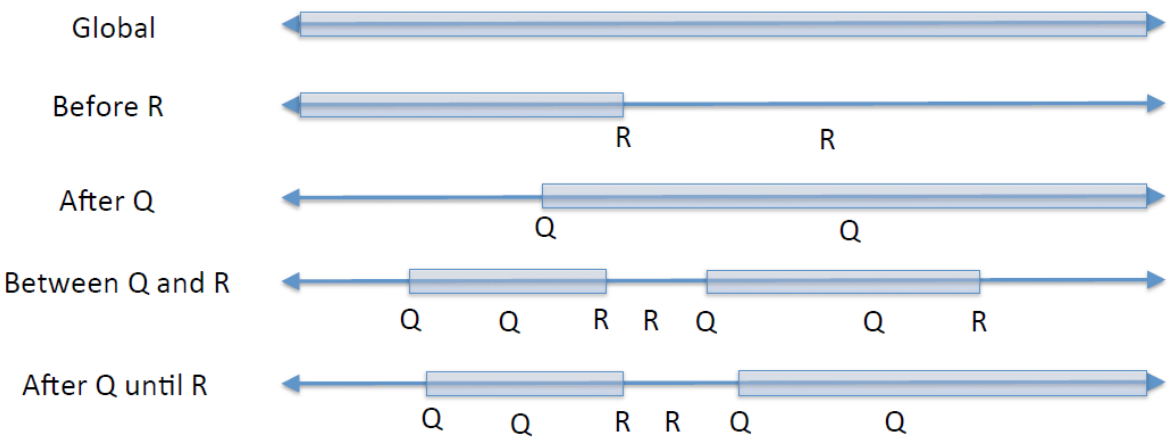
  $G\ ((\text{Szellőztet} \wedge X(\neg\text{Szellőztet})) \Rightarrow X(\neg\text{ErősenHűt}\ U\ \text{GyengénHűt}))$

Mivel kezdtük: Jellegzetes követelményminták

Minták:
Sorrendi előírások



Hatókörök:
További eseményekhez képest



Követelményminták formalizálása (példák)

Universality within scope	**Property in LTL**
P occurs in each step of the execution globally.	$G\ P$
P occurs in each step of the execution before Q.	$F\ Q \rightarrow (P\ U\ Q)$
P occurs in each step of the execution after Q.	$G(Q \rightarrow G\ P)$
P occurs in each step of the execution between Q and R.	$G((Q \wedge \neg R \wedge F\ R) \rightarrow (P\ U\ R))$

Existence within scope	**Property in LTL**
P occurs in the execution globally.	$F\ P$
P occurs in the execution before Q.	$\neg\ Q\ WU\ (P \wedge \neg\ Q)$
P occurs in the execution after Q.	$G\ (\neg Q) \vee$ $F\ (Q \wedge F\ P)$
P occurs in the execution between Q and R.	$G((Q \wedge \neg R \wedge F\ R) \rightarrow (\neg R\ WU\ (P \wedge \neg R)))$

Szöveges követelmények formalizálása (példák)

Ha α és β igaz, akkor α-nak igaznak kell maradnia mindaddig, amíg β is igaz.

$$\mathbf{G}\big((\alpha \wedge \beta) \rightarrow (\alpha\ \mathbf{U}\ \neg\beta)\big)$$

Ha az `alarm` be van kapcsolva és `alert` történik, a `safety` kimenet váljon igazzá, amíg az `alarm` be van kapcsolva.

$$\mathbf{G}((\mathrm{alarm} = \mathrm{ON} \wedge \mathrm{alert}) \rightarrow \mathrm{X}(\mathrm{safety}\ \mathbf{U}\ \neg\mathrm{alarm}))$$

A PLTL formális szintaxisa és szemantikája

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 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

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 true U p
  G p jelentése $\neg F(\neg p)$
- „Mielőtt” operátor (before):
  p WB q = $\neg((\neg p)$ U $q)$ (weak before)
  p B q = $\neg((\neg p)$ U $q) \wedge$ F q (strong before)

Informálisan:
Nem igaz, hogy nem fordul elő p a q előtt

PLTL szemantika: Jelölések

- $M = (S, R, L)$ Kripke-struktúra
- $\pi = (s_0, s_1, s_2, \ldots)$ 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:
  az M modellben a π útvonalon igaz p

A PLTL szemantikája megadja, hogy mikor igaz egy adott útvonalon egy adott PLTL kifejezés.

