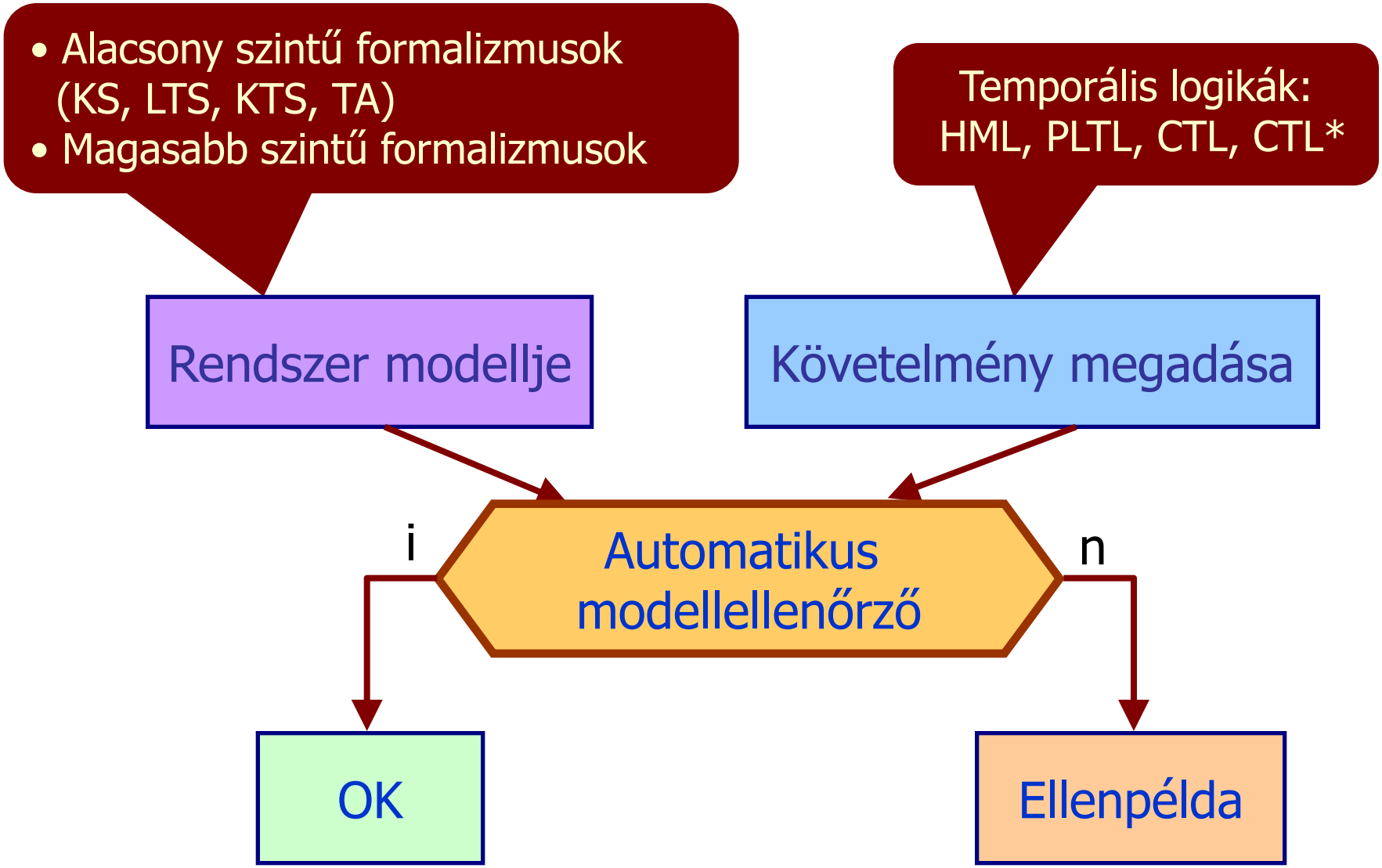
# A modellellenőrzés hatékony technikái: Részleges rendezés redukció az LTL modellellenőrzés során

Majzik István
BME Méréstechnika és Információs Rendszerek Tanszék

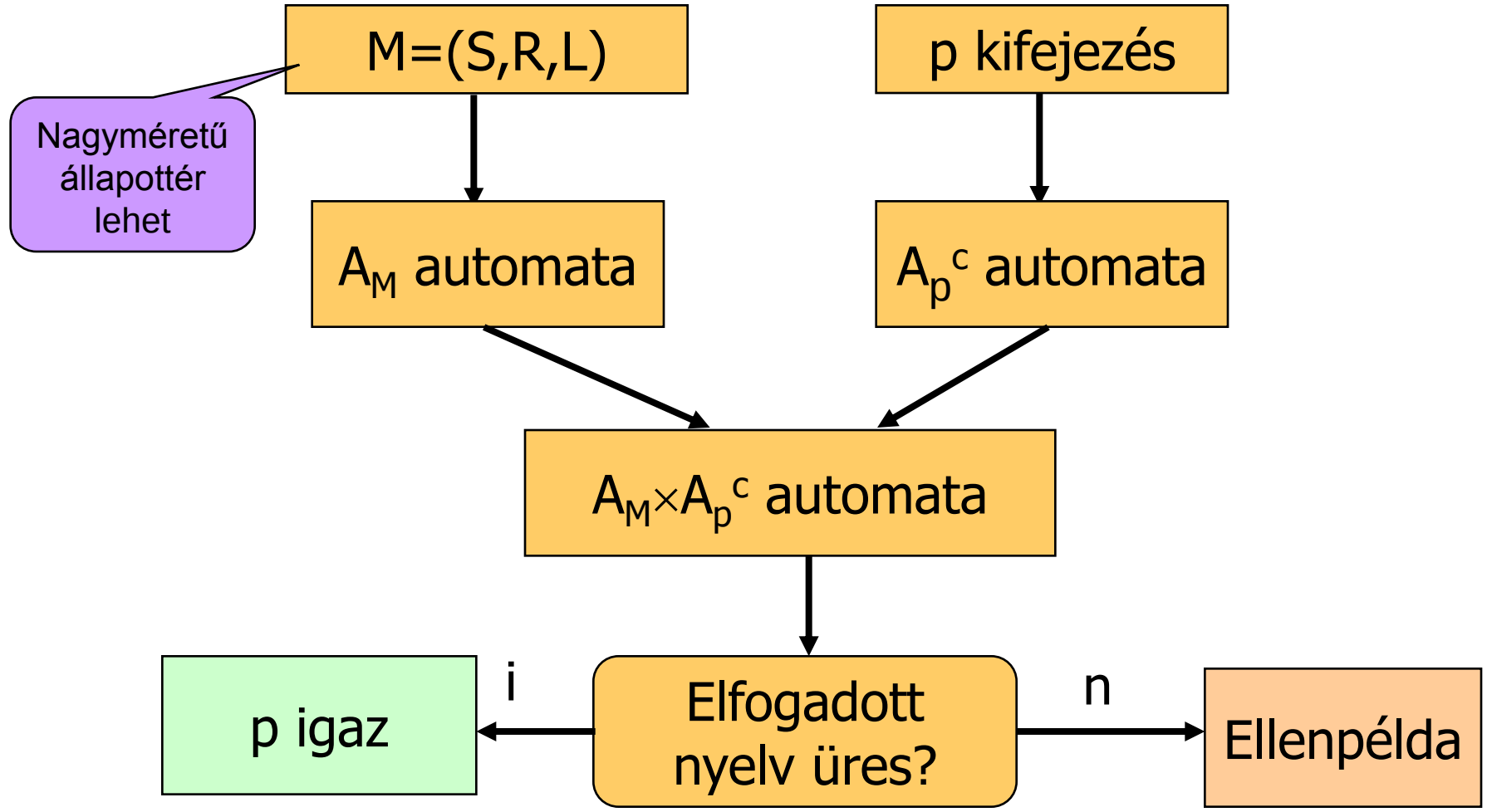
# Ismétlés: Mit szeretnénk elérni?



# Áttekintés: Az állapottér kezelése

- CTL modellellenőrzés: Szimbolikus technika
  - (Címkézett) állapothalmazok tárolása és manipulálása helyett Boole függvényeken végzett műveletek
  - A Boole függvények hatékony tárolása ROBDD alkalmazásával
  - Első ilyen modellellenőrzők: SMV, nuSMV
- Invariánsok modellellenőrzése: Korlátos modellellenőrzés
  - Logikai függvények igazságának keresése SAT technikával
  - Adott mélységig folytatható modell ellenőrzés: Korlátos hosszúságú ellenpéldák keresése
- LTL modellellenőrzés: Részleges rendezés
  - A lehetséges útvonalak közül reprezentatív útvonalak kiválasztása az adott követelmény ellenőrzéséhez
  - Bemutatott technika: SPIN modellellenőrző alapja
- Általános módszer problémaméret csökkentésre: Absztrakció

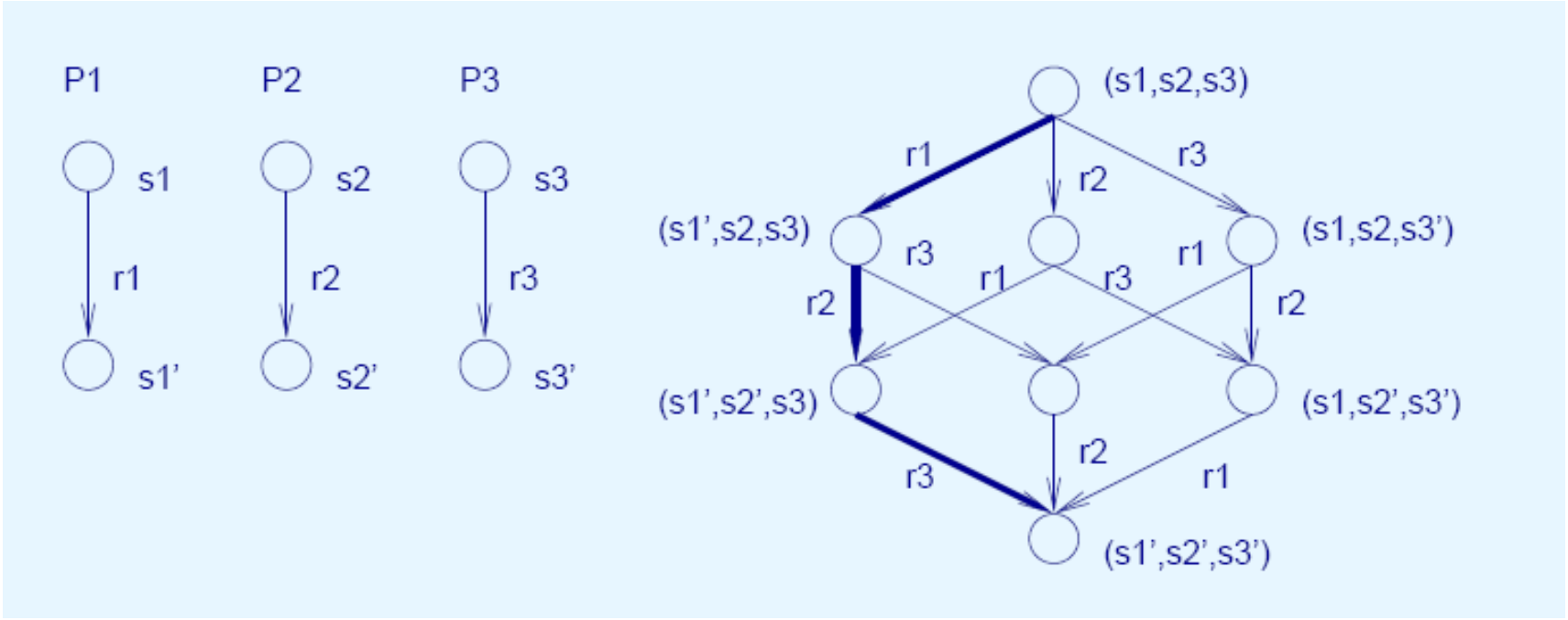
# Ismétlés: Automata alapú PLTL modellellenőrzés



# Részleges rendezés (partial order) redukció

- Elosztott rendszerek modellellenőrzése
  - Sok résztvevő, időnként interakciók
  - Akciók részben független, konkurens végrehajtása
  - A globális állapottér: minden lehetséges sorrend (átlapolt végrehajtás) felvétele –> állapottér robbanás
- Az állapottér robbanás kezelésére:
  - A független akcióknak nem kell minden lehetséges sorrendjét figyelembe venni
    - Ha az ellenőrizendő követelmény erre nem érzékeny
  - Elég egy reprezentatív sorrendet felvenni
  - Így az állapottér mérete jelentősen csökkenthető

# A részleges rendezés bemutatása




- Reprezentatív útvonal: r1 – r2 – r3 sorrend
- 4 állapot tárolása elkerülhető

# A modellellenőrzés

- Cél: LTL modellellenőrzés a reprezentatív útvonalak alapján M=(S, R, L) KS-en AP mellett
- A reprezentatív útvonalakat az állapottér felépítése közben (nem pedig a teljes állapottér felvétele után) szeretnénk kijelölni
- Alkalmazott jelölések:
  - enabled(s) az s állapotban engedélyezett átmenetek
  - ample(s)$\subseteq$enabled(s) az s állapotban a reprezentatív útvonal(ak) bejárásához választott átmenet(ek)

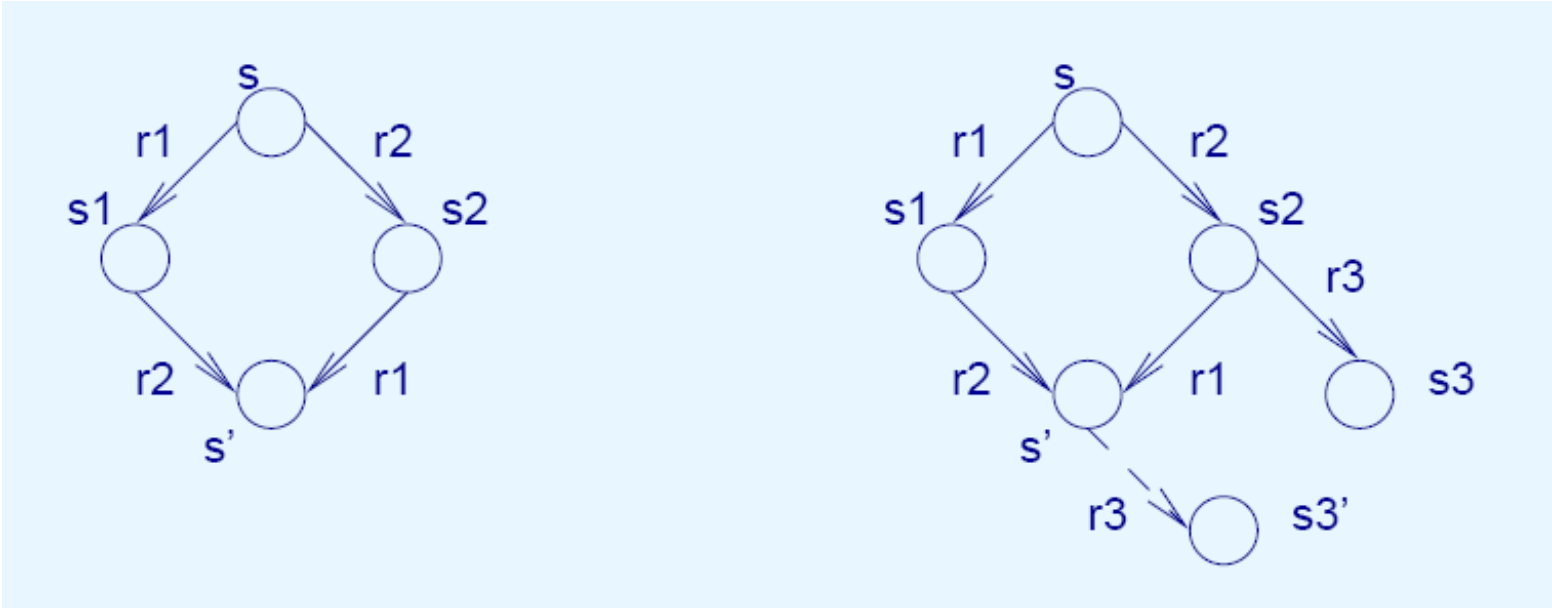
# Kritériumok ample(s) kiválasztásához

- Lokális választás kellene
  - A teljes állapottér vizsgálata nélkül
  - A mélységi keresés (rendszerszintű állapottér előállítás) során
- Szükséges feltételek:
  - A modellellenőrzés helyessége
  - Az állapottér méretének csökkentése
  - Elfogadható számítási igény

# Átmenetek tulajdonságai I.

- Független átmenetek: $I \subseteq R \times R$ reláció, ahol
  - A reláció szimmetrikus és antireflexív
  - Ha $(r1,r2) \in I$ akkor az átmenetek egymást nem tiltják le
    - Ha $r1,r2 \in enabled(s)$ akkor $r1 \in enabled(r2(s))$, valamint $r2 \in enabled(r1(s))$
  - Ha $(r1,r2) \in I$ akkor a különböző végrehajtási sorrend azonos állapothoz vezet
    - $r1(r2(s)) = r2(r1(s))$
- Első ötlet:
  - A független átmenetek közül elég az egyiket választani, és ezzel indítani a reprezentatív útvonalat
  - Problémák: Utak elhagyása, címkézés

# Független átmenetek - példa



