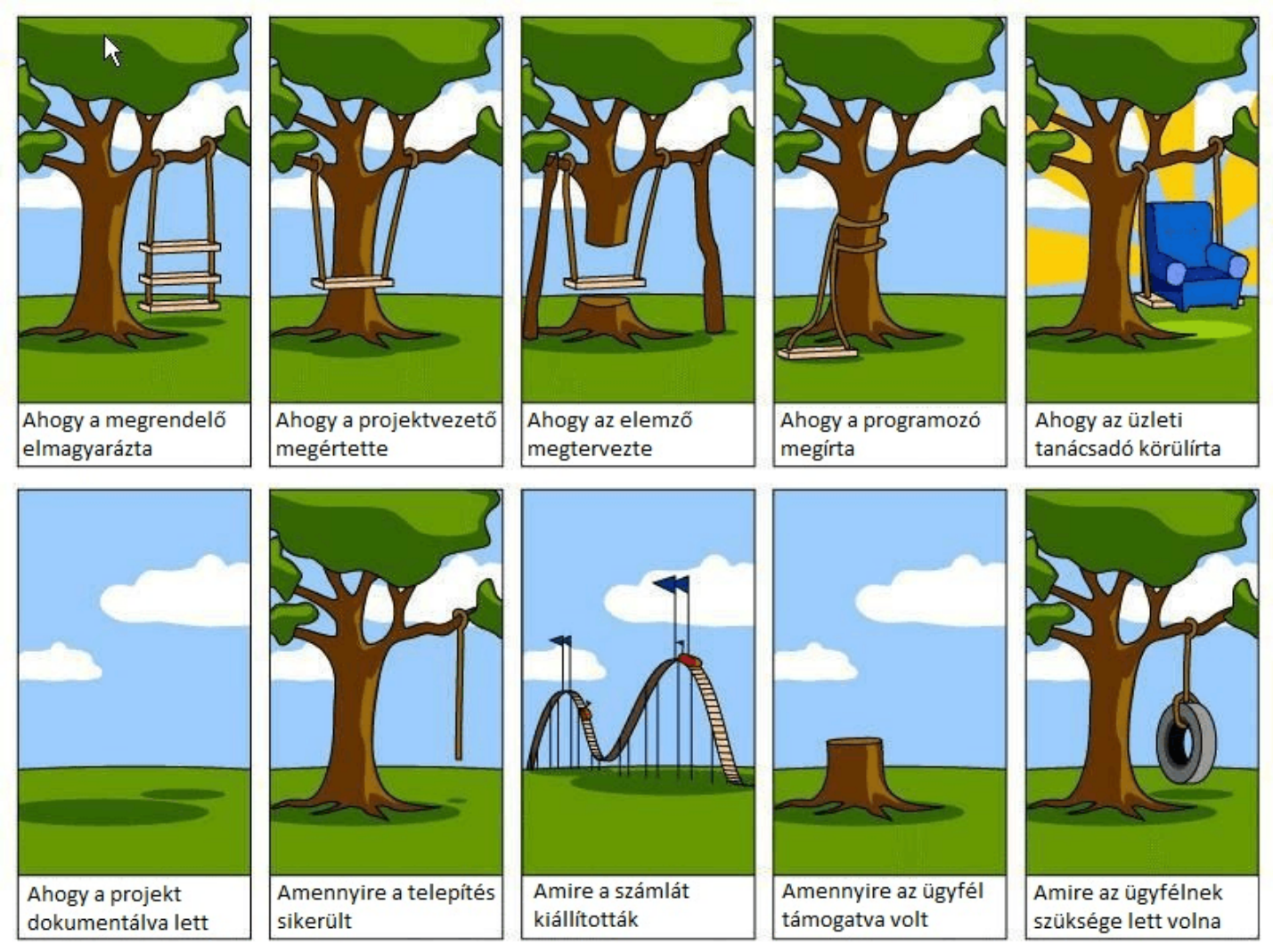
Követelmények formalizálása: Temporális logikák

dr. Bartha Tamás
BME Közlekedés- és Járműirányítási Tanszék

dr. Majzik István
BME Méréstechnika és Információs Rendszerek Tanszék

Miért jó a követelményeket formalizálni?



Mintapélda: Kölcsönös kizárás

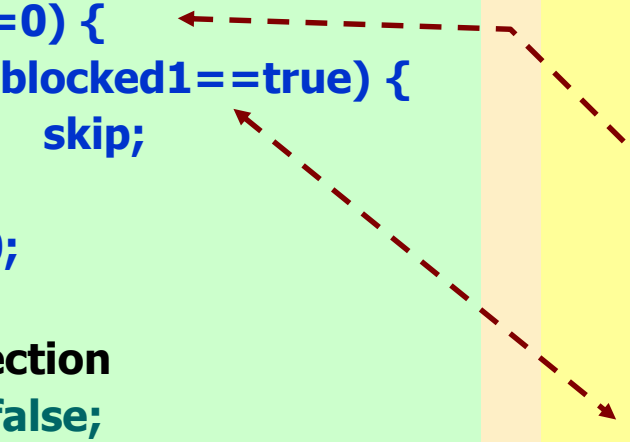
- 2 résztvevőre, 3 megosztott változóval (H. Hyman, 1966)
  - **blocked0**: Első résztvevő (P0) be akar lépni
  - **blocked1**: Második résztvevő (P1) be akar lépni
  - **turn**: Ki következik belépni (0 esetén P0, 1 esetén P1)

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
}
```



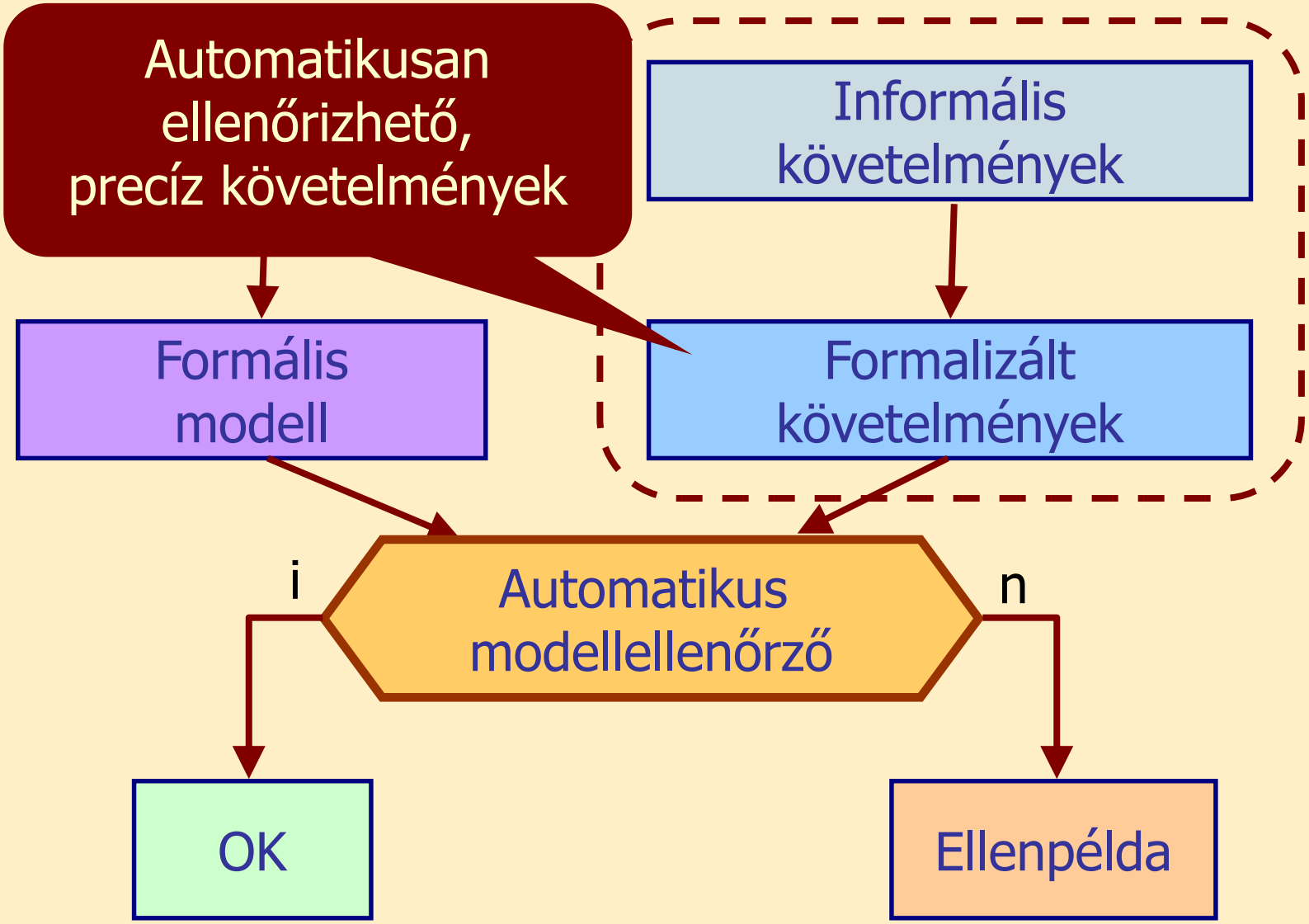
Helyes-e ez az algoritmus?

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 formalizálhatjuk ezeket?

Mit szeretnénk elérni?



Követelmények rögzítése és formalizálása

Mit tudunk a jellegzetes követelményekről (kritikus rendszerekben)?

Mit érdemes formalizálni?

Szöveges követelmények kezelése

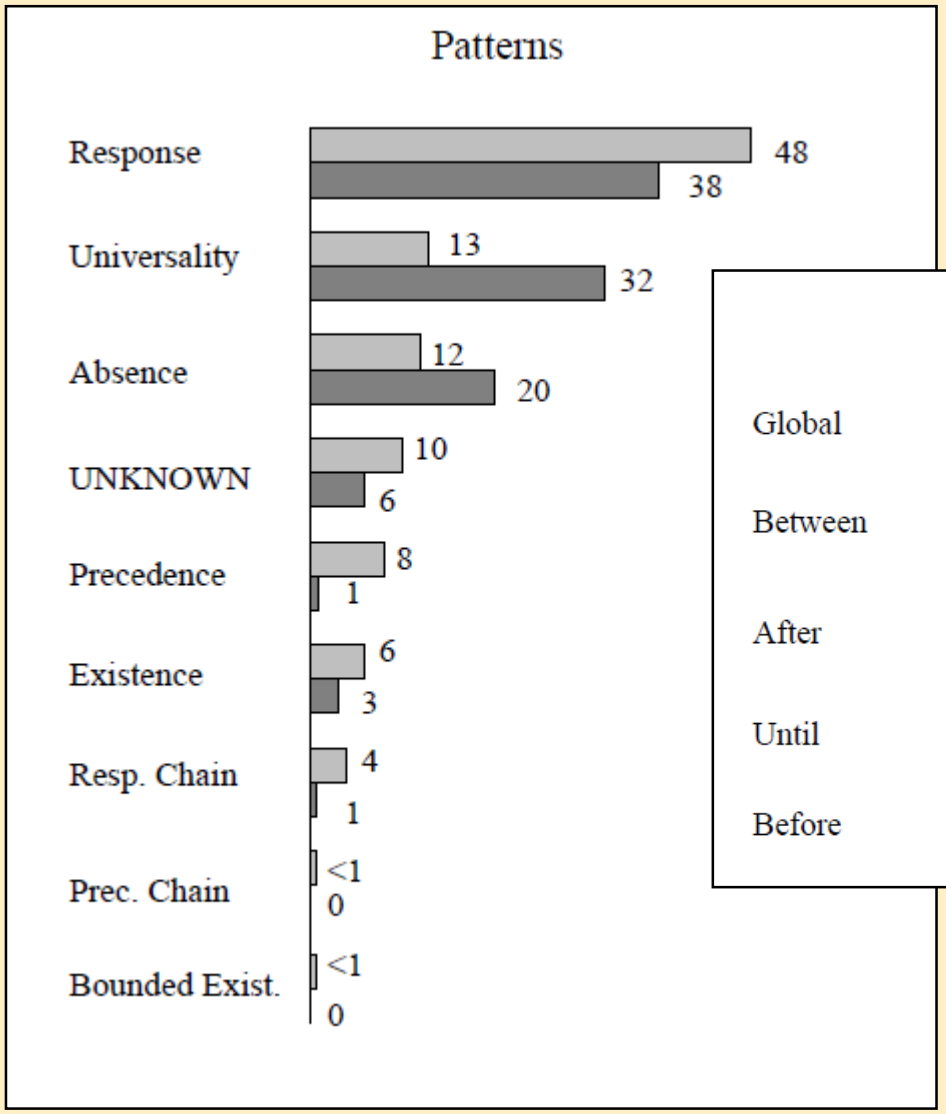
- Tipikus követelmény megadás: Szöveges forma

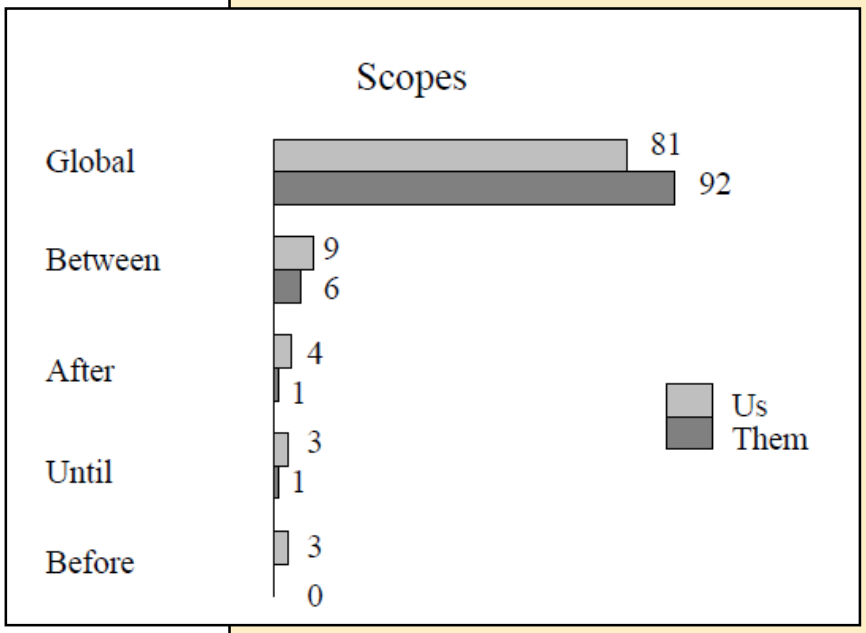
Ha az `alarm` be van kapcsolva és `alert` történik, a `safety` kimenet legyen igaz, amíg az `alarm` be van kapcsolva.

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, ...)

Egy felmérés eredménye





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

A minták jelentése (magyarázat)

Occurrence: Előfordulás

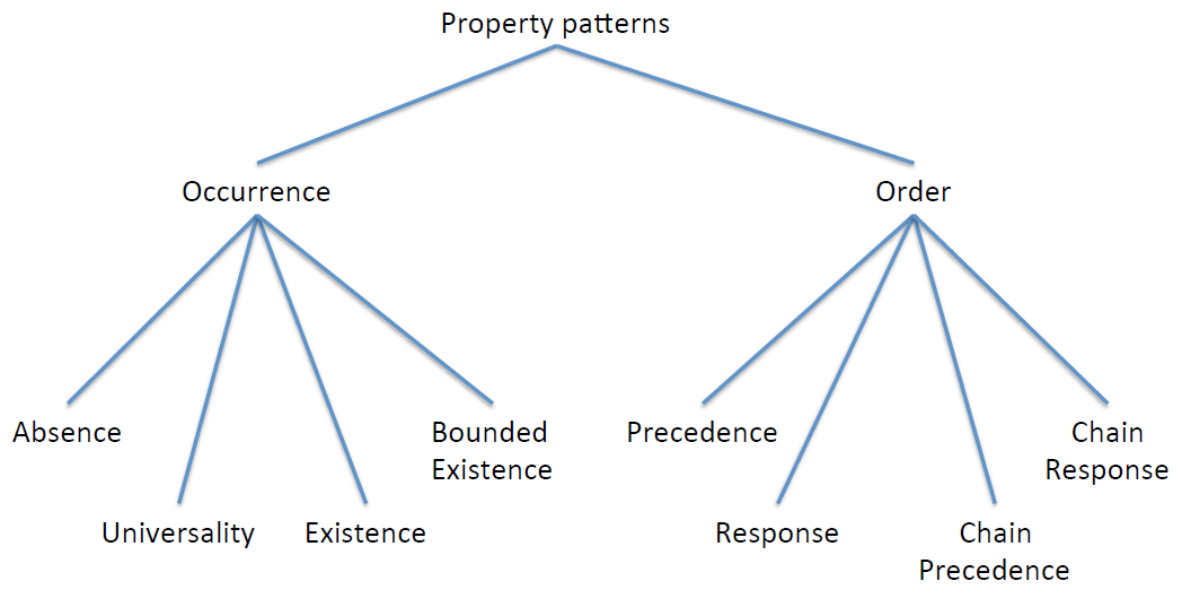
- Absence: A hivatkozott állapot/esemény nem fordul elő.
- Universality: A hivatkozott állapot/esemény folyamatosan előfordul.
- Existence: A hivatkozott állapot/esemény egyszer biztosan előfordul.
- Bounded existence: A hivatkozott állapot/esemény biztosan előfordul „k” alkalommal (létezik „legalább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

Order: Sorrendezés

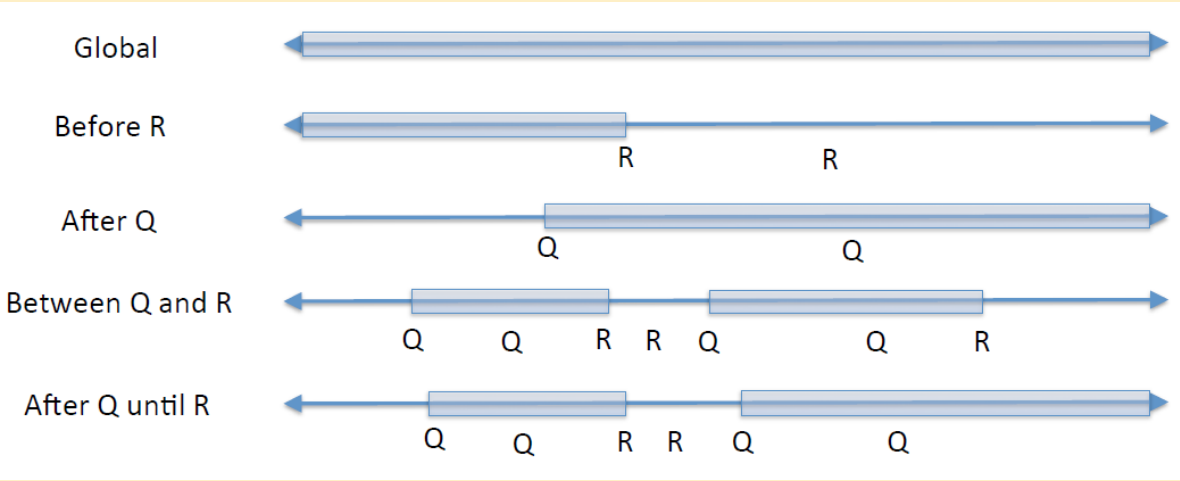
- Precedence: Az egyik állapot/esemény mindenképpen meg kell, hogy előzze a másik állapotot/eseményt.
- Response: Az egyik állapotot/eseményt mindenképpen kell, hogy kövesse egy másik meghatározott állapot/esemény.
- Chain precedence: A Precedence minta általánosítása állapot / esemény sorozatokra.
-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

Követelmény minták csoportosítása

Minták: Sorrendi előírások



Hatókörök: További eseményekhez képest



Példák a minták megjelenésére

- Response minta Global hatókörben:

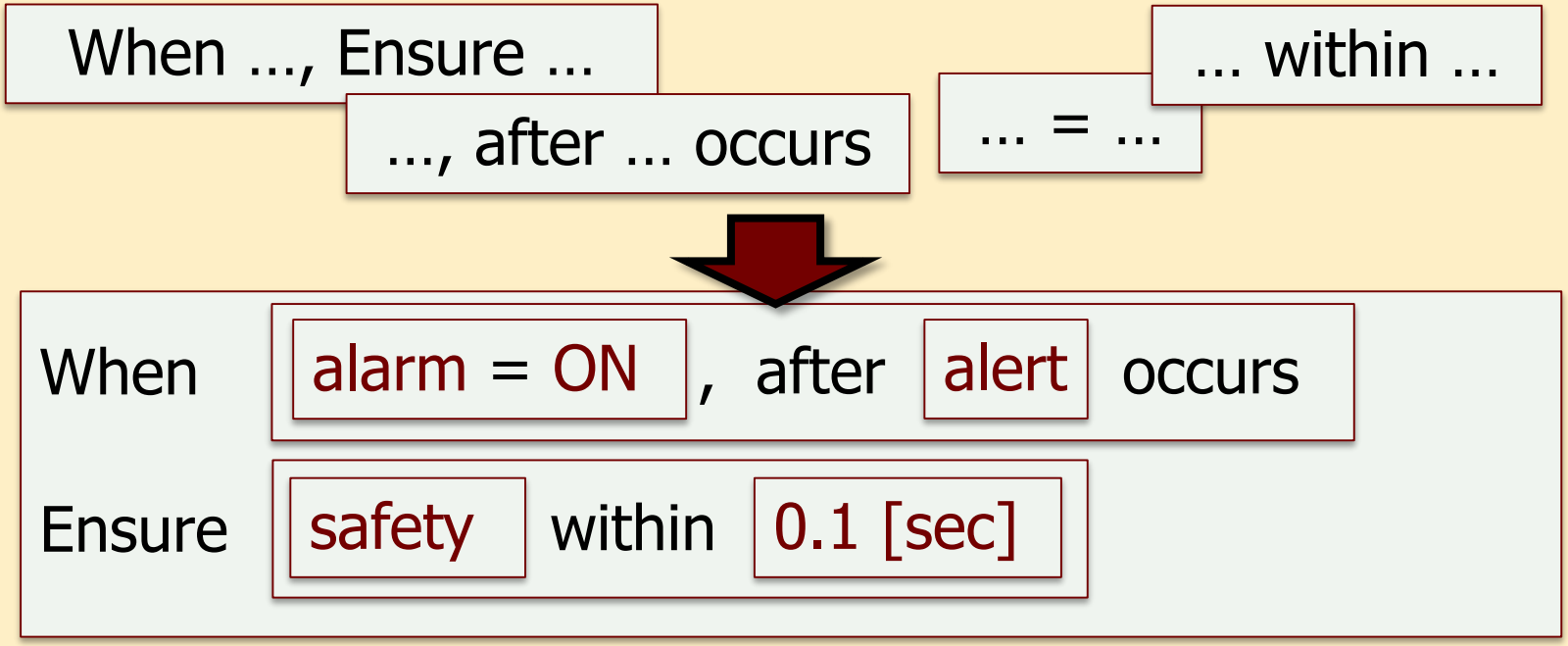
A végrehajtás során bármikor,
ha Request kérés következik be, akkor vagy egy Reply
vagy egy Reject válasznak kell következnie.

- 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



* Darvas Dániel által készített példák a STIMULUS eszköz alapján

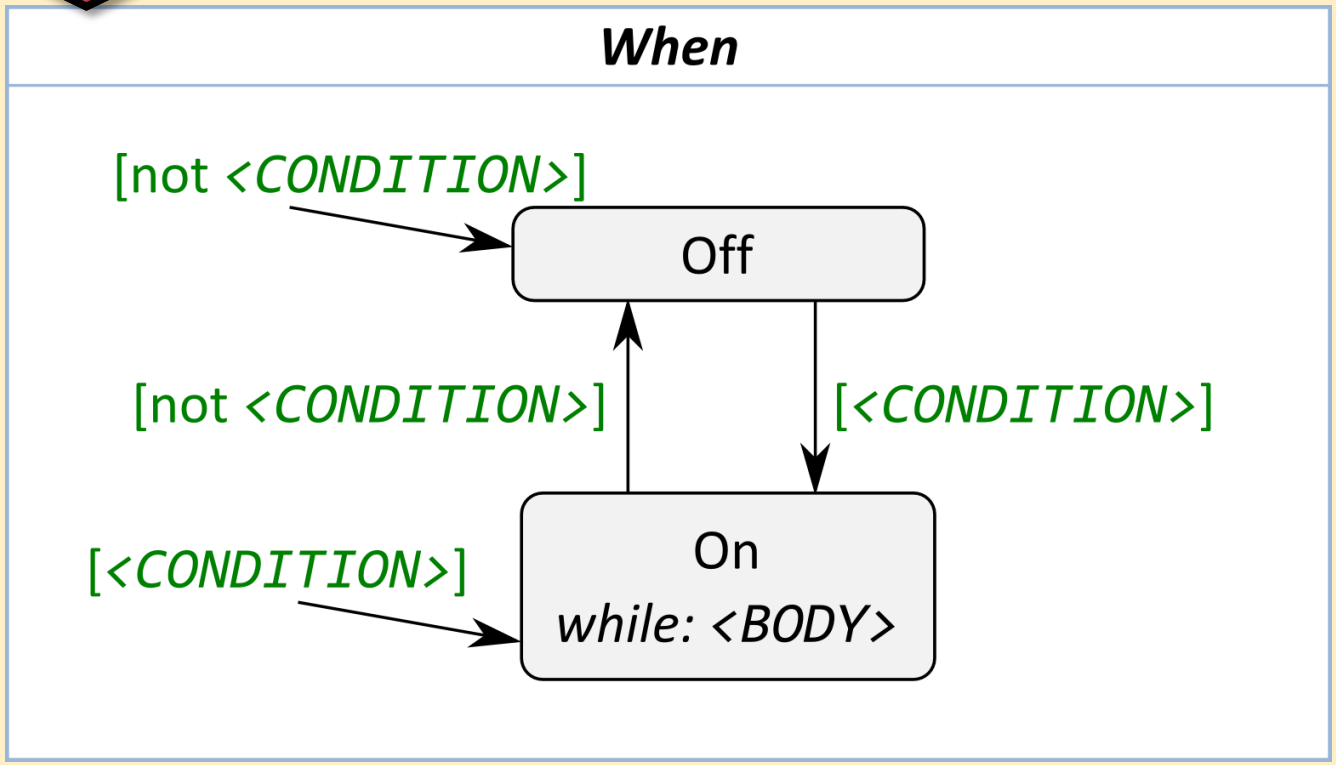
Példa: A követelményminták szemantikája

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

=

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

A blokkok szemantikáját itt egy-egy állapotgép írja le

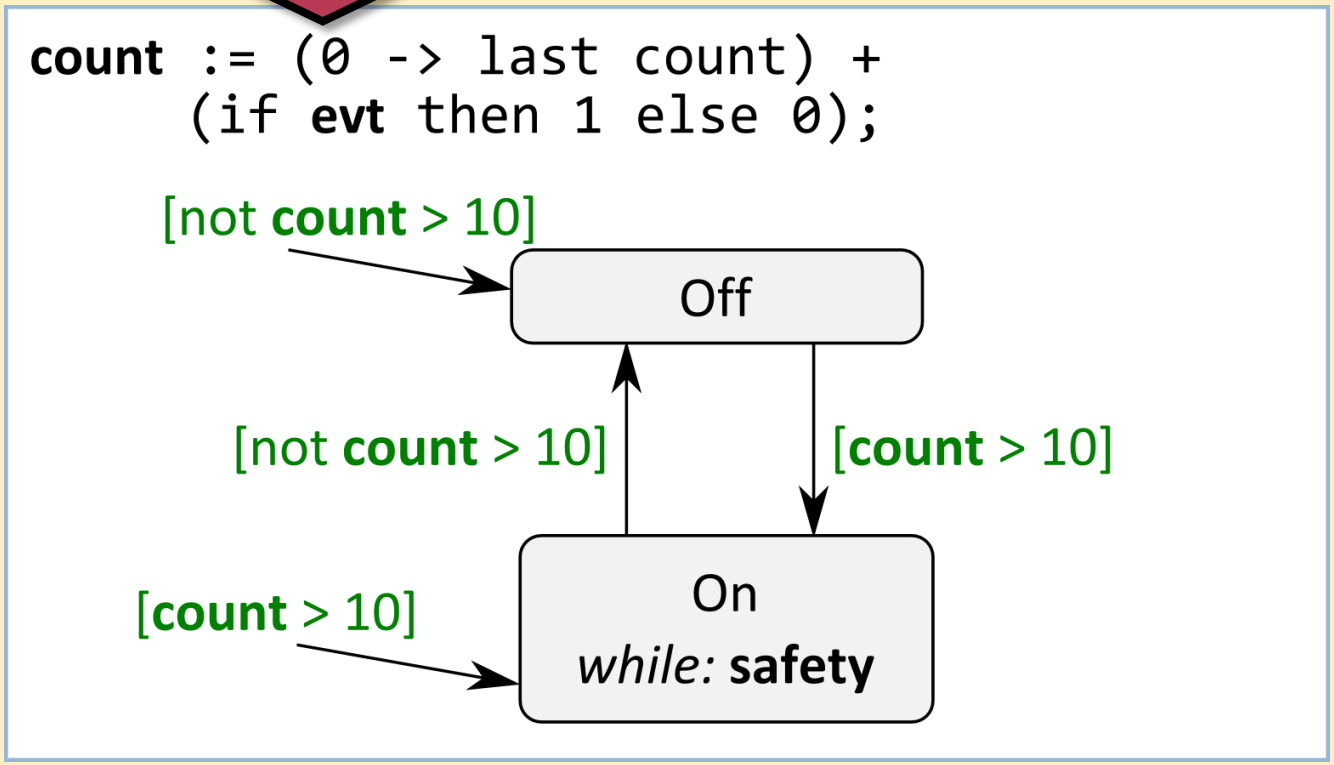


Példa: Kompozit minták szemantikája

When number of occurrences of evt > 10

Ensure safety

```
count := (0 -> last count) +
         (if evt then 1 else 0);
```



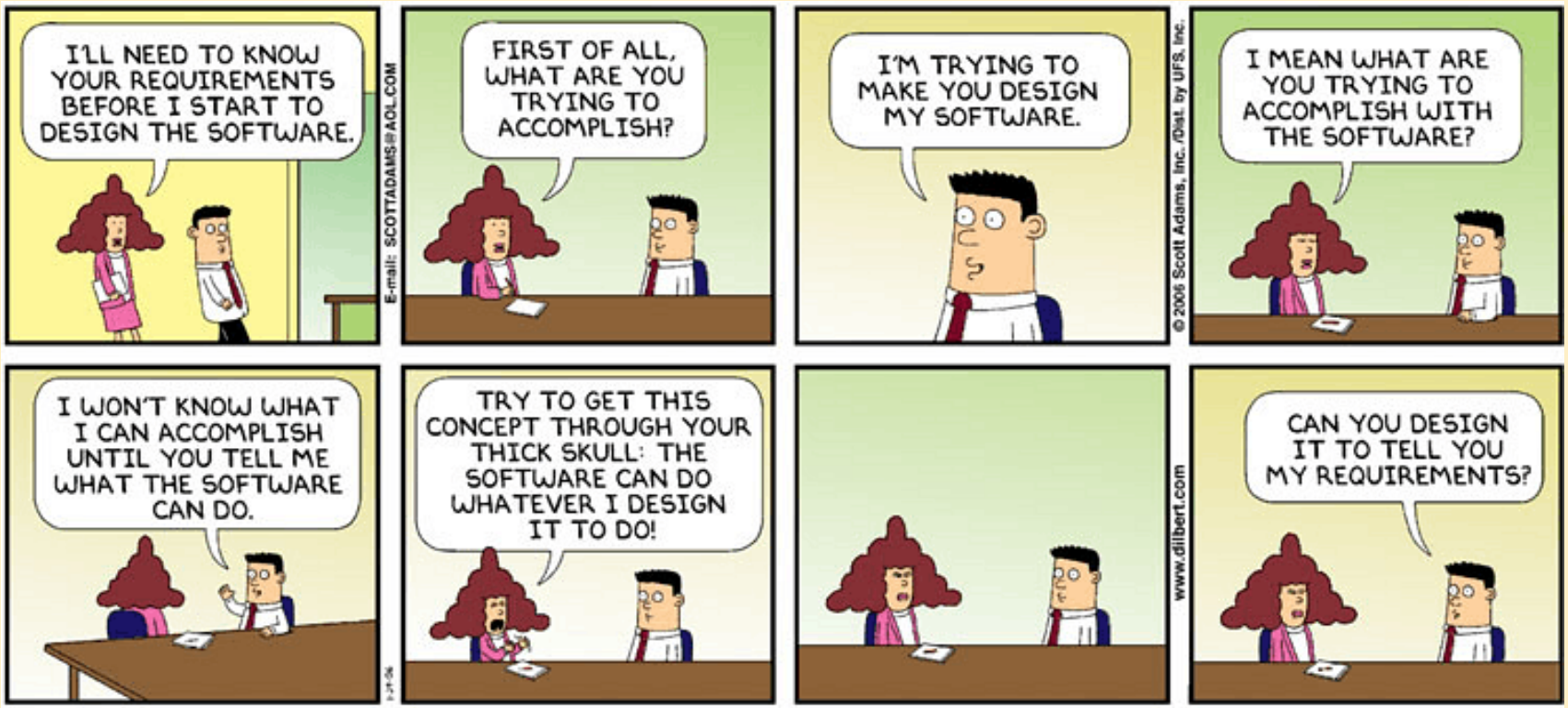
A formalizált követelmények felhasználása

- 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

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

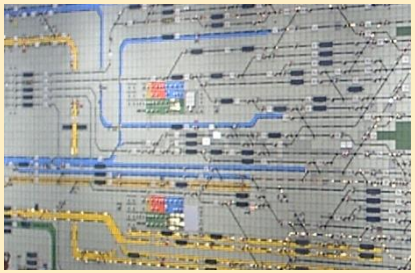
Követelmények formalizálása
Temporális követelmények



Milyen jellegű követelményeket formalizálunk?

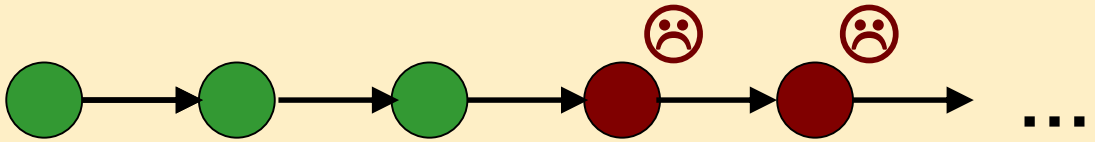
- Verifikáció: Modell ↔ Sokféle követelmény
  - Funkcionális: Logikailag helyes a működés ← Most ez a célunk
  - Extra-funkcionális: Teljesítmény, megbízhatóság, … ← Később
- Célkitűzés: Állapotok elérhetőségének ellenőrzése
  - Állapotok bekövetkezési sorrendjét vizsgáljuk
    - Eljuthatunk-e kedvező állapotba? → Élő jellegű követelmények
    - Elkerüljük-e a kedvezőtlen állapotokat? → Biztonsági követelmények

    Ezek az állapottér teljes felderítésével ellenőrizhetők!
  - Állapotok lokális tulajdonságára hivatkozunk
    - Állapot neve, állapotváltozók értéke
- Fontos állapot alapú, eseményvezérelt rendszerekben

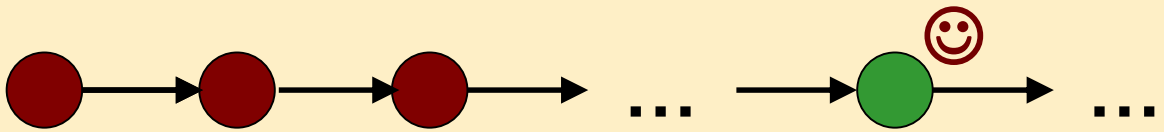

„Biztonsági” jellegű követelmények

- Veszélyes helyzetek elkerülését írják elő:
  - „Minden állapotban kisebb a nyomás a kritikusnál.”
  - „A présgép csak lecsukott ajtó mellett üzemelhet.”
- Univerzális tulajdonság az elérhető állapotokon:
  - „Minden elérhető állapotban igaz, hogy …”
  - Invariánst fogalmaznak meg
- Ha egy állapotsorozat nem teljesíti:
  - nem is egészíthető ki úgy, hogy teljesítse
- Közlekedési példák:
  - Holtpontmentesség:
    nem lehet minden irányból végtelen sokáig piros jelzés
  - Kölcsönös kizárás:
    keresztező irányok nem kaphatnak egyszerre zöldet



„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 tulajdonság az elérhető állapotokon
  - „Létezik (elérhető) olyan állapot, hogy …”
- Ha egy állapotsorozat nem teljesíti:
  - elvileg kiegészíthető úgy, hogy teljesítse
- Közlekedési példák:
  - Előbb-utóbb a közlekedési lámpa zöldre vált
  - Szükség esetén a biztosító berendezés működésbe lép
  - A fékezés során a kerék megcsúszását az ECU érzékeli és aktiválja az ABS-t

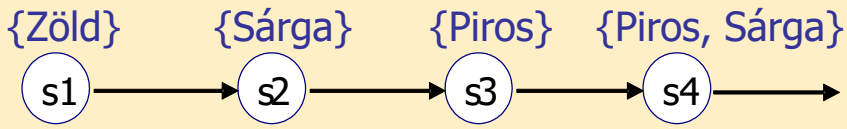


Az elérhetőségi követelmények leíró nyelve

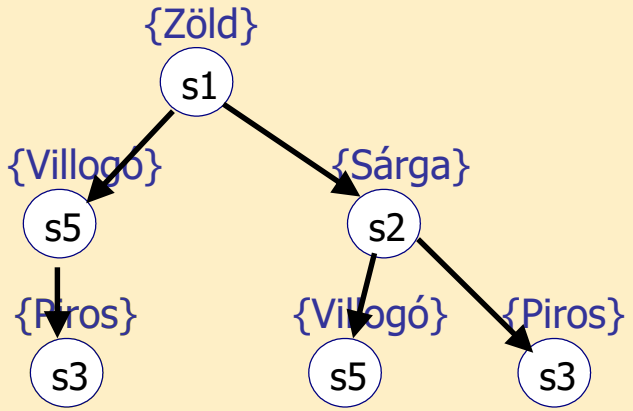
- Állapotok bekövetkezési sorrendjére vonatkozik
  - Bekövetkezési sorrend: Megfeleltethető a logikai időnek
    - Jelen időpillanat: Aktuális állapot
    - Következő időpillanat(ok): Rákövetkező állapot(ok)
  - Temporális (logikai időbeli, sorrendi) operátorok használhatók a követelmények kifejezésére
- 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

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

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

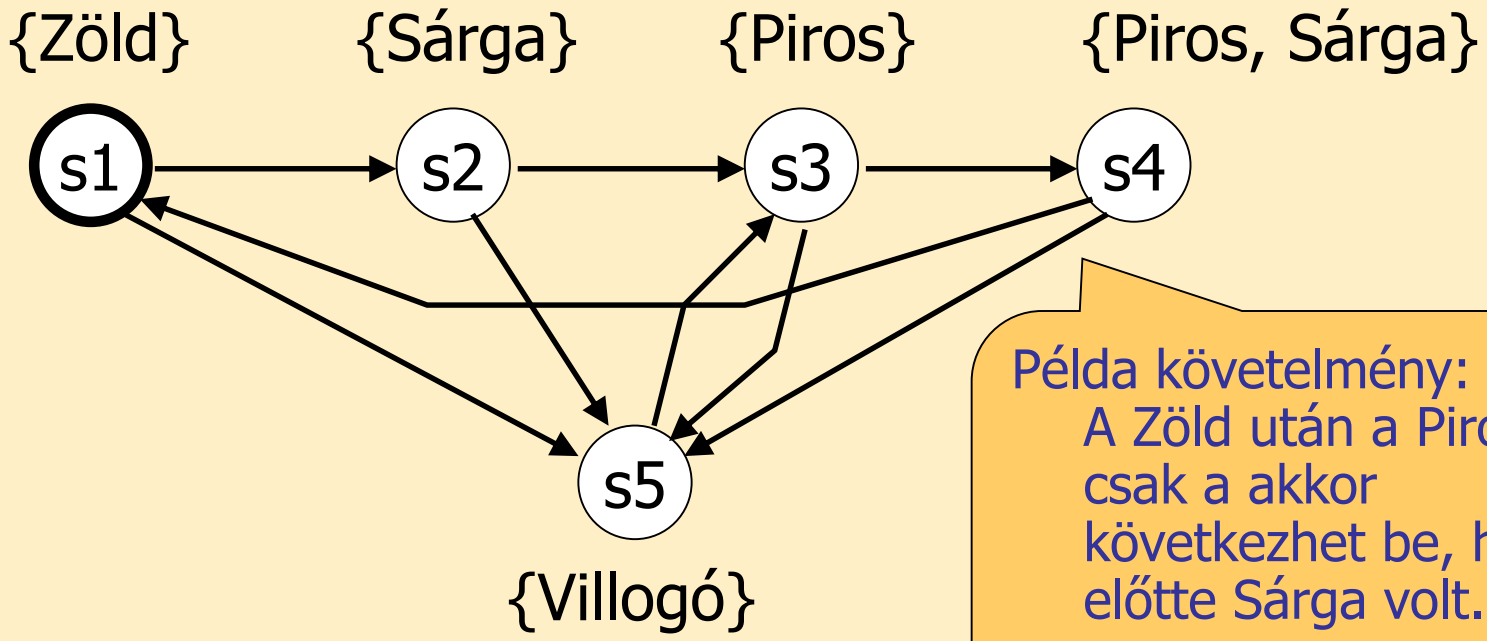
$\mathrm{R} \subseteq \mathrm{S} \times \mathrm{S}$: állapotátmeneti reláció

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

Kripke-struktúra példa

Közlekedési lámpa vezérlője

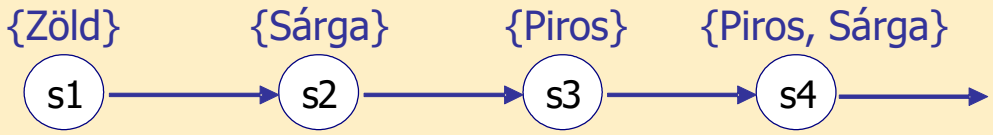
- AP = {Zöld, Sárga, Piros, Villogó}
- S = {s1, s2, s3, s4, s5}



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

Teljesül-e?

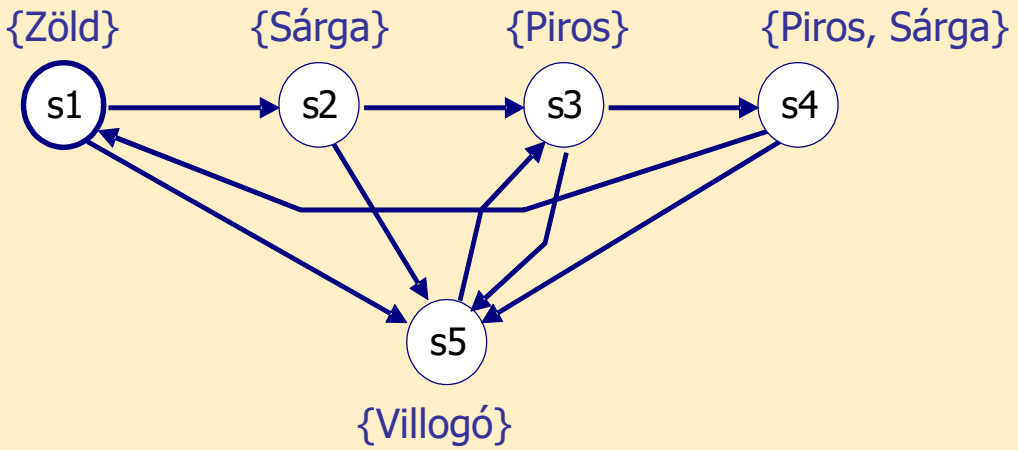
Lineáris idejű temporális logika: PLTL



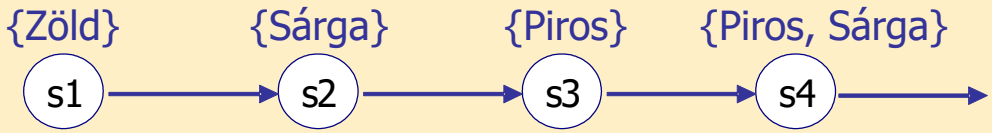
Lineáris idejű temporális logikák

- 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)

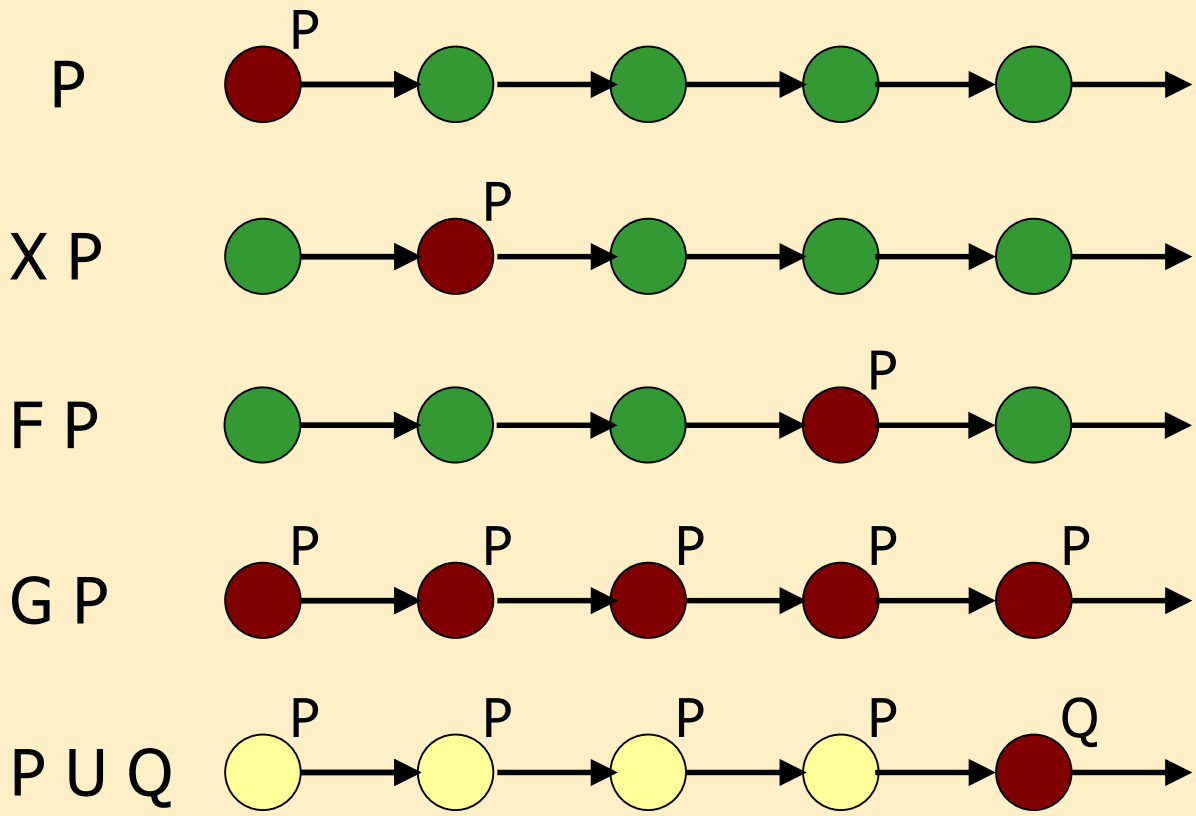
p, q, r, ... kifejezések konstruálása:

- Atomi kijelentések (AP elemei): P, Q, ...
- 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.
  - Példa: Start $\Rightarrow$ F End
- $G(p \Rightarrow Fq)$

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

  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 (valós) viselkedést még nem ismerjük

  → épp az a cél, hogy a követelményeknek megfelelően tervezzük meg!

Példa (folytatás)

AP={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 (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 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:

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

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

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)

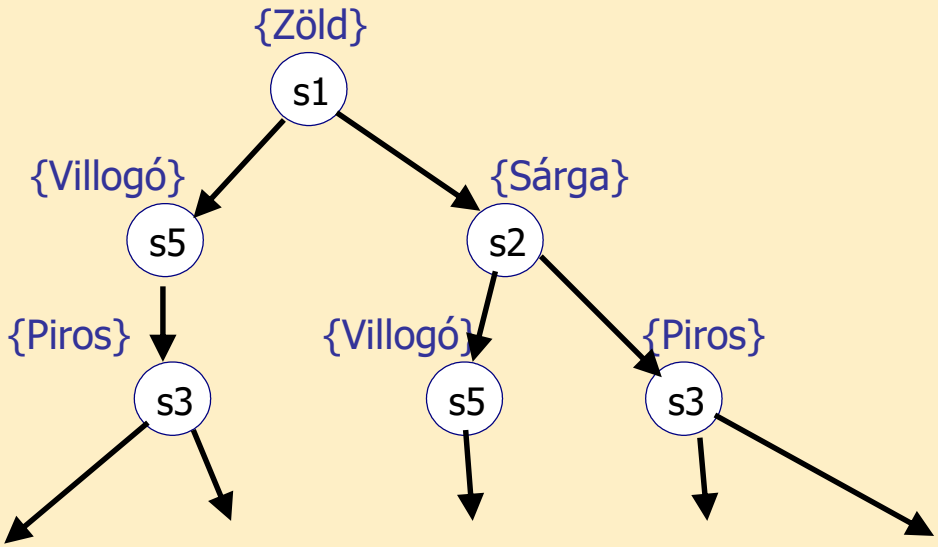
Ha α és β igaz, akkor α-nak igaznak kell maradnia mindaddig, amíg β is igaz.

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

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

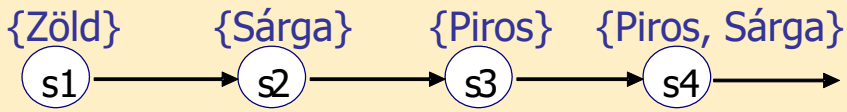
$$\mathbf{G}((\text{alarm} = \text{ON} \wedge \text{alert}) \rightarrow \text{X}(\text{safety} \, \mathbf{U} \, \neg\text{alarm}))$$

Elágazó idejű temporális logikák: CTL, CTL*

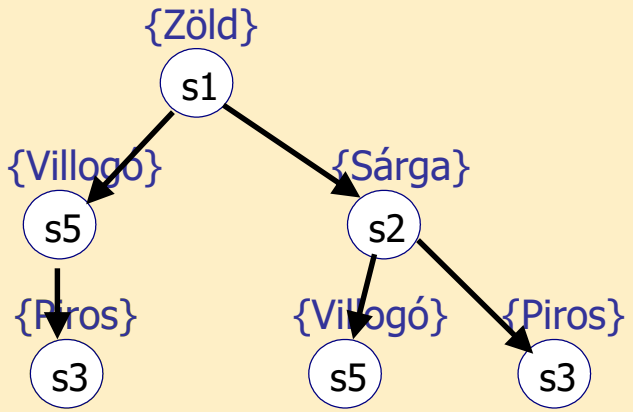


Ismétlés: Temporális logikák osztályozása

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

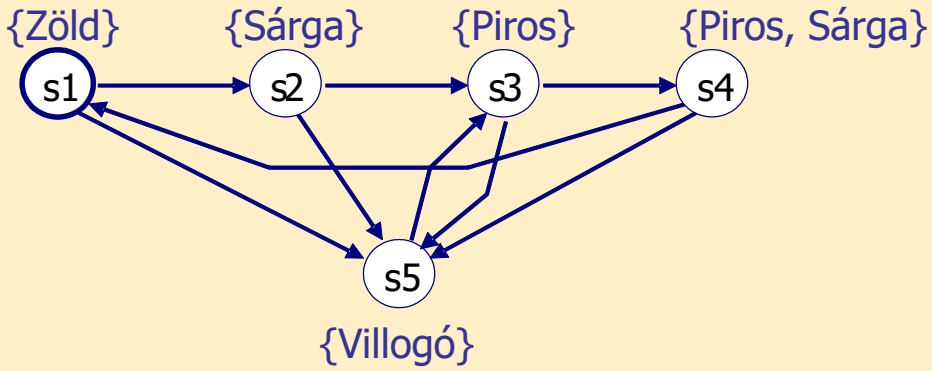


- Elágazó idejű:
  - A modell minden lehetséges végrehajtását tekintjük
  - Az állapotoknak több rákövetkezője lehet
  - Logikai idő elágazó idővonalak mentén jelenik meg (számítási fa)

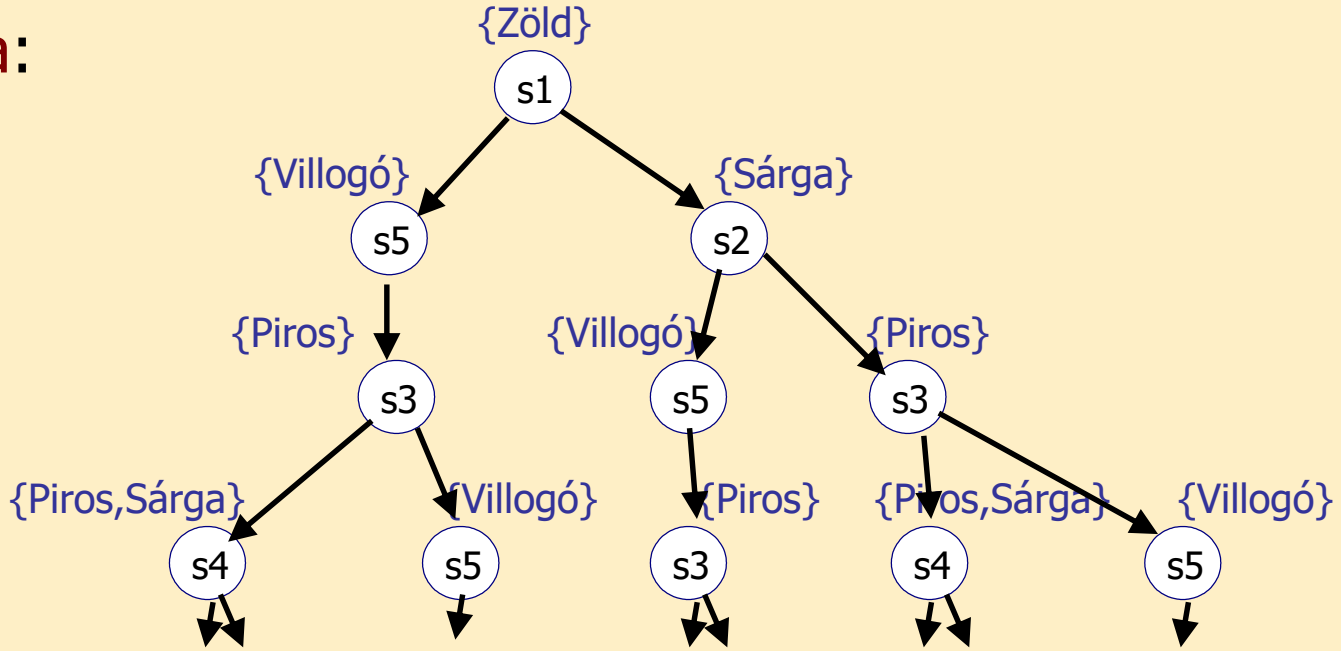


Számítási fa konstrukciója

Kripke-struktúra:



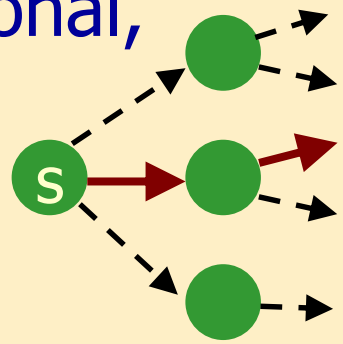
Számítási fa: Lehetséges elágazások



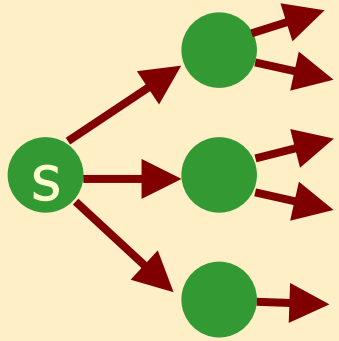
Elágazások vizsgálata

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

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



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



Elágazó idejű temporális logikák

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

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

CTL*: Computational Tree Logic *

$$A(p \Rightarrow F\ q)$$

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

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

CTL* operátorok (informális)

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

CTL* kifejezések

$$A(p \Rightarrow F\, q)$$

Minden útvonalra igaz, hogy …

amennyiben p fennáll az útvonal elején, …

akkor ezt a jövőben olyan állapot fogja követni …

amelyben, vagy ahonnan indulva található olyan állapot, amelyben q fennáll.

Példa CTL* kifejezések

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

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

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

  Létezik olyan útvonal, hogy
  - vagy ennek negyedik állapotán fennáll p,
  - vagy valamikor q fennáll az útvonalon

A CTL* formális kezelése

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

CTL* szintaxis

- Állapot-kifejezések: Állapotokon kiértékelhető
  - **S1**: Minden P atomi kijelentés egy állapot-kifejezés
  - **S2**: Ha p és q állapot-kifejezések, $\neg p$ és $p \wedge q$ is
  - **S3**: Ha p útvonal-kifejezés, akkor E p és A p állapot-kifejezések.
- Útvonal-kifejezések: Útvonalakon kiértékelhető
  - **P1**: Minden állapot-kifejezés útvonal-kifejezés
  - **P2**: Ha p és q útvonal-kifejezések, akkor $\neg p$ és $p \wedge q$ is
  - **P3**: Ha p és q útvonal-kifejezések, akkor X p és p U q is
- Érvényes CTL* kifejezések:
  A szabályok alapján generált állapot-kifejezések

CTL: Computational Tree Logic

AG EF p

AG AF p

AG (p $\Rightarrow$ AF q)

Háttér: Az ellenőrzés komplexitása

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

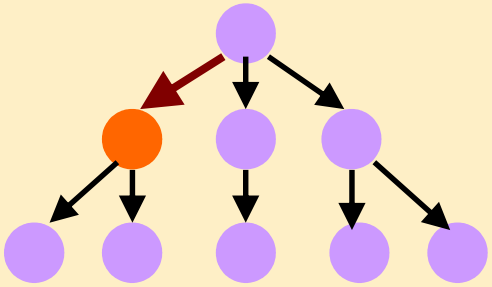
CTL operátorok (informális bevezető)

Állapotokon értelmezhető összetett operátorok:

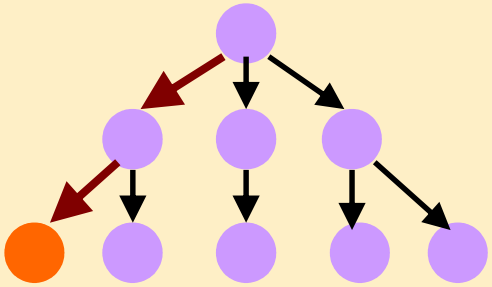
- EX p: létezik útvonal, aminek következő állapotán p
- EF p: létezik útvonal, aminek jövőbeli állapotán p
- EG p: létezik útvonal, aminek minden állapotán p
- E(p U q): létezik útvonal, amin p amíg q

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

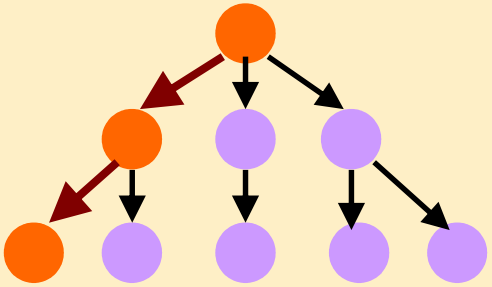
CTL operátorok (példák)



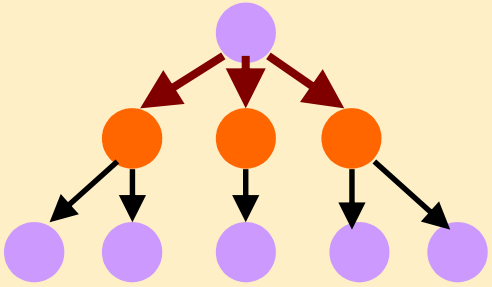
EX P



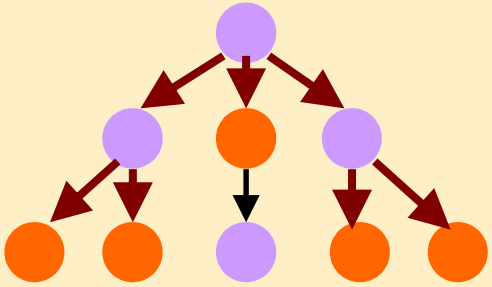
EF P



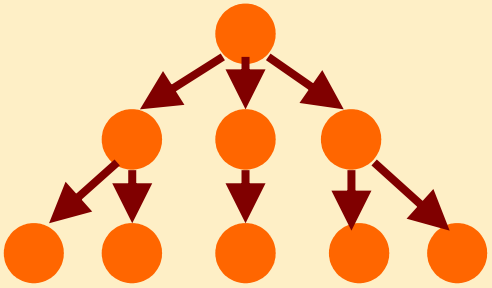
EG P



AX P



AF P



AG P

CTL kifejezések (példák)

- AG EF p

  Bárhonnan indulva olyan állapotba vihető a rendszer, ahol p igaz

  - Pl. AG EF Reset
- AG AF p

  Bárhonnan indulva mindenképpen eljutunk olyan állapotba, ahol p igaz

  - Pl. AG AF Terminated
- AG (p $\Rightarrow$ AF q)

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

  - Pl. AG (Request $\Rightarrow$ AF Reply)

CTL kifejezések (példák)

- EF AG p

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

- AF AG p

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

  - Pl. AF AG Normal

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

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

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

Követelmények formalizálása: Egy példa

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

Mintapélda (folytatás)

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

Mintapélda (folytatás)

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

P1 nincs a kritikus szakaszban

P1 van a kritikus szakaszban

P2 lép a kritikus szakaszba