PLTL szemantika

A szintaxis szabályok alapján képzett kifejezésekhez induktívan (a kifejezés felépítése szerint) megadható a formális szemantika:

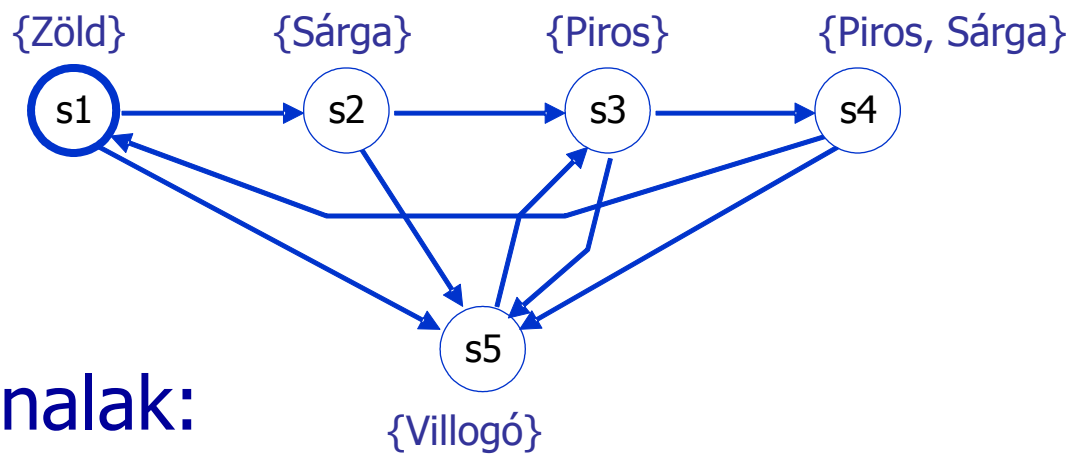
- **L1**: $M,\pi \models P$ a.cs.a. $P \in L(s_0)$
- **L2**: $M,\pi \models p \wedge q$ a.cs.a. $M,\pi \models p$ és $M,\pi \models q$
  $M,\pi \models \neg q$ a.cs.a. $M,\pi \models q$ nem igaz.
- **L3**: $M,\pi \models (p\ U\ q)$ a.cs.a.

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

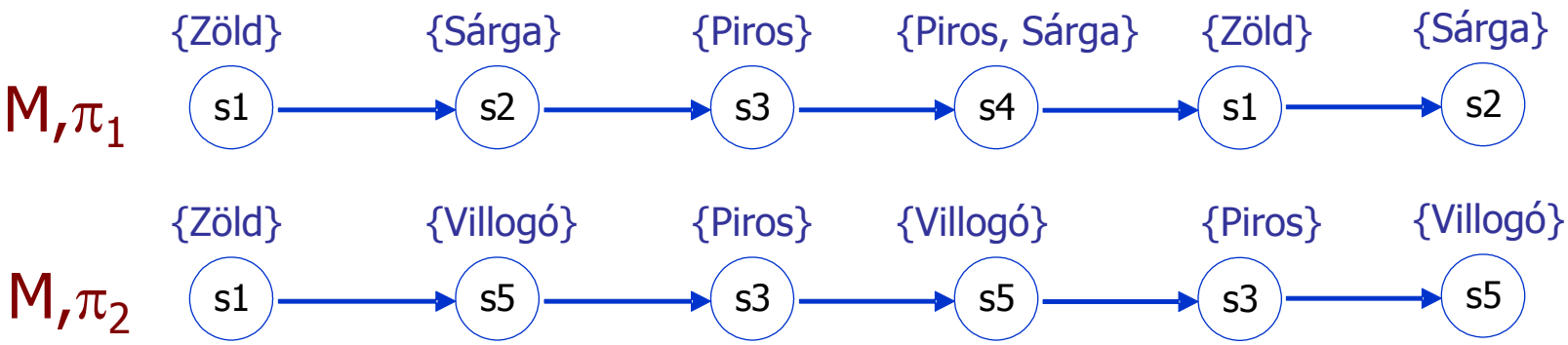
$M,\pi \models X\ p$ a.cs.a. $\pi^1 \models p$

PLTL kifejezések értelmezése: Példák

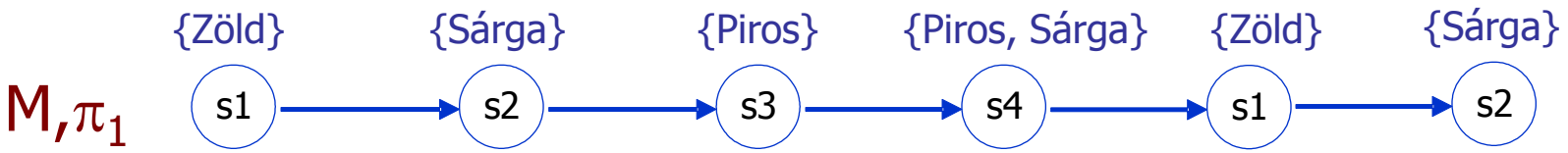
- M Kripke-struktúra:



- Példa útvonalak:

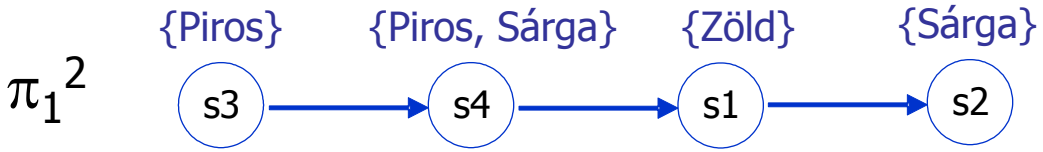


Példák (folytatás)



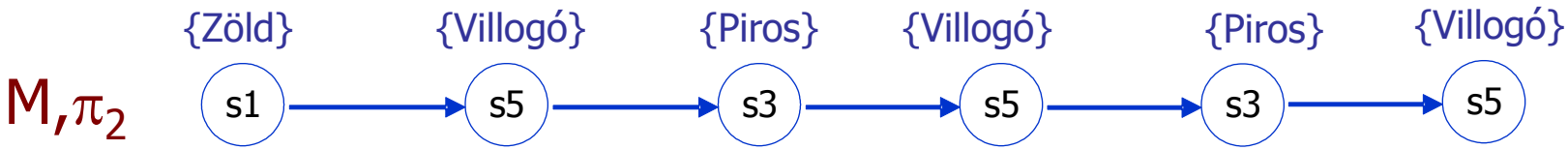
- M,π_1 |= Zöld, mert Zöld a kezdőállapot címkéje

- M,π_1 |= F (Piros U Zöld) igaz,
  mert van olyan szuffix,
  amire teljesül a (Piros U Zöld):

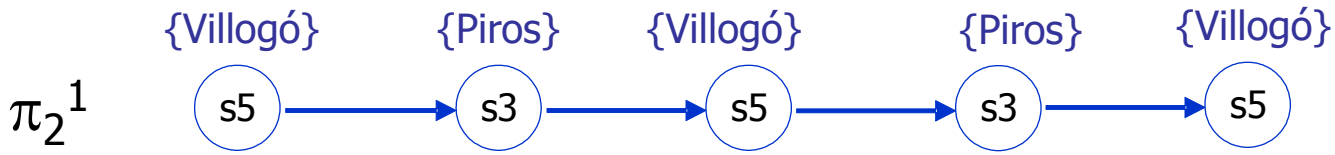


- M,π_1 |= Piros U Zöld igaz (itt F operátor nélkül is)

Példák (folytatás)



- $M,\pi_2 \models X\ F\ (XX\ Piros)$, mert az első szuffixre:



F (XX Piros) teljesül, mert
π_2^1-nek van olyan szuffixe, ahol XX Piros igaz:

PLTL modell kiterjesztése LTS-re

- LTS, Labeled Transition System
- Állapotátmenetek címkézhetők egy-egy ún. akcióval, egy átmeneten csak egy akció szerepelhet
- Állapotátmenetek tulajdonságait fejezzük ki

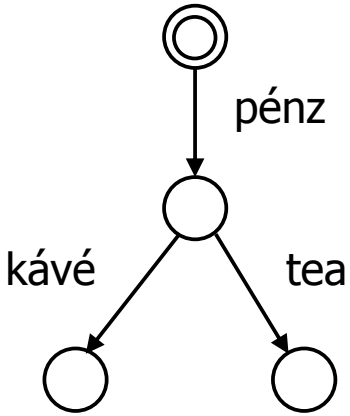
$LTS = (S, Act, \rightarrow)$, ahol

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

$Act = \{a, b, c, \ldots\}$ akciók (címkék) halmaza

$\rightarrow \subseteq S \times Act \times S$ címkézett állapotátmenetek

Állapotátmenetek szokásos jelölése: $s_1 \xrightarrow{a} s_2$



PLTL értelmezése LTS-en

A struktúra bővülése miatt az útvonal:

- $\pi = (s_0, a_1, s_1, a_2, s_2, a_3, \ldots)$

A szintaxis módosítása:

- **L1***: Ha a egy akció, akkor (a) egy PLTL kifejezés.

A kapcsolódó szemantikai szabály:

- **L1***: $M,\pi \models (a)$ a.cs.a. $a_1=a$
  ahol a_1 az első akció π-ben.

Ilyen módon üzenetküldéssel kommunikáló rendszerek tulajdonságai is megfogalmazhatók.

PLTL összefoglalás

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