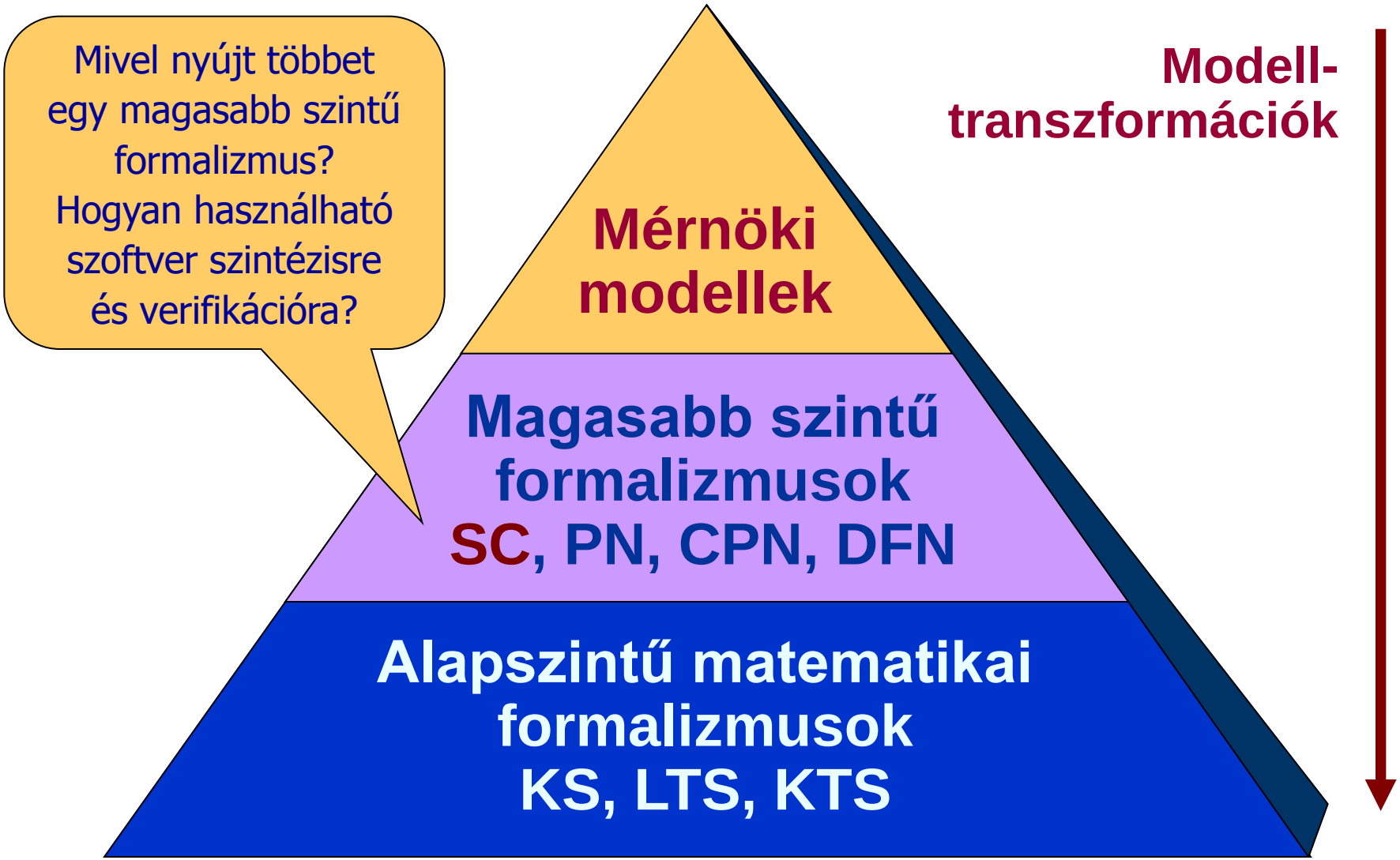
Magasabb szintű formalizmus: Állapottérképek (statecharts)

dr. Majzik István

BME Méréstechnika és Információs Rendszerek Tanszék

Modellek a formális ellenőrzéshez



Tartalomjegyzék

- Alapelemek
- Az állapottérkép szintaxisa
  - UML 2 statechart diagram
- Az állapottérkép szemantikája
  - UML 2 State Machine szemantika
  - (Más szemantika is lehet: pl. Harel-féle szemantika)
- Állapottérképek használata

Mi az állapottérképek célja?

- Állapot alapú, eseményvezérelt rendszerek viselkedésének megadására alkalmasak
  - Egy állapotgép viselkedésének leírása
  - Reaktív viselkedés:
    Külső események hatására történő állapotváltást ír le
    - Pl. bejövő üzenet, jelzés, hívás, …
  - Akciók: az állapotátmenetekhez rendelt tevékenységek, történések
    - Pl. értékadás, kimenő üzenet, …
- Szokásos használat:
  - Beágyazott rendszerek: bejövő események feldolgozása (pl. robot vezérlése, vagyonvédelmi rendszer, …)
  - Protokollok: üzenetek feldolgozása

Alapfogalmak

- Állapot, aktív állapot
  - Adott feltételek teljesülése (pl. művelet végrehajtható)
  - Állapotváltozók adott értékei
- Állapotátmenet
  - Állapot változása
  - Trigger esemény válthatja ki
    - Trigger nélküli átmenet: „önmagától” következik be
  - Őrfeltétel rendelhető hozzá
    - Állapotátmenet csak akkor történhet meg, ha az őrfeltétel igaz
  - Akció rendelhető hozzá
    - Állapotátmenethez rendelt tevékenység, történés
- Esemény
  - Aszinkron történés, paraméterei is lehetnek
  - Önálló elem, eseményosztály példánya
    - Öröklés: esemény attribútumok bővítése

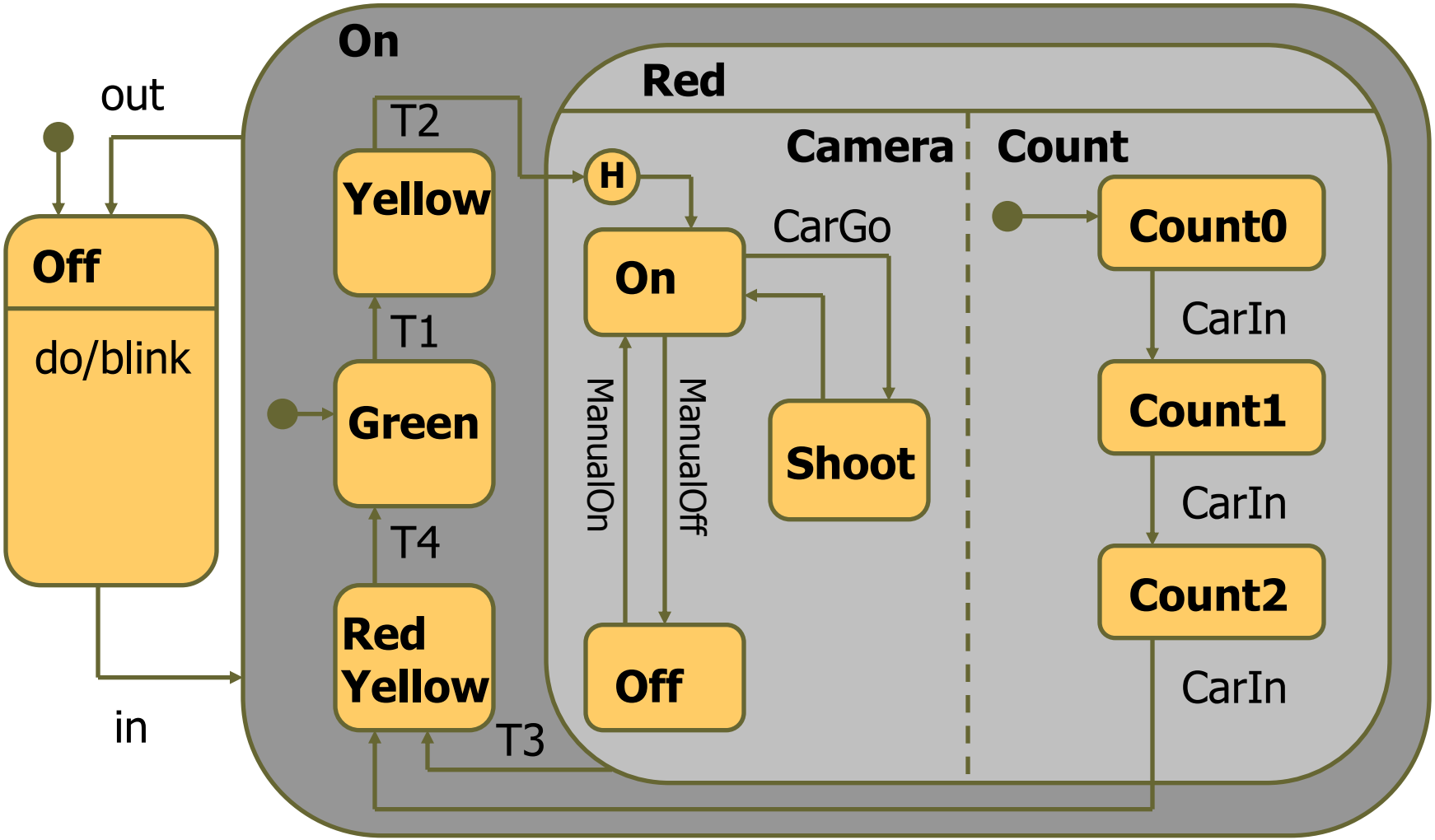
Új igények a könnyebb használat érdekében

- Állapotok finomítása: Állapothierarchia
  - Szuperállapot: alállapotok közös tulajdonságaihoz
- Konkurens viselkedés leírása
  - Nem akarunk sorrendi kötöttséget megadni (egyidejűleg, vagy tetszőleges sorrendben végzett feldolgozás)
  - Többszálú / elosztott / párhuzamos végrehajtás esetén
- Összetett állapotátmenetek
  - Szétváló, egyesülő, feltételtől függő elágazó átmenet
- Emlékezés: Visszatérés egy korábbi aktív állapotkonfigurációra
  - Visszatérés adott feldolgozáshoz közbenső esemény után
  - Egy állapotfinomítási szinten vagy mélyebben is

Állapotdiagramok és állapottérképek

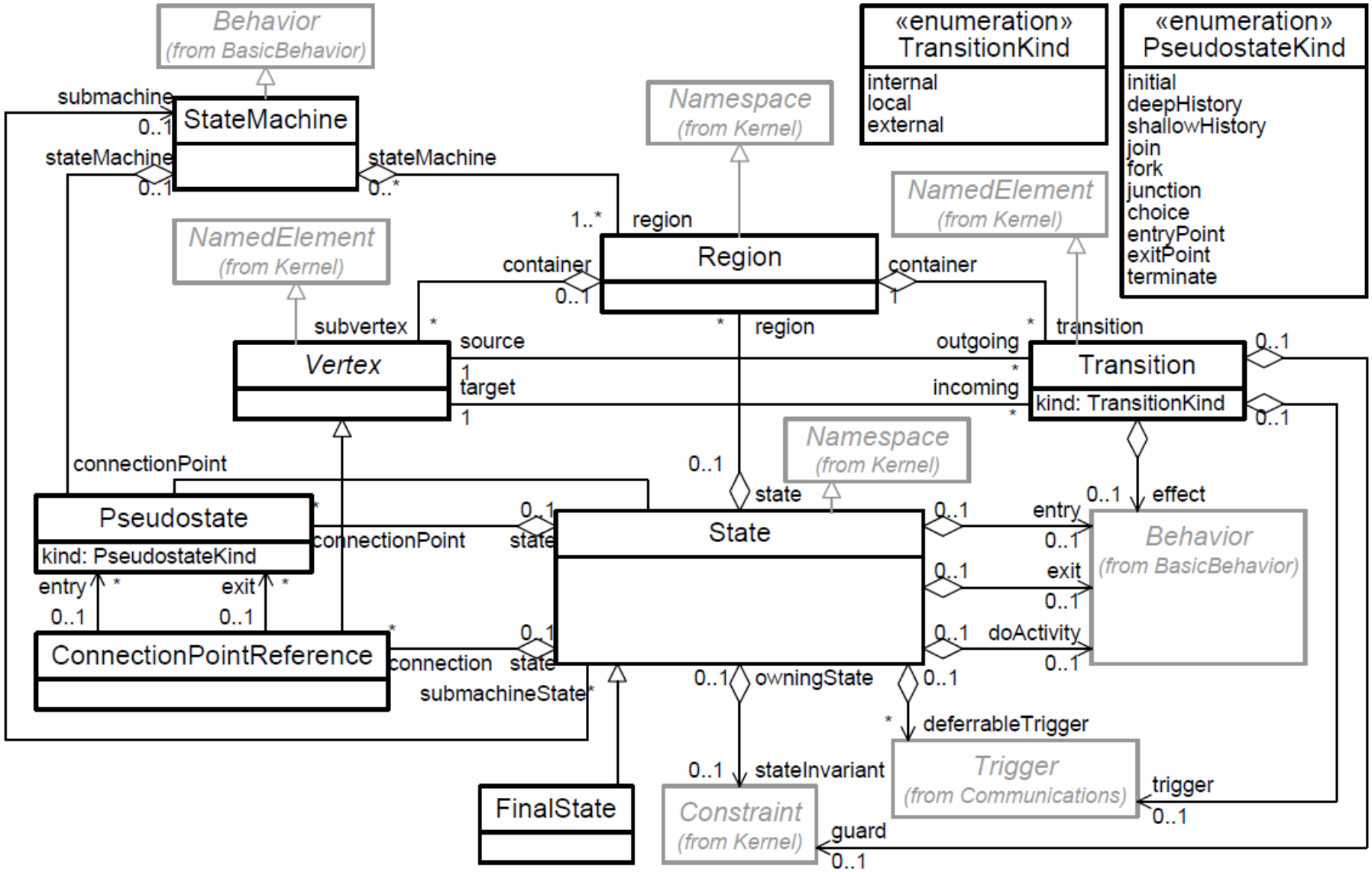
- Állapotdiagram:
  - Egyszintű, egyszerű állapotok és átmenetek
    - Pl. UPPAAL esetén látott automaták leírása
- Állapottérkép: az állapotdiagram kiterjesztése
  - Állapothierarchia: állapotok finomítása
  - Konkurens régiók: konkurens viselkedés leírása
  - Összetett átmenetek: szétváló, egyesülő, feltételes
  - Emlékezés: Legutolsó aktív állapotkonfiguráció tárolása
  - Szintaktikai segédelemek
  - Ritkán használt (nem intuitív) kiegészítések
    - Késleltetett esemény
    - Szinkronizációs állapot
    - …

Egy állapottérkép



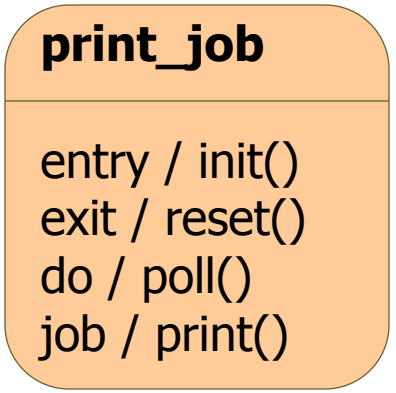
Az állapottérképek szintaxisa (UML szerinti szintaxis)

UML State Machine metamodell

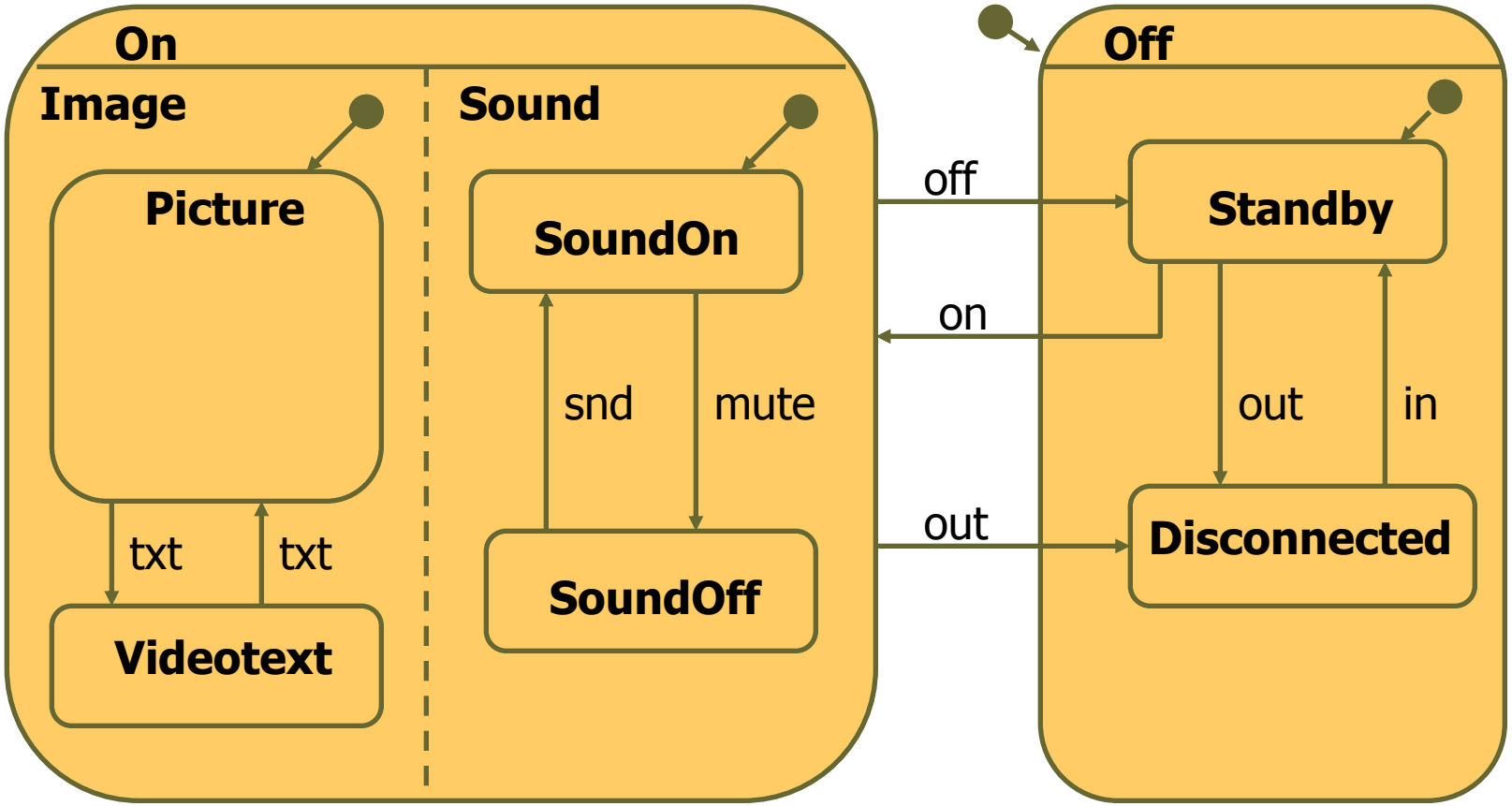


Állapotok: Akciók és állapotfinomítás

- Állapotok: Alapszintű modellelem
- Állapotokhoz kötődő akciók:
  - Belépési akció (entry / ...)
  - Kilépési akció (exit / ...)
  - Belső aktivitások (do / ...)
- Állapotfinomítás
  - Egyszerű állapot: nincs finomítása
  - OR jellegű finomítás: alárendelt alállapotok
    - Ezek közül egyszerre egy állapot lehet aktív
  - AND jellegű finomítás: konkurens régiók
    - Egyidejűleg minden egyes régióban kell legyen aktív állapot!



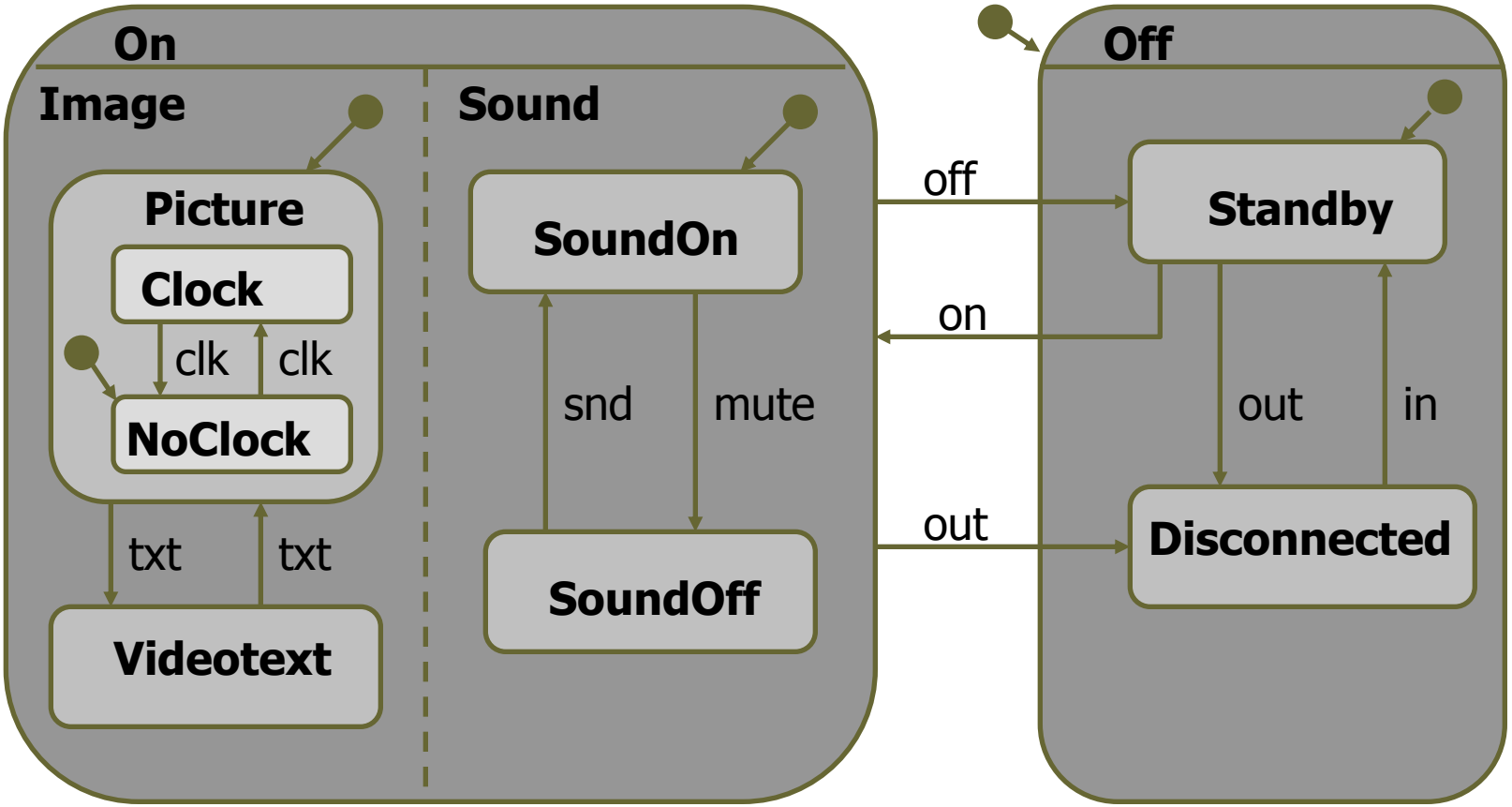
Példa: Állapotfinomítás



AND jellegű finomítás

OR jellegű finomítás

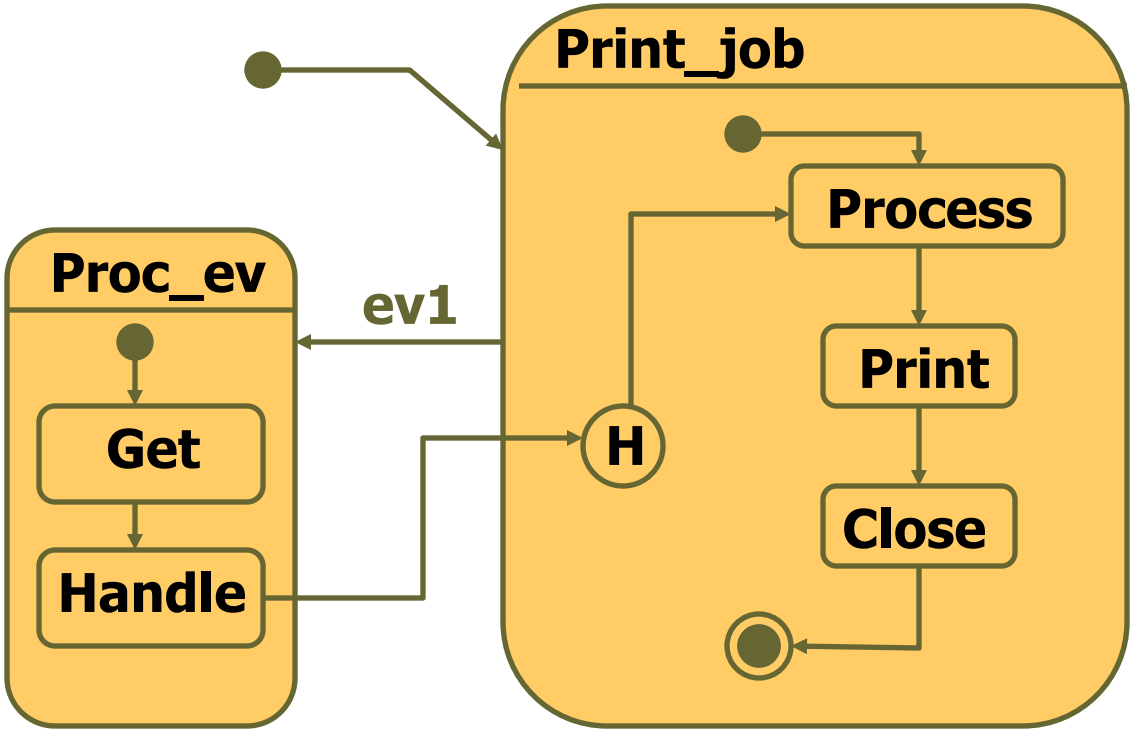
Példa: Állapotfinomítás



Pszeudo-állapotok

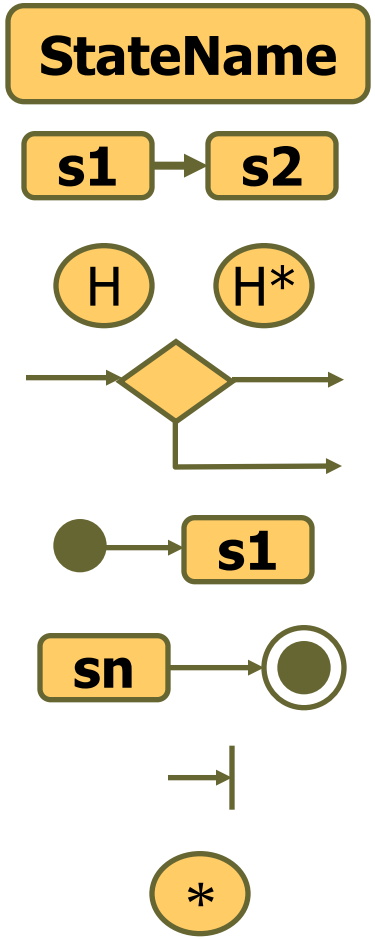
- Kezdőállapot jelzés: régióba való belépéskor lesz aktív 
  - Minden OR finomításban egy
  - Minden AND régióban egy
- Végállapot jelzés: állapotgép terminálás 
- Emlékező állapotok (history state)
  - A legutolsó aktív állapotkonfigurációt „tárolja”
    - Egyszerű emlékező állapot: csak az adott finomítási szinten 
    - Mélyen emlékező állapot: a mélyebb finomítási szinteket is 
  - Mit jelent az emlékező állapothoz húzott bemenő átmenet?
    - Tüzelésekor a „tárolt” állapotkonfigurációba kerül az objektum
    - Az emlékező állapot egy „hivatkozás” a tárolt állapotkonfigurációra
  - Mit jelent az emlékező állapotból húzott kimenő átmenet?
    - Alapértelmezett „tárolt” állapotot jelöl ki arra az esetre, ha előzőleg még nem volt aktív állapot

Példa: Emlékező állapot



Állapottérképek elemei

- Állapot
- Állapotátmenet
- Emlékező állapot
- Feltétel
- Kezdőállapot
- Végállapot
- Állapotcsonk
- (Szinkronizáció)



(Állapot)átmenetek

- Állapotátmenetek megadása
- Szintaxis:

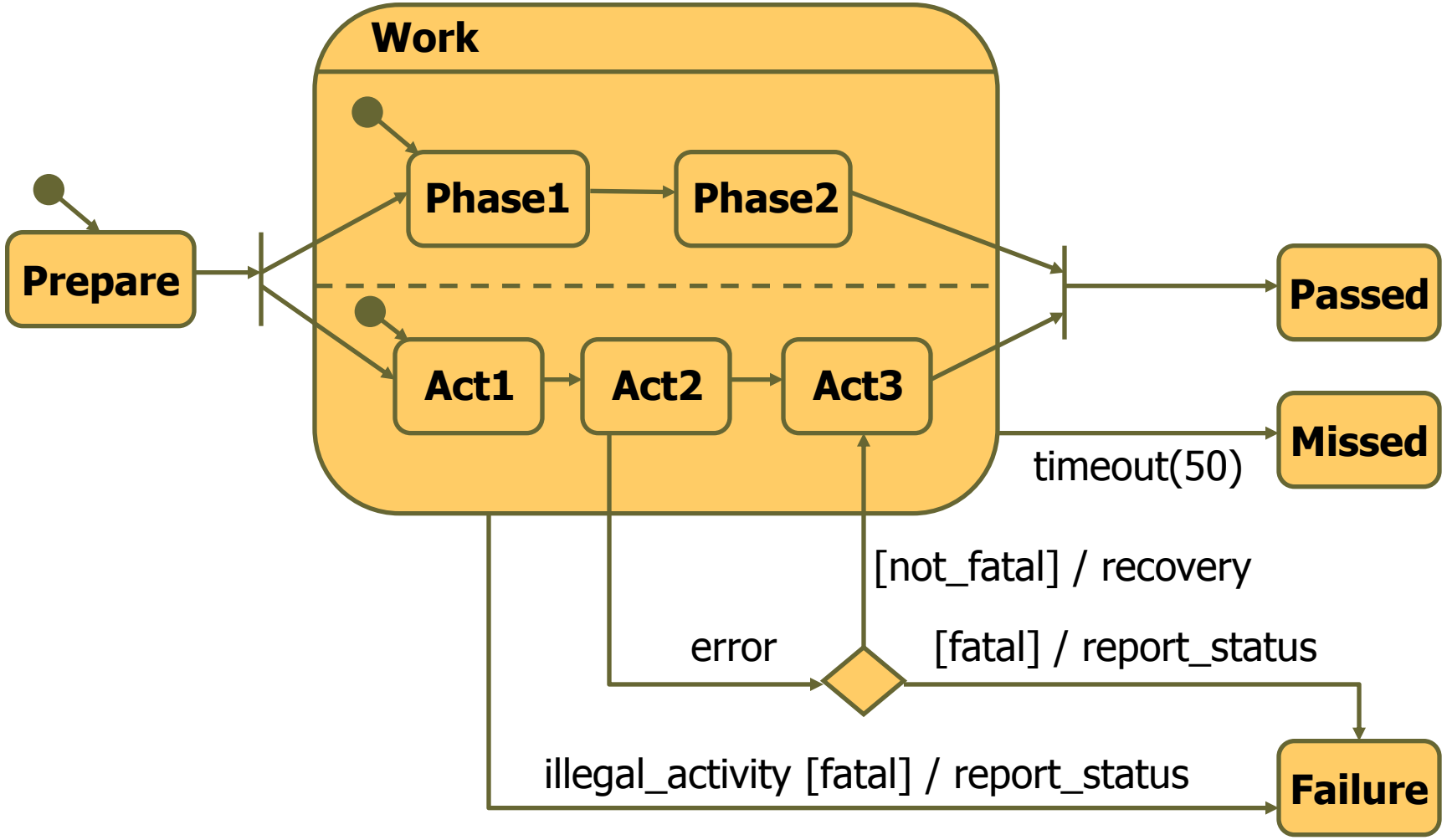
```
trigger [guard] / action
```

  - **`trigger`**: kiváltó esemény
  - **`guard`**: az átmenet őrfeltétele
    - Predikátum az állapotváltozók és esemény paraméterek felhasználásával
    - Állapotra való hivatkozás is lehet: is_in(state)
  - **`action`**: akció (művelet)
    - akció szemantika: művelet részletezése

Átmenetek specialitásai

- Time-out mint trigger:
  - Fennáll, ha az objektum az átmenethez tartozó kiindulási állapotban tartózkodik végig az adott időintervallumban
- Összetett átmenetek:
  - Szétváló (fork): konkurens régiókban lévő állapotokba való együttes belépés
  - Egyesülő (join): konkurens régiókban lévő állapotokból való együttes kilépés
  - Elágazó (condition): több, őrfeltételtől függő átmenet egyszerűsített jelölése (szegmensek)
- Állapothierarchián átívelő átmenetek
  - Megengedett (bár nem elegáns)

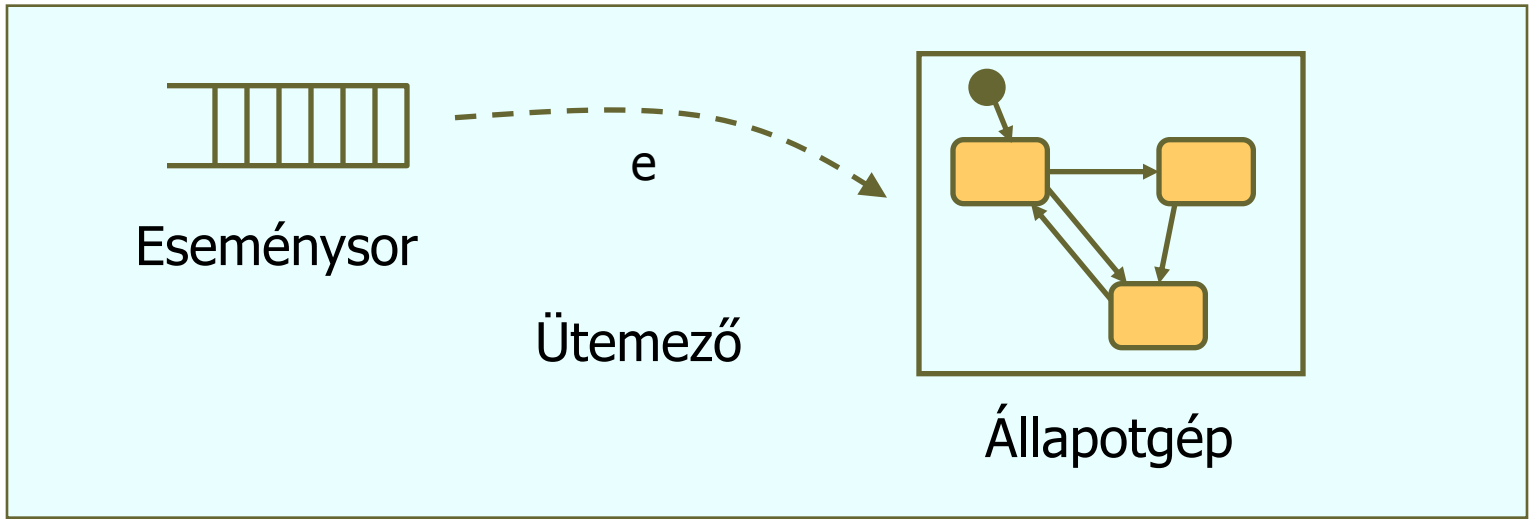
Példa: Állapotátmenetek



Az állapottérképek informális szemantikája (UML szerinti szemantika)

Szemantika: Hogyan működik?

- Alapelemek:
  - Állapotgép: Az állapottérkép írja le a viselkedését
  - Eseménysor + Ütemező: „Futtató rendszer” (az állapotgép szempontjából külső elemek)



Mit ad meg a szemantika?

Mit tesz az állapotgép egy esemény hatására
$\rightarrow$ állapotgép egy lépése

- Állapotátmenetek tüzelnek
  - Újdonság: egy esemény hatására több konkurens állapotátmenet tüzelhet
- Állapotkonfiguráció változik
  - Több aktív állapot lehet
    - Aktív állapot minden régiójában kell legyen egy aktív állapot
    - Aktív állapot OR finomításában kell legyen egy aktív állapot
  - Egy OR finomításban illetve egy régióban csak egy aktív állapot lehet
  - Rekurzívan érvényes

A szemantika alaptulajdonságai

- Egyenként feldolgozott események
  - Az ütemező akkor küld újabb eseményt, ha az előző feldolgozása teljes egészében megtörtént
    - Stabil állapotkonfiguráció kialakult: nincs trigger nélküli átmenet
- Események teljes feldolgozása (run to completion)
  - Átmenetek maximális halmaza tüzel
    - Minden engedélyezett átmenet tüzel, kivéve ha ezt konfliktus megakadályozza
  - Ezek tüzelése után dolgozza fel a következő eseményt
- Az eseményfeldolgozás a szemantika lényege
  - Ez alapján lehet programkód alakjában megvalósítani az állapotgépet (forráskód generálás)

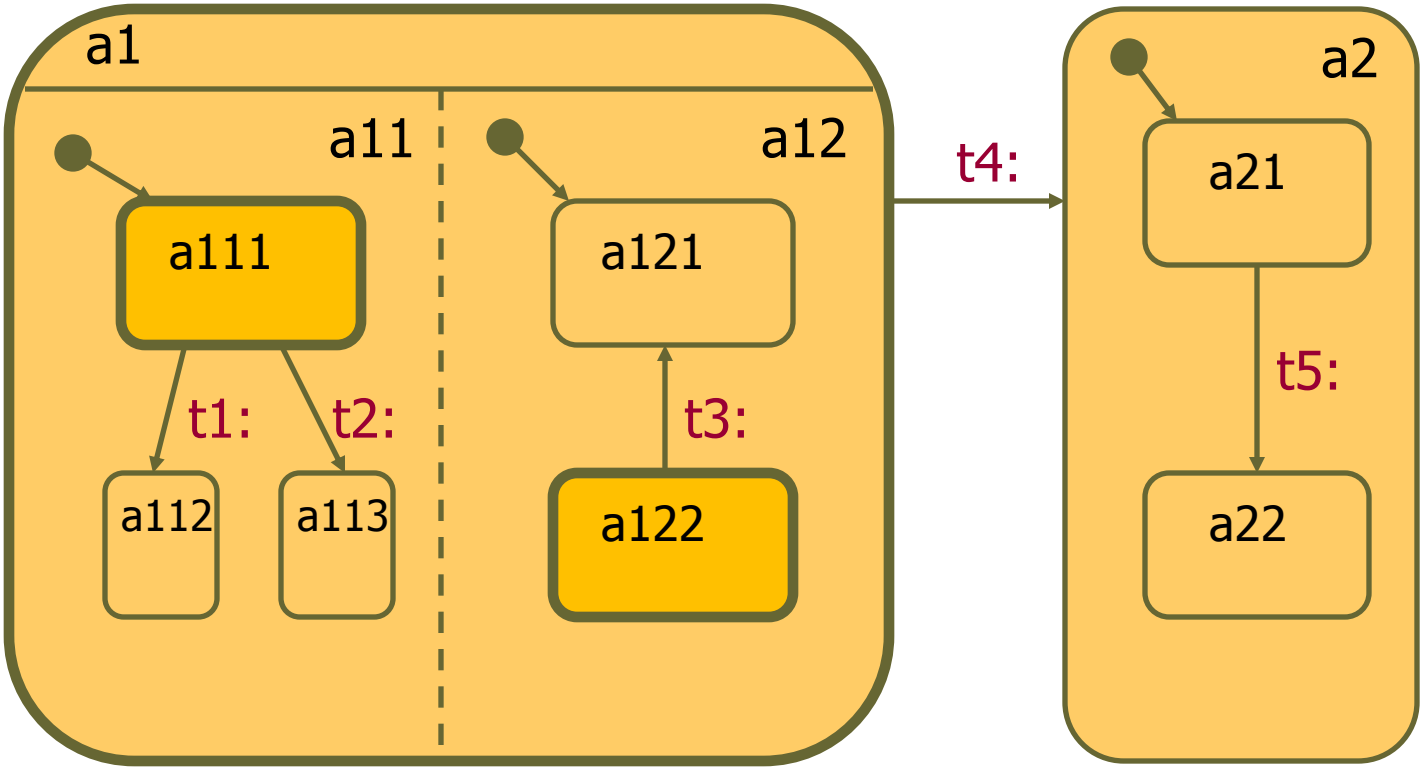
Az eseményfeldolgozás lépései 1/4

- Külső feltétel: Ütemező a stabil konfigurációban lévő állapotgépnek egy eseményt juttat
- Engedélyezett átmenetek:
  - Kiindulási (forrás-) állapot aktív
  - A kiválasztott esemény az átmenet triggere
  - Az őrfeltételek teljesülnek

  Esetek az engedélyezett átmenetek száma alapján:
  - Ha csak egy van: Tüzelhet!
  - Ha nincs: Halasztott-e az esemény?
    - Igen: tárolás, új esemény kérése az ütemezőtől
    - Nem: esemény eldobható (hatás nélküli)...
  - Ha több van: Tüzelő átmenetek kiválasztása szükséges
    - Befolyásol: A konfliktus

Példa: Konfliktus

A t1, …, t5 átmenetek ugyanazon e eseményre triggereltek.
Melyek nem tüzelhetnek együtt?



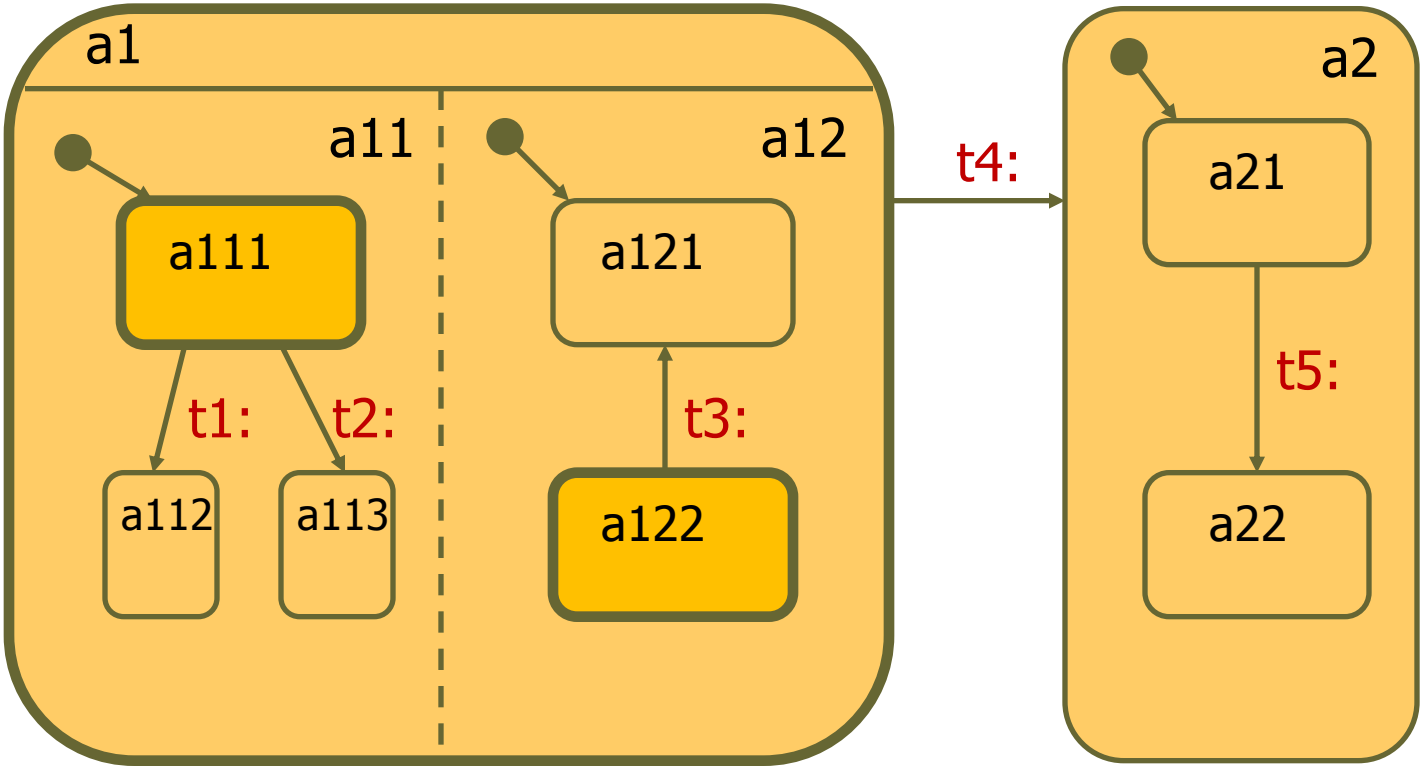
- Nem engedélyezett: t5 (forrásállapota nem aktív)
- Egyszerre nem tüzelhetnek: (t1,t2); (t1,t4); (t2,t4); (t3,t4)
- Egyszerre is tüzelhetnek: (t1,t3); (t2,t3);

Eseményfeldolgozás lépései 2/4

- Tüzelő átmenetek kiválasztása:
  - Maximális számú átmenet, nem lehet közöttük konfliktus
    - Konkurens átmenetek szimultán tüzelése
- Konfliktusban lévő átmenetek:
  - Ugyanazt az állapotot hagyják el, pontosabban az elhagyott állapothalmazok metszete nem üres
- Konfliktus feloldása:
  - Prioritás alapján: egy átmenet prioritása nagyobb, ha az átmenet kiindulási állapota az állapothierarchiában (finomításban) alacsonyabb szintű
    - OO koncepció: a finomítás „felüldefiniál”
  - Véletlenszerű választás, ha azonos prioritásúak

Példa: Konfliktusfeloldás

A t1, …, t5 átmenetek ugyanazon e eseményre triggereltek.
Melyek tüzelhetnek együtt?

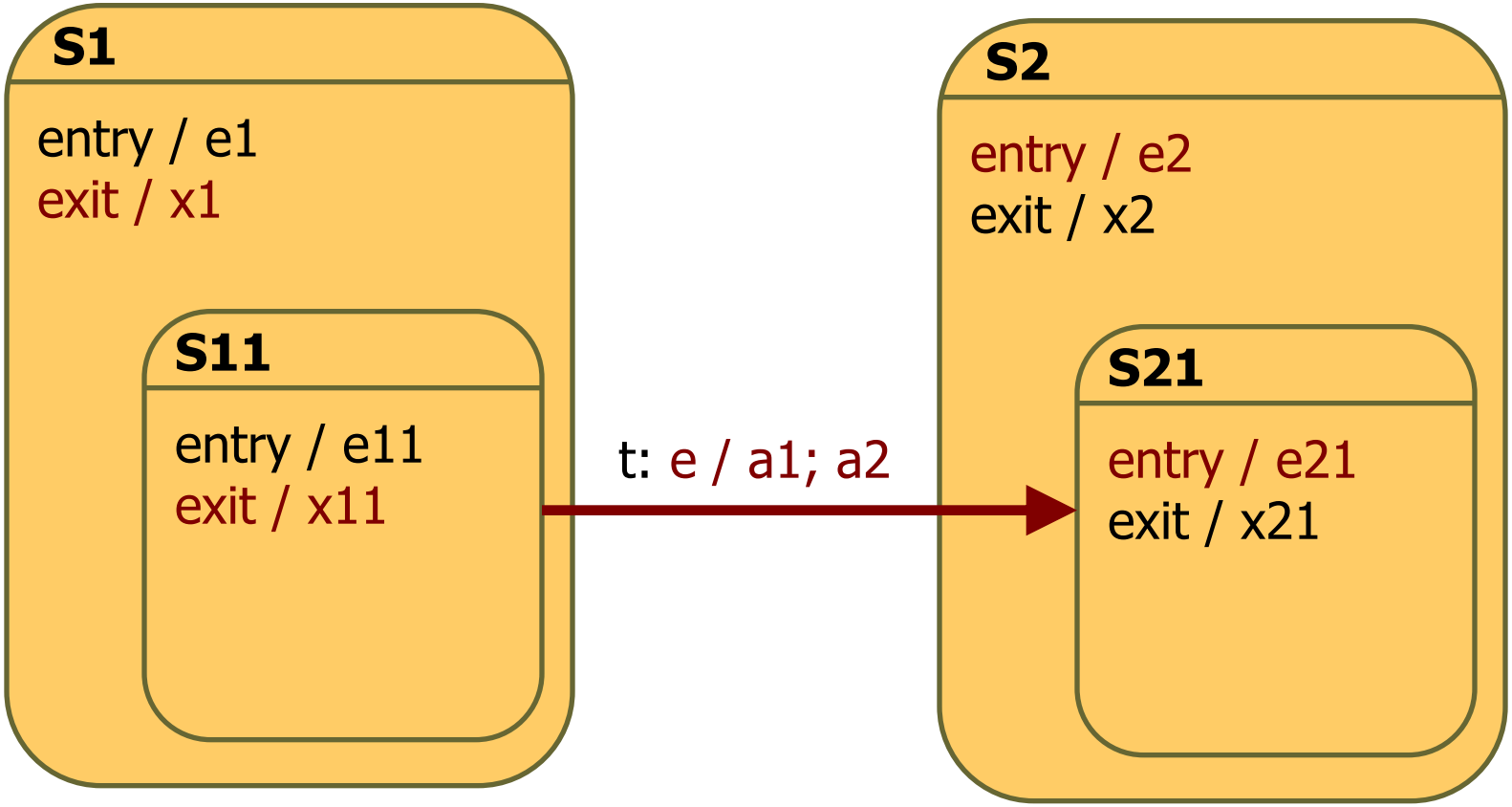


- Nagyobb prioritású t4-nél: t1, t2 és t3
- Tüzelhet: (t1,t3) vagy (t2,t3)

Eseményfeldolgozás lépései 3/4

- Kiválasztott átmenetek tüzelnek:
  - Sorrend véletlenszerű (nincs köztük konfliktus)
  - Akcióik közötti sorrend is véletlenszerű
- Egy átmenet tüzelése és akciói:
  1. Kiindulási állapotok elhagyása
     - Alacsonyabb hierarchiaszinten először
     - Kilépési akciók végrehajtása (exit akciók)
  2. Átmenetek akcióinak végrehajtása
  3. Célállapotokba való belépés $\rightarrow$ új konfiguráció
     - Magasabb hierarchiaszinten először
     - Belépési akciók végrehajtása (entry akciók)

Példa: Akciók sorrendezése

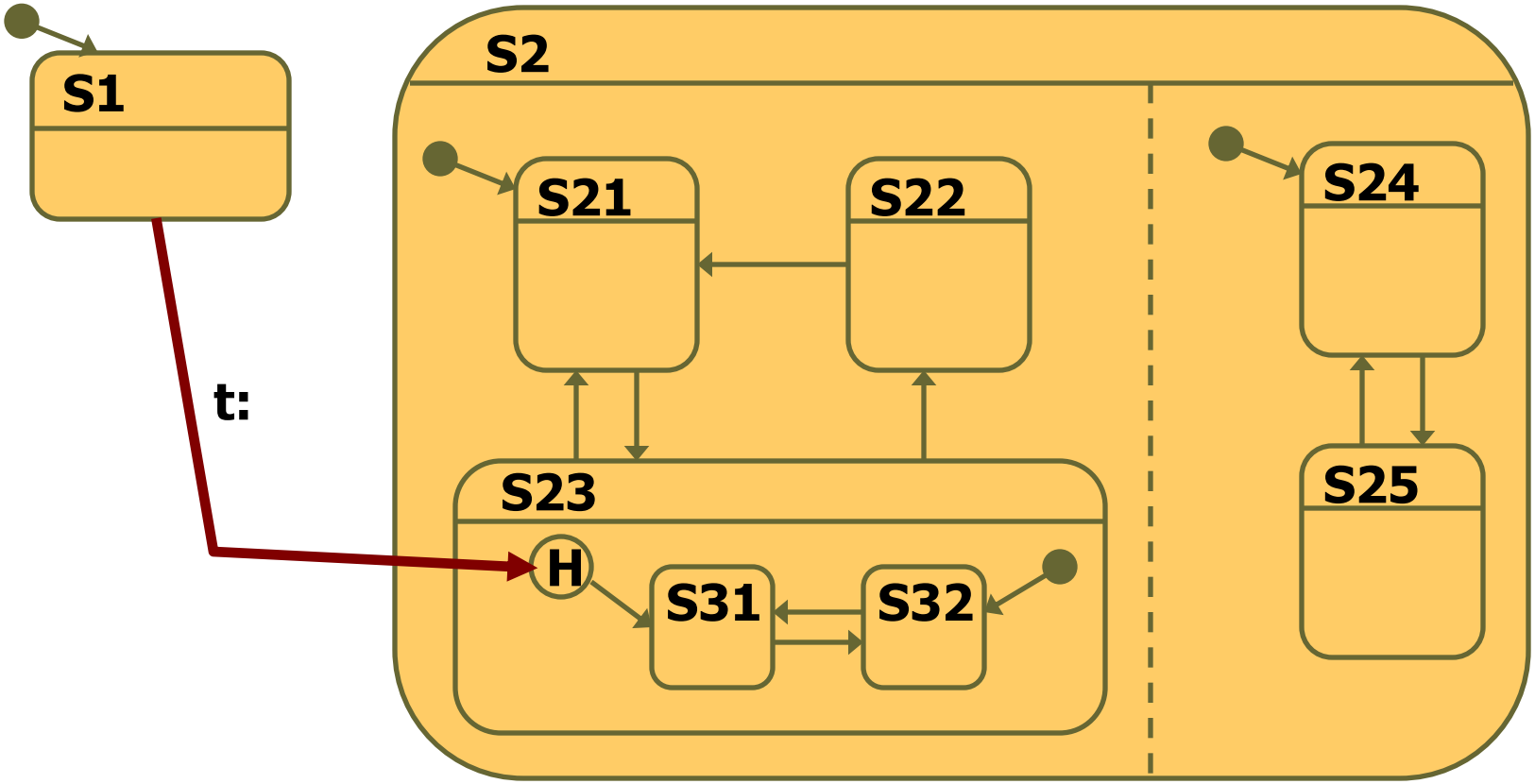


Akciók sorrendje: x11; x1; a1; a2; e2; e21

Eseményfeldolgozás lépései 4/4

- Belépés új konfigurációba különféle jellegű célállapotok esetén:
  - Ha egyszerű (nem finomított) a célállapot:
    - Új konfiguráció része lesz
    - Ős állapotai (amelyek finomításában szerepel) is aktívak lesznek
    - Aktívvá váló ős állapotok minden régiójában lesz aktív állapot (kezdőállapot jelöli ki)
  - Ha OR finomítása van a célállapotnak:
    - A finomításban a kezdőállapot jelöli ki az aktív állapotot
  - Ha AND finomítása van a célállapotnak:
    - Minden régiójában kell legyen aktív állapot (kezdőállapot jelöli ki)
  - Ha emlékező állapot:
    - Az utoljára elhagyott állapotkonfiguráció lesz újra aktív
    - Ha nem volt még ilyen: a kimenő él határozza meg
  - Ha nem stabil az állapot: azonnali továbblépés

Példa: Belépés konkurens állapotba

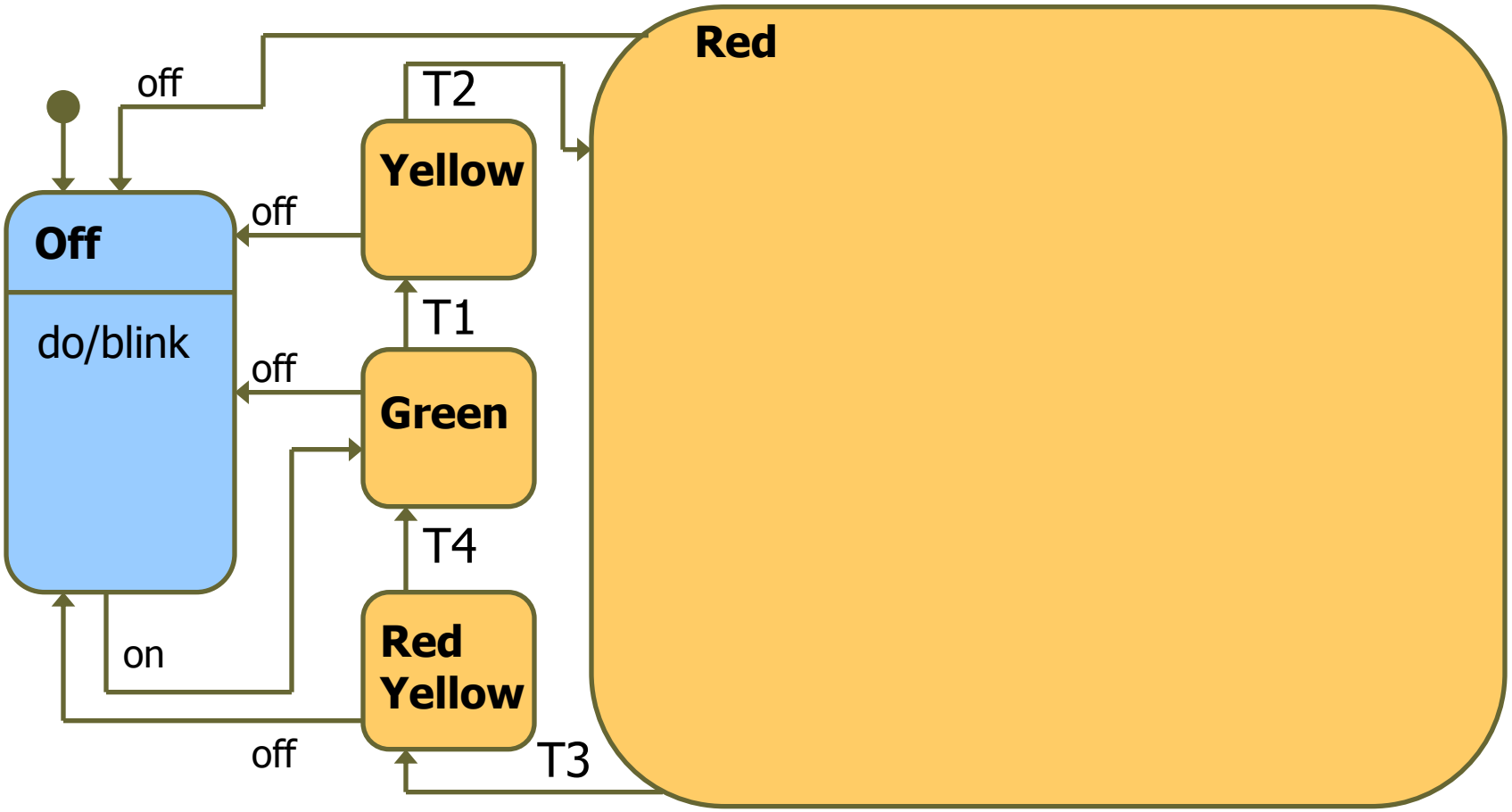


A **t** átmenet tüzelése után mi lesz az új állapotkonfiguráció?

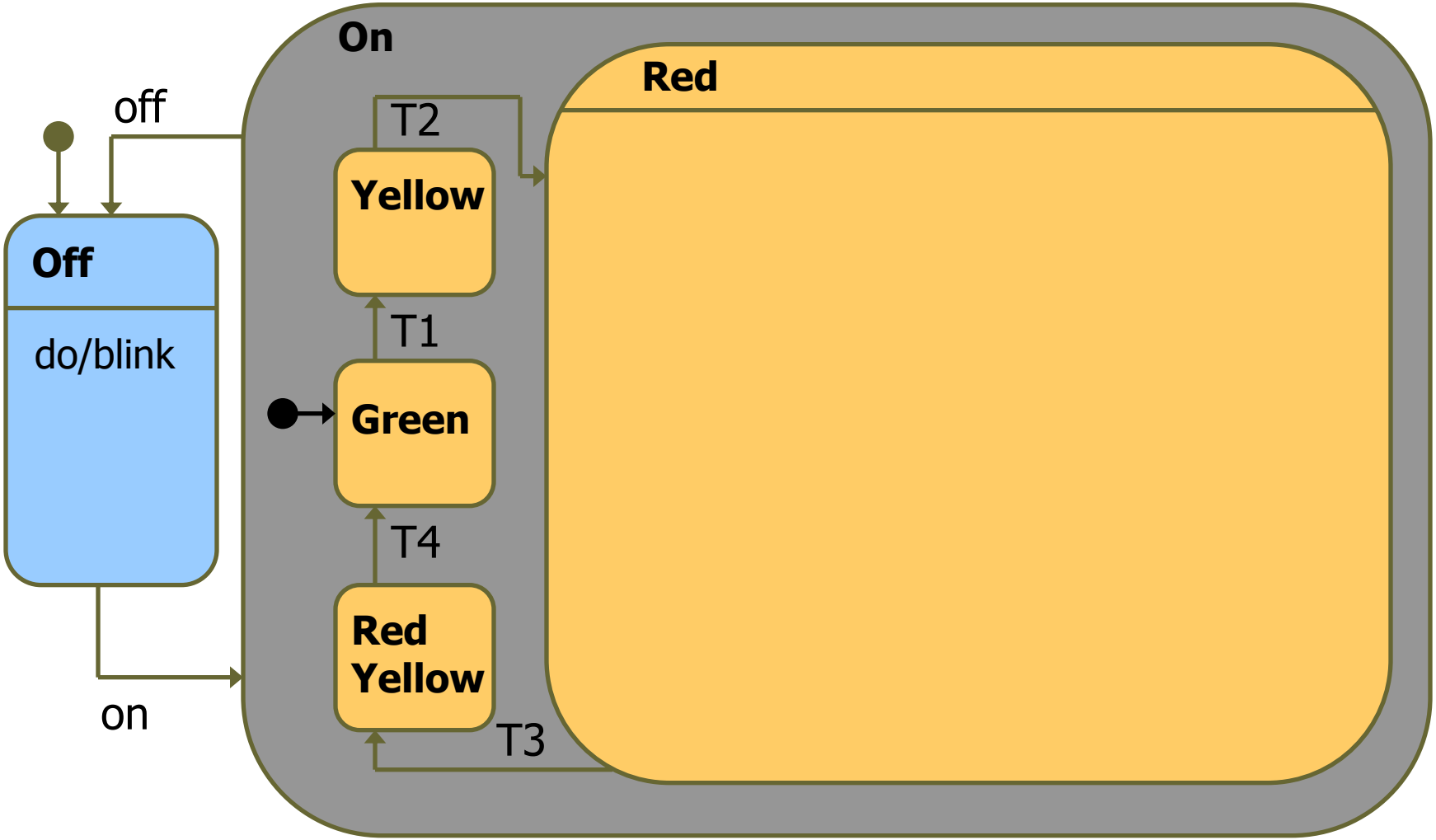
Modellezési mintapélda

- Közlekedési lámpa vezérlője egy főútvonal és egy mellékútvonal kereszteződésében
  - Bekapcsoláskor: kezdetben főútvonal számára zöld
  - Kikapcsoláskor: villogó sárga
  - Zöld, sárga, piros váltás: időzítő eseményre
  - Főútvonalon 3 várakozó: időzítőtől függetlenül zöld jelzés szükséges
  - Főútvonalon tilosban áthajtók: fényképezés
    - Ez a funkció kézzel ki-be kapcsolható

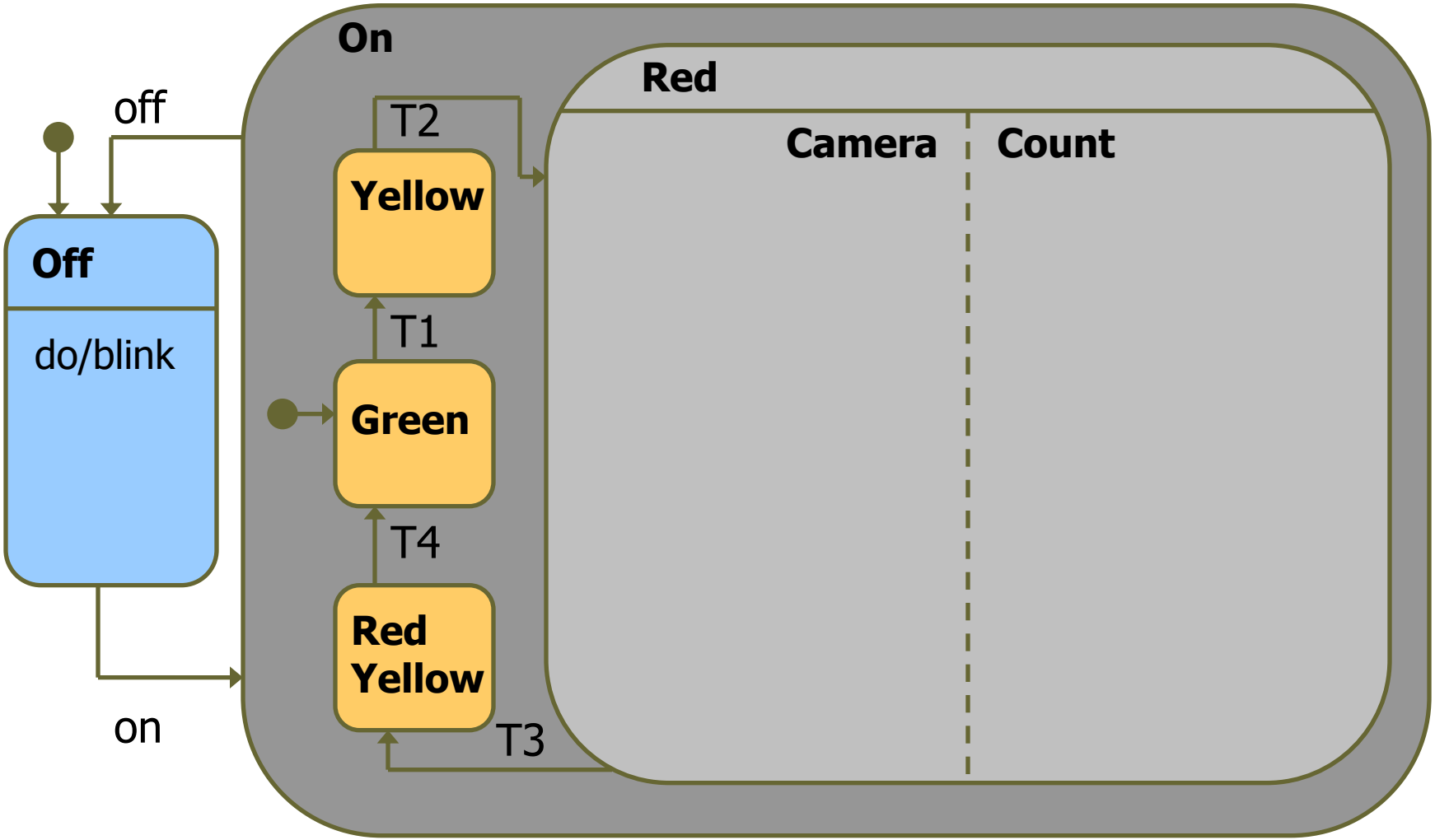
1. Lámpaváltás



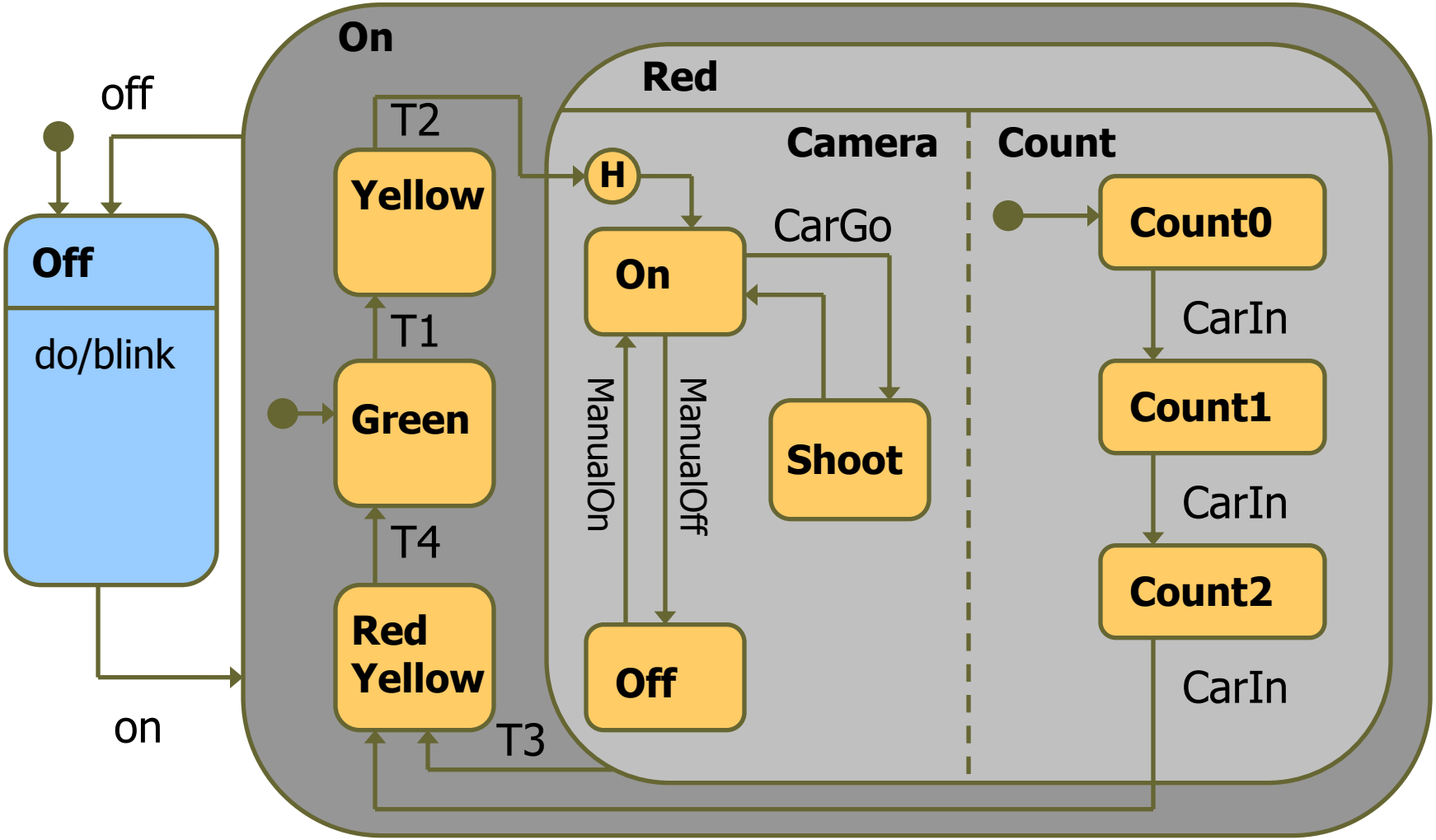
2. Állapothierarchia



3. Konkurens alállapotok



4. Teljes vezérlő



Állapottérképek szerepe az UML 2 esetén

- Állapot alapú, eseményvezérelt működés leírása
  - Aktív objektumok viselkedésének leírása
- Felhasználás:
  - Viselkedés szintű kódgenerálás
  - Viselkedés verifikációja
- Akciók formalizálása: UML 2 Action Semantics
  - Metódushívás
  - Attribútum olvasás, írás
  - … (sokféle művelet)
  - Színezett Petri-hálókhoz (ld. később) hasonlító alapelvek

Állapottérképek alapjai (összefoglalás)

- Kiterjesztések
- Állapottérkép szintaxis
  - Állapothierarchia, konkurens régiók, emlékező állapotok
  - Összetett átmenetek
- Állapottérkép (nem formális) szemantika
  - Átmenetek engedélyezettsége
  - Tüzelő átmenetek kiválasztása
  - Átmenetek tüzelése
  - Új állapotkonfiguráció kialakulása
- Állapottérkép eszközök
  - UML 2 támogató eszközök
  - Yakindu Statechart Tools (statecharts.org)
  - Quantum Programming (state-machine.com)

Mit kezdhetünk az állapottérképekkel?

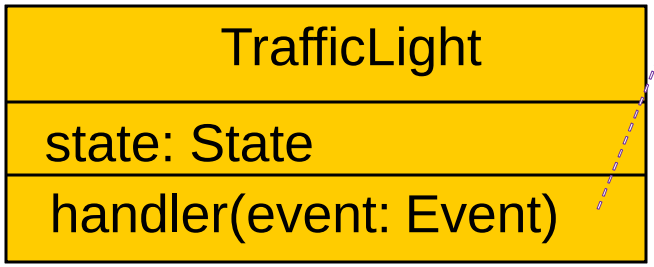
- Forráskód generálása
  - Több megvalósítási minta
- Modellellenőrzés
  - PLTL „testreszabható” állapottérkép modellekre
  - Alsóbb szintű modellre való leképzéssel is ellenőrizhető
- Tesztek generálása
  - Modellellenőrzővel megvalósítható (ld. korábban)
- Futási idejű verifikáció: Monitorkód generálása
  - Állapottérkép mint referencia
    (helyes vezérlési folyamat megadása az ellenőrzéshez)

Forráskód szintézis állapottérkép specifikáció alapján

Megvalósítási minták

- Cél:
  - Vezérlési struktúra szintézise
  - Akciók részletezése: Akció nyelv szükséges
- Többféle megvalósítási minta
  - Kettős elágazás
  - State, Hierarchical State Machine
  - Extended Hierarchical Automata
- Különbségek
  - Állapottérkép elemkészlet támogatása
  - Memóriafoglalás
  - Sebesség (eseményfeldolgozás, vezérlési struktúra)

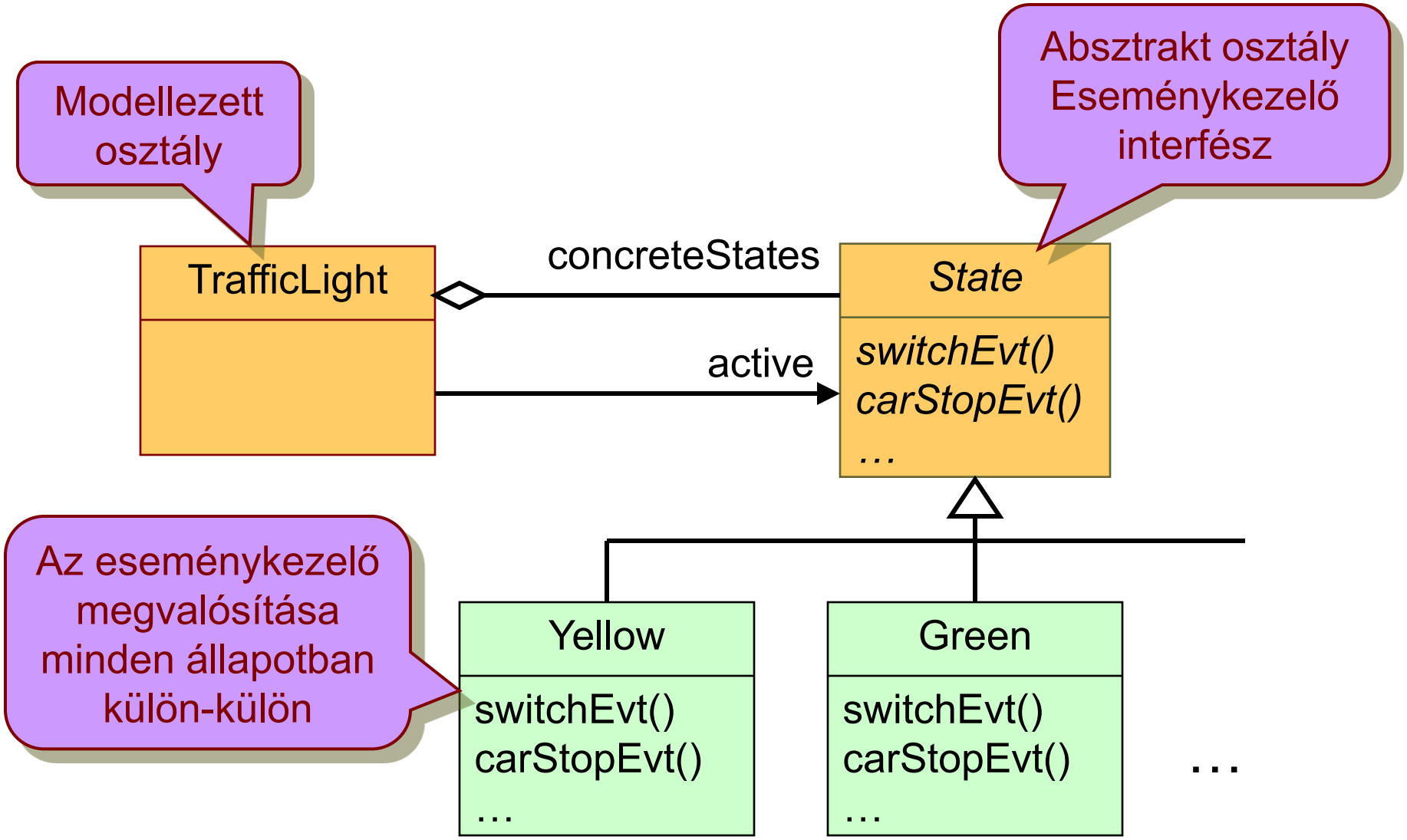
„Kettős elágazás” megvalósítási minta



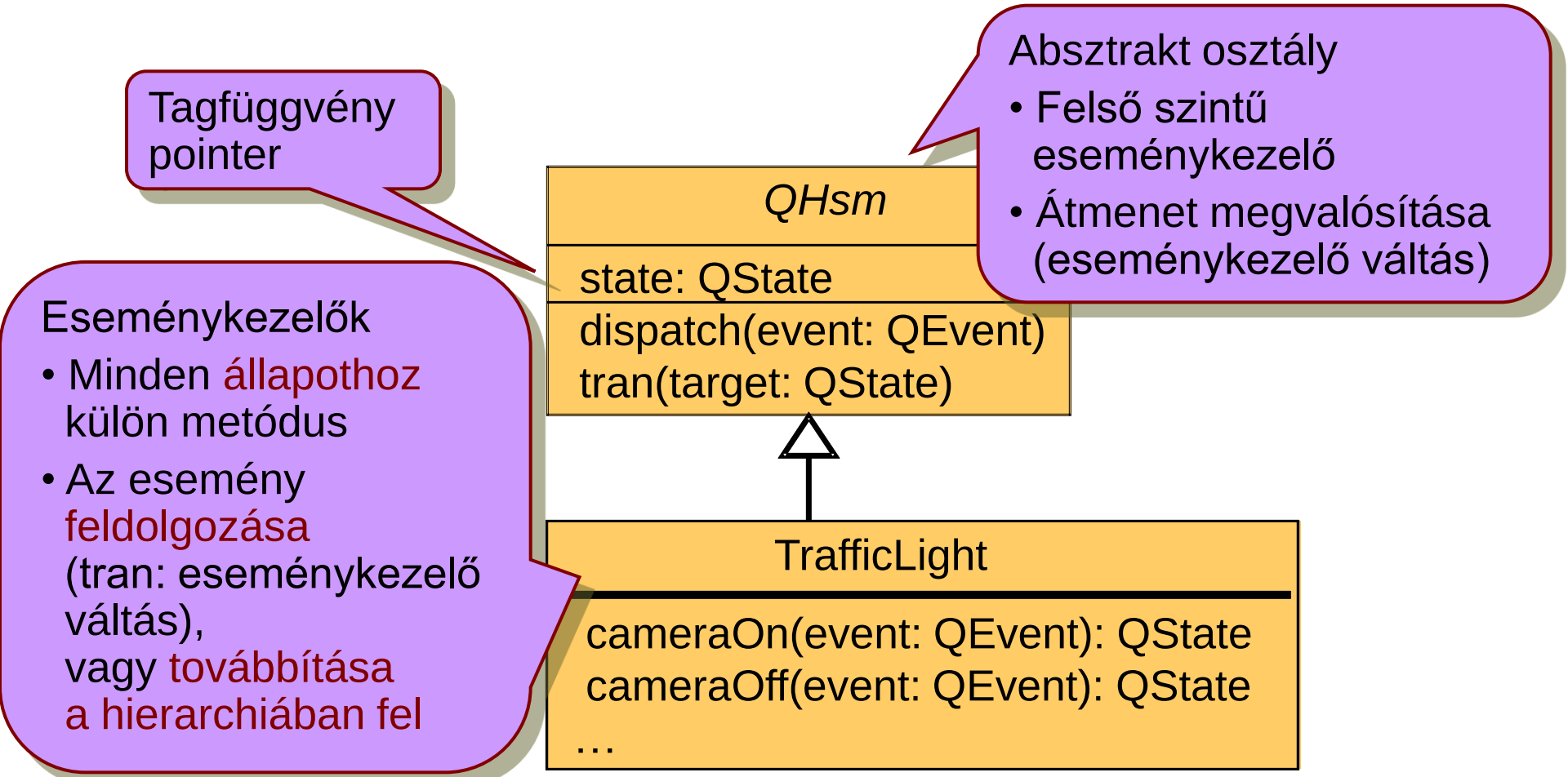

```
switch (state) {
  case GREEN:
   switch (event) {
     case tGreen:
       /* Yellow – entry */
       state = YELLOW;
       break;
     …
  }
  …
}
```

- Egyszerű és hatékony
- Állapothierarchiát és konkurens viselkedést nem támogat

A „State” megvalósítási minta (Gamma, 1994)



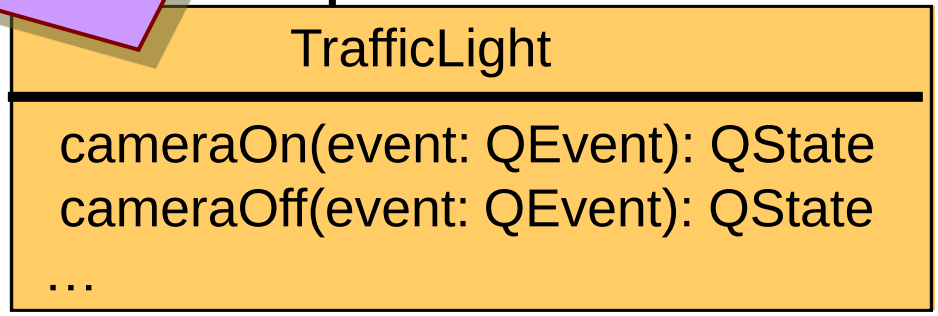
„Hierarchikus állapotgép” minta (Samek, 2002)



- OR jellegű finomítást (állapothierarchia) támogat
- Konkurens viselkedést nem támogat

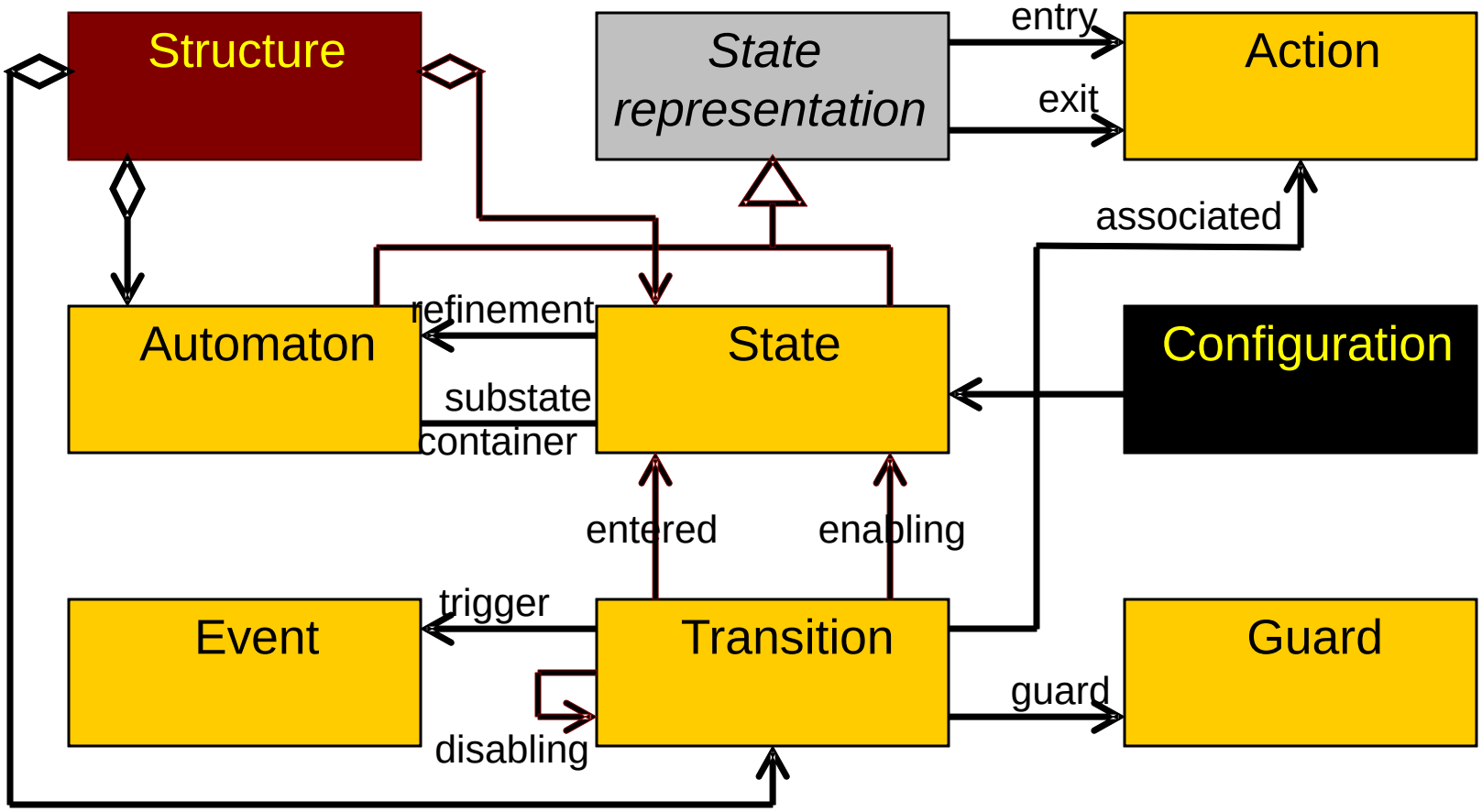
„Hierarchikus állapotgép” minta (Samek, 2002)

```
QState TrafficLight_CameraOn (TrafficLight *me, QEvent const *e) {
        switch (e->sig) {       /* demultiplex events based on signal */
                case ManualOff:
                        tran(TrafficLight_CameraOff); /* target state */
                        return 0; /* event handled */
                …
        }
        return (QState)TrafficLight_Red; /* designate the superstate */
}
```



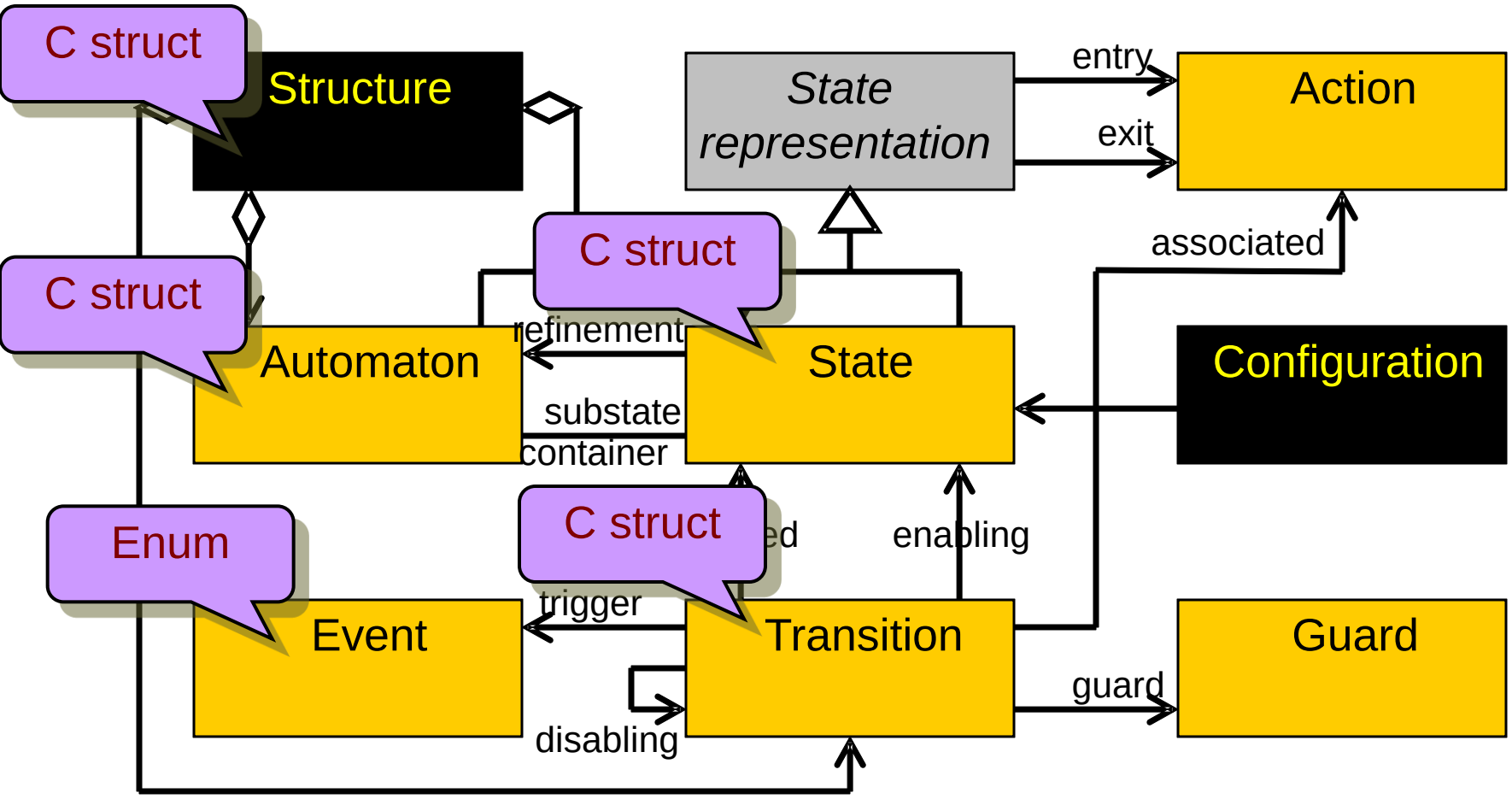
- OR jellegű finomítást (állapothierarchia) támogat
- Konkurens viselkedést nem támogat

„Kiterjesztett hierarchikus automata” minta



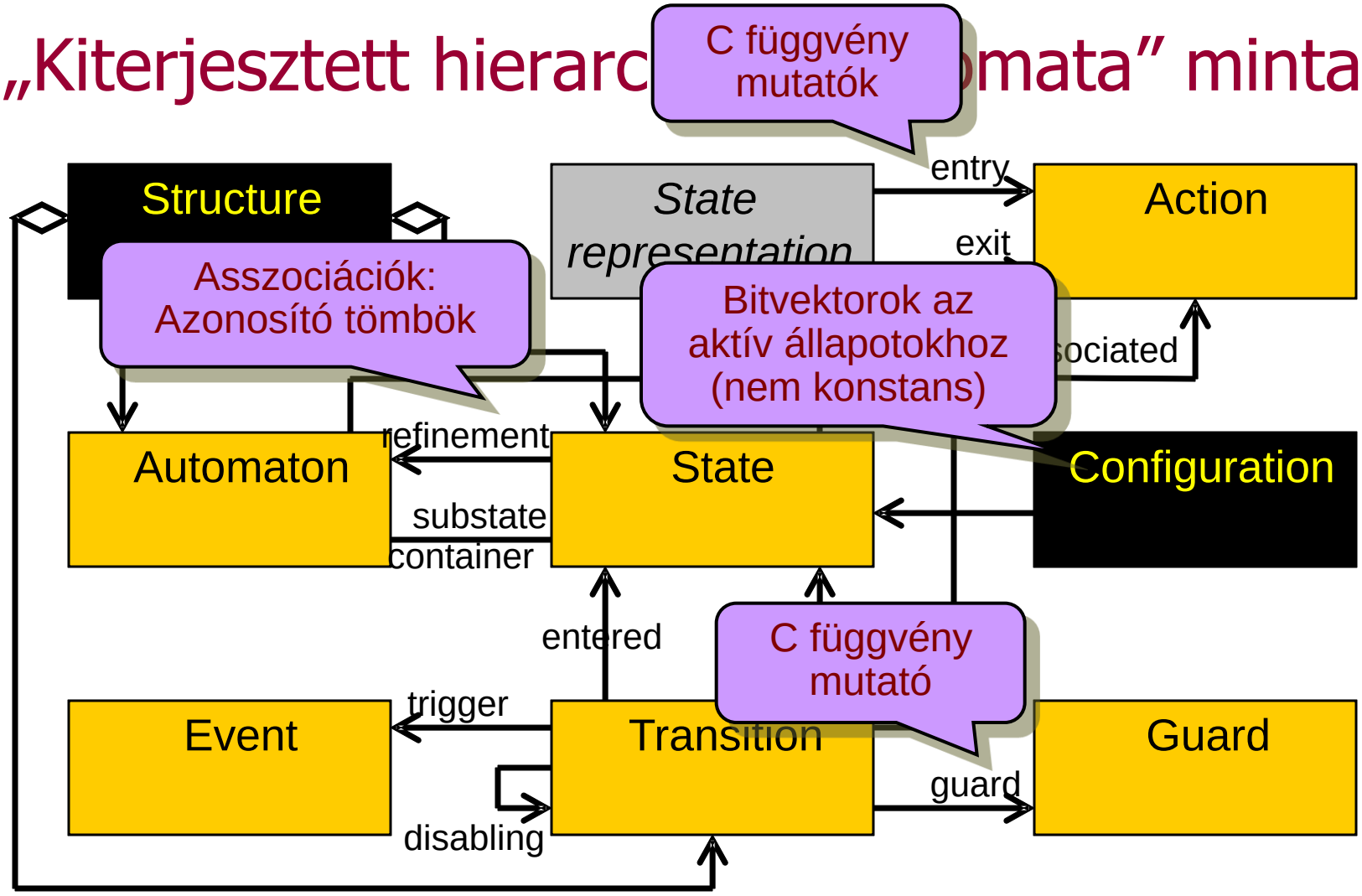
- A statikus struktúra a hierarchikus automata metamodellje alapján
  - <<ordered>> asszociációk (sorrend kötött a modell alapján)
  - Pl. belépési sorrend, akció sorrend, …
- Konfiguráció: aktív állapotok kijelölése

„Kiterjesztett hierarchikus automata” minta



- Megvalósítás ANSI C-ben:
  - C struktúrák

„Kiterjesztett hierarc omata” minta



- Megvalósítás ANSI C-ben:
  - Függvény mutatók
  - Azonosító tömbök

„Kiterjesztett hierarchikus automata” megvalósítása

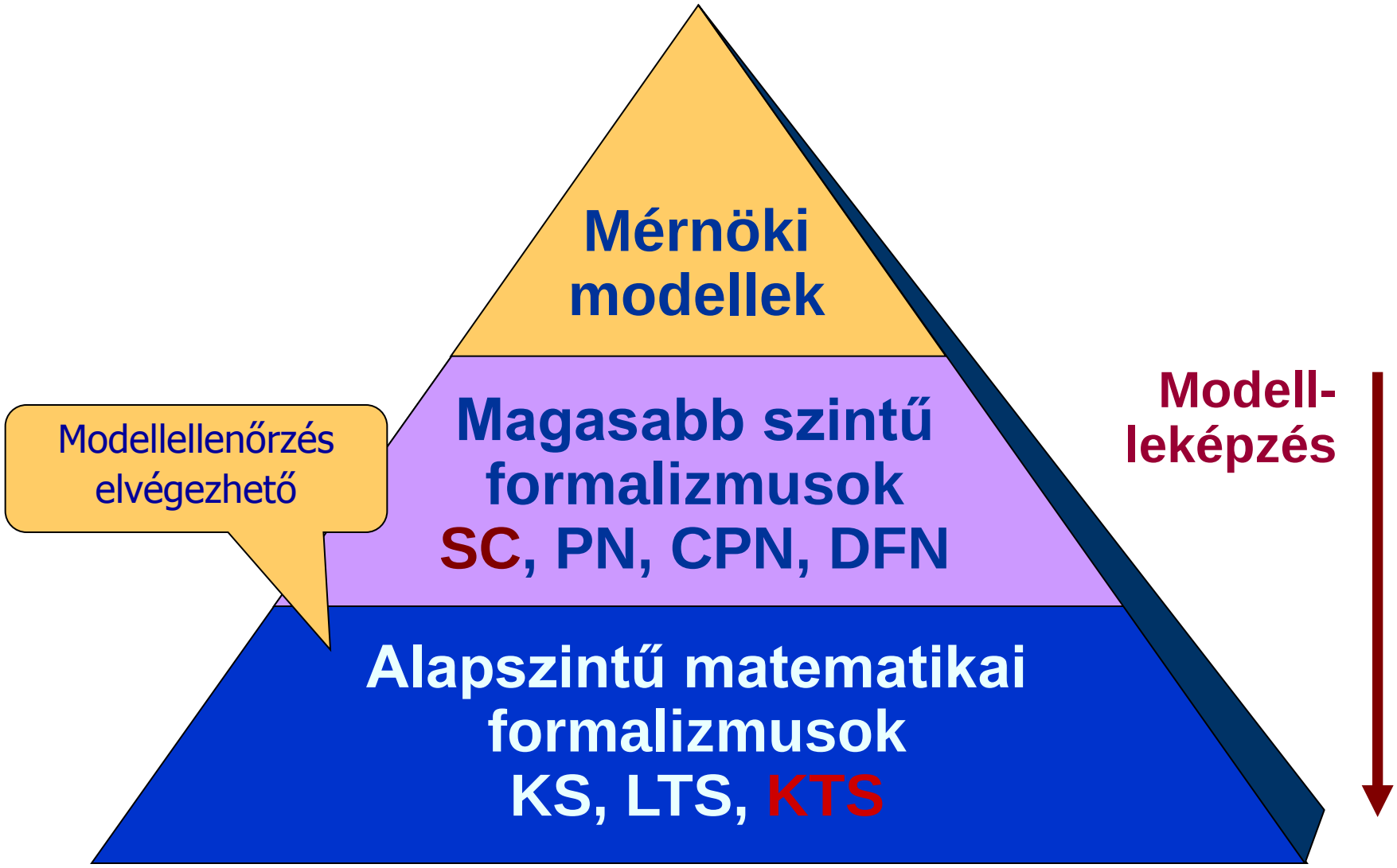
- Közös „interpreter” funkció
  - Formális szemantikán alapul
    - Állapot belépés: előre kiszámolva
    - Állapot kilépés: rekurzív függvénnyel (pl. konkurens régióból is)
  - Paraméterek:
    - Statikus struktúra: osztályonként egy
    - Konfiguráció: példányonként egy-egy
    - Feldolgozandó esemény
  - Visszatérési érték: Új állapotkonfiguráció
- Statikus struktúra (pl. TrafficLight)
  - Állapotok azonosítóval hivatkozhatók pointer helyett; méret csökkenthető

Összefoglaló

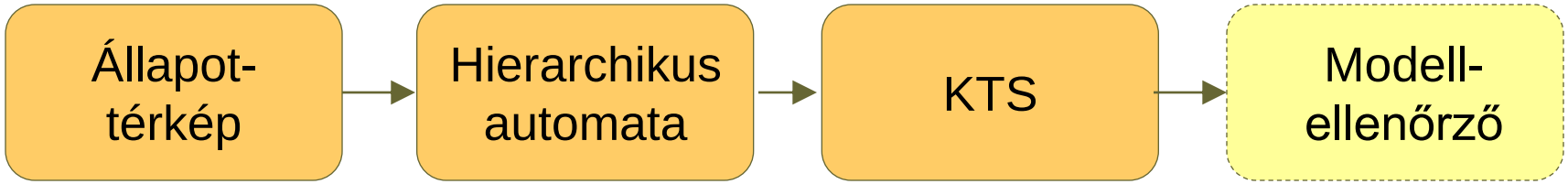
- Néhány kódgenerálási lehetőség állapottérkép modellek alapján
  - Kettős elágazás
  - „State” megvalósítási minta
  - „Hierarchikus állapotgép” (HSM) minta
  - „Kiterjesztett hierarchikus automata” (EHA) minta
- A modellelemek megvalósítása C-ben

Állapottérképek egy formális szemantikája: Leképézés KTS-re hierarchikus automatákon keresztül (kiegészítő anyag)

Modellek a formális ellenőrzéshez



Szemantika hozzárendelés leképezéssel

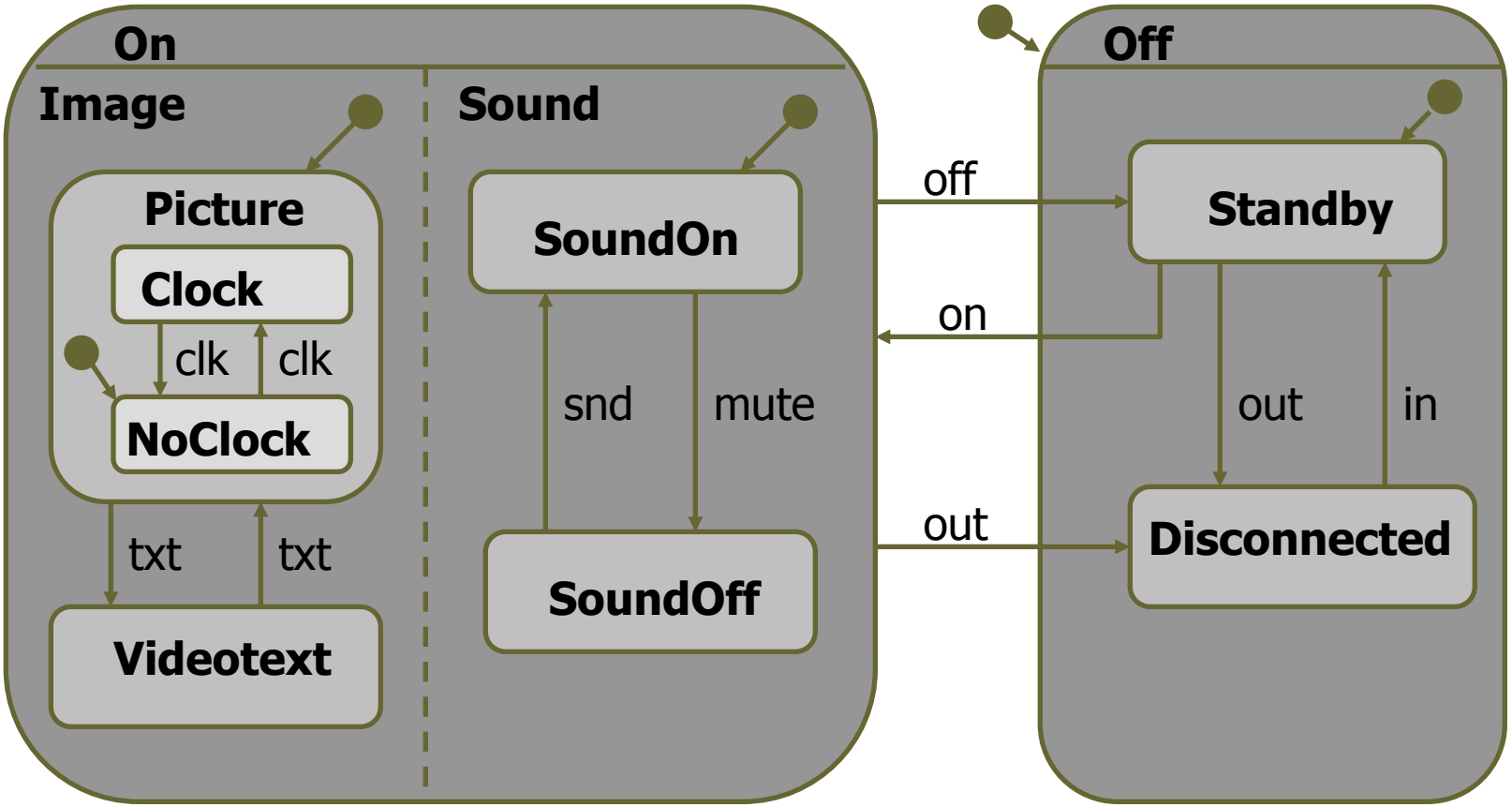


- Hierarchikus automata:
  - Absztrakt szintaxis az UML állapottérképekhez
  - Szintaktikai transzformáció
- Kripke tranzíciós rendszer
  - UML állapottérkép formális szemantikája
  - Formális verifikációra alkalmas formalizmus
- Modellellenőrzés
  - A formális szemantika egy felhasználása

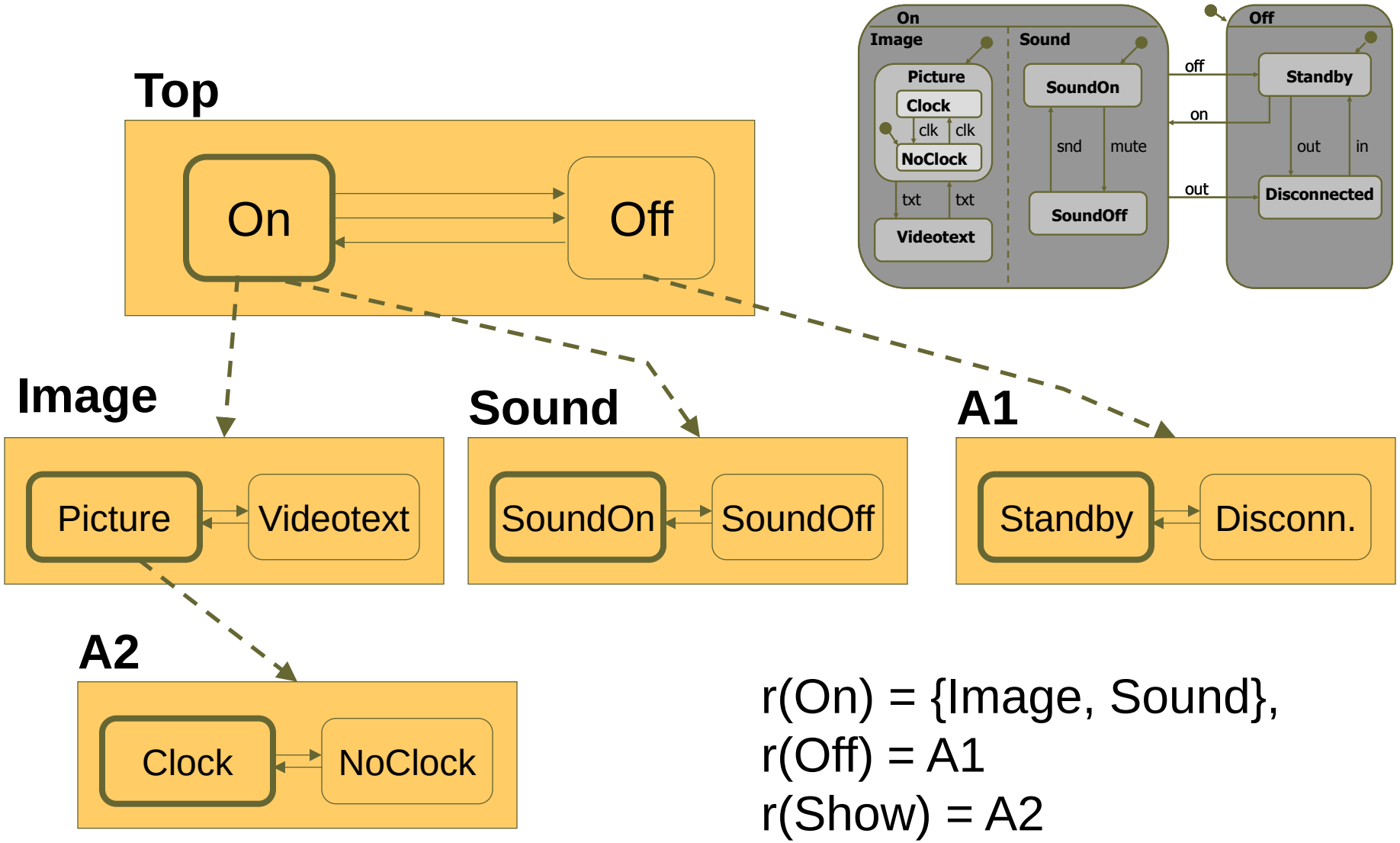
Hierarchikus automata

- Szekvenciális automata: A = (S, Act, T)
  - S állapotok halmaza, $s0 \in S$ kezdőállapot
  - Act az állapotátmenetek címkéi (trigger, akció, ...)
  - T az állapotátmenetek $T \subseteq S \times Act \times S$
- Hierarchikus automata: HA = (F, E, r, B)
  - F szekvenciális automaták halmaza
  - E események halmaza
  - r finomítási reláció
  - B címkék halmaza

Példa: TV hierarchikus állapottérkép



Példa: TV hierarchikus automata



r(On) = {Image, Sound},
r(Off) = A1
r(Show) = A2

A hierarchikus automata átmeneteinek címkéi

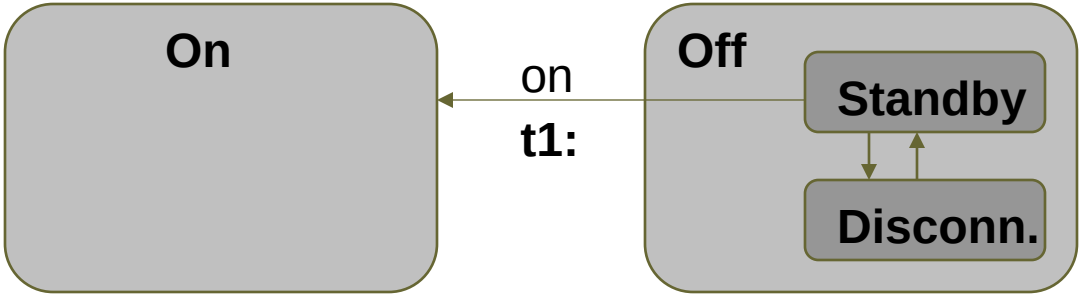
- Az állapotfinomítási hierarchián átívelő átmenetek: Legkisebb közös ős szintjére vannak hozva
  - Forrás állapotok (amiket elhagy) befoglalva
  - Cél állapotok (ahová belép) befoglalva

  Így látszik, mit kell elhagyni és hová kell belépni

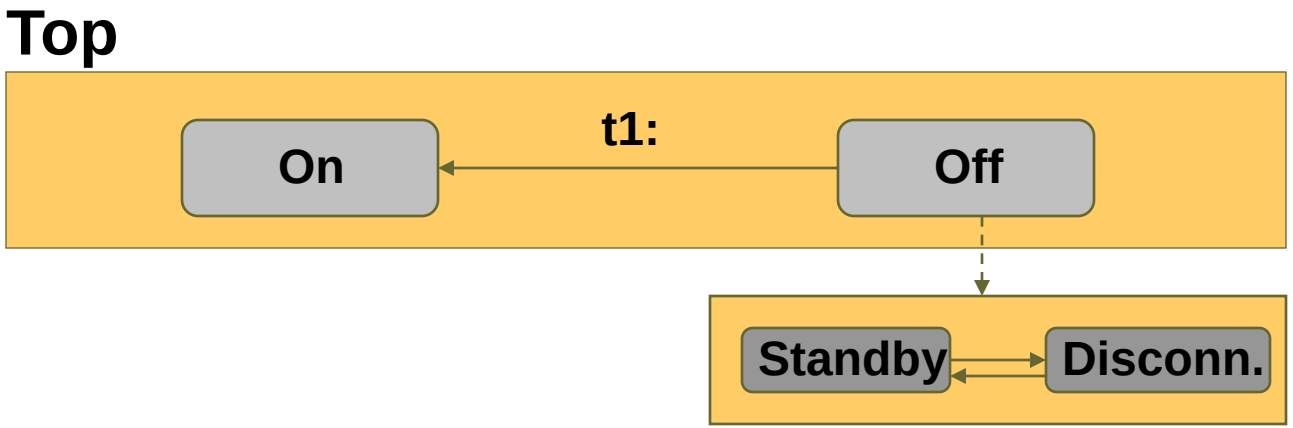
  Konfliktusok explicit megjelennek
- HA átmenet címkéjének elemei:
  - Eredeti forrás állapot (mélyebben lehet): SR(t)
  - Eredeti cél állapot (mélyebben lehet): TD(t)
  - Trigger esemény: TG(t)
  - Akciók: AC(t)
  - Őrfeltétel: GD(t)

Példa: HA átmenetek címkéi

Állapottérkép részlet:



HA részlet:



t1 címkéje: (eredeti forrás, cél, trigger, akció, őrfeltétel)

({Standby}, ∅, on, ∅, ∅)

SR(t) = {Standby}

Állapot-hierarchia és prioritás

- Konfliktusok meghatározása:
  - HA modellben az átmenet forrásának jelölése: SRC(t)
  - t1 és t2 átmenetek konfliktusban vannak, ha a finomítási hierarchiában (fában) azonos út mentén vannak, azaz

    SRC(t1) = SRC(t2) vagy

    SRC(t1) $\in$ r*(SRC(t2)) vagy SRC(t2) $\in$ r*(SRC(t1))
    itt r* reláció kiterjesztett az alatta lévő teljes hierarchiára
- Állapotátmenetek prioritása:
  - A mélyebben lévő SR(t) lehet a meghatározó
  - Ha nincs SR(t) akkor SRC(t),
    egyébként SR(t) határozza meg:

$$\{s \mid s \in SRC(t) \text{ ha } SR(t)=0 \} \cup SR(t)$$

Prioritás kezelése

- Egy állapotátmenet tüzelésének feltétele:
  - Nincs nagyobb prioritású engedélyezett átmenet a többi szekvenciális automatában
- HA hierarchia alsóbb szintjein:
  - Nagyobb prioritású átmenetek vannak
- HA hierarchia felsőbb szintjein:
  SR miatt lehet nagyobb prioritású átmenet!
  - Bevezetett jelölés: $A \uparrow P$,
    P a „felülről származó”, de nagyobb prioritású engedélyezett átmenetek halmaza,
    mint kényszer az A automata tüzelésekor

A környezet modellezése

Lehetőségek:

- Zárt rendszer:
  - A környezetet is modellezzük
  - Csak állapotgép(ek) állít(anak) elő új eseményeket
- Nyitott rendszer:
  - A környezetet nem modellezzük
  - Érkeznek „külső” események, amiknek a forrása nincs a modellben

Továbbiakban:

  - Zárt rendszerrel foglalkozunk
  - Adatkezeléstől eltekintünk (csak események)
  - Emlékező állapotok nincsenek

Szemantika: Kripke tranzíciós rendszer

KTS = (S, $\rightarrow$, L) és Act

- S állapotok halmaza: (C,Q)
  - C: állapotkonfigurációk („top” HA-tól indulva lefelé)
  - Q: eseménysor állapotok
  - s0 kezdőállapot: (C0, Q0)
    kezdő állapotkonfiguráció és eseménysor
- L állapot címkézés:
  - C állapotkonfigurációkhoz összegyűjtve
- $\rightarrow$ átmenet reláció (lépés): $(C,Q) \rightarrow (C',Q')$

(1) $Q'' = \text{remove}(e,Q)$

(2) $$\frac{\text{Top}\uparrow\varnothing :: (C, \{e\}) \rightarrow^{T} (C', Q')}{(C,Q) \rightarrow (C', \text{join}(Q',Q''))}$$

Szabálykészlet

- Kripke-struktúra $\rightarrow$ relációjának megadása:
  HA $\rightarrow^{T}$ relációja segítségével történt
  - T-ben lévő átmenetek tüzelnek a HA-ban
  - Mely átmenetek lesznek T-ben?

- HA $\rightarrow^{T}$ relációjának megadása:
  A szekvenciális automaták tüzelése 3 szabály szerint történhet:
  - Lépés („progress”)
  - Kompozíció („composition”)
  - Henyélés („stuttering”)

1. A lépés szabálya

- Egy $A\uparrow P$ automata t átmenete lokálisan tüzel
- Feltételek:
  - A t átmenet az automatában engedélyezett: SR(t), SRC(t) az aktuális konfiguráció része
  - Nincs sem az SRC(t) alatt lévő HA-kban, sem P-ben (a felső szintű automatákban) nagyobb prioritású engedélyezett átmenet
- Következmény: $A\uparrow P :: (C, \{e\}) \rightarrow^{T} (C', Q')$
  - T tüzelő átmenetek: T = {t}
  - C' új konfiguráció: t és TD(t) határozza meg
  - Q' új események: AC(t)

2. A kompozíció szabálya

- Egy $A \uparrow P$ automata nem tüzel, hanem „összefogja” az alacsonyabb szintű automaták tüzeléseit
- Feltételek:
  - Az alatta lévő A_i HA-k tüzelnek:
    $A_i \uparrow P' :: (C, \{e\}) \rightarrow^{T_i} (C_i', Q_i')$
  - Vagy ha $\cup T_i = \varnothing$, akkor lokálisan sem tüzel (P miatt)
- Következmények:
  - Tüzelő átmenetek: $T = \cup_j T_i$
  - Új konfiguráció: $\{s\} \cup_j C_i'$ (s az eredeti állapot)
  - Új eseménysor: $join\_all_i (Q_i')$

3. A henyélés szabálya

- Egy A↑P automata nem tüzel, állapota nem változik
- Feltételek:
  - Minden lokálisan engedélyezett átmenetnél van nagyobb prioritású P-ben, vagy
  - Nincs alsóbb szintű automata, amit összefoghat
- Következmények:
  - Tüzelő átmenetek: T = ∅
  - Új állapot: marad a régi...
  - Új eseménysor: nincs új generált esemény

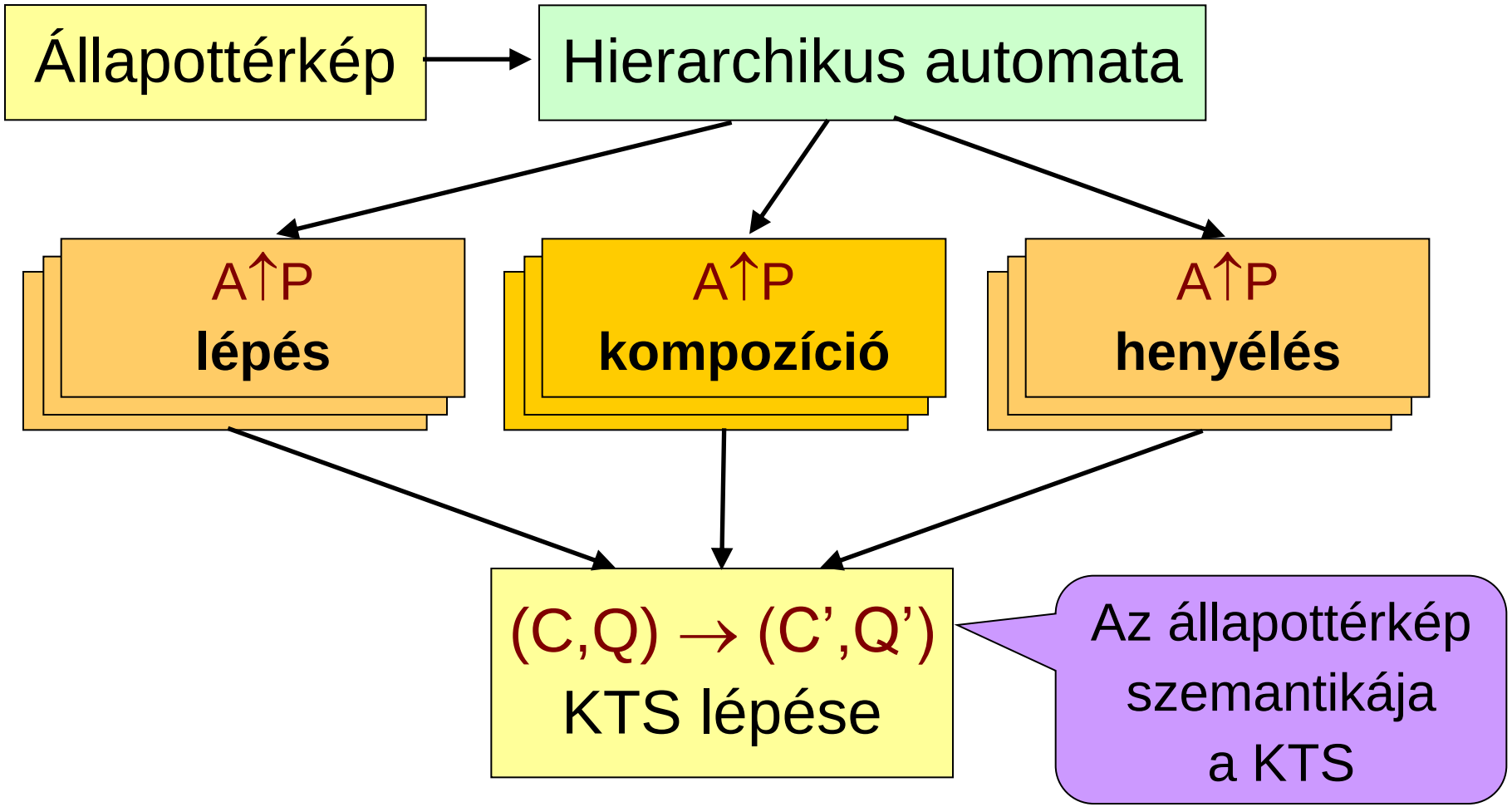
Szabályok használata

Rekurzív jellegű feltételek:

- Kompozíció szabálya
- Egy adott automata tüzelése függ
  - Hierarchiában alatta elhelyezkedő HA-któl: van-e engedélyezett átmenetük
  - Hierarchiában fölötte elhelyezkedő HA-któl; P mint "felülről származó" kényszer (ld. kompozíció)

Minden átmenet tüzel, amire van érvényes szabály

A szabályok áttekintése



KTS: ((C,Q), →, L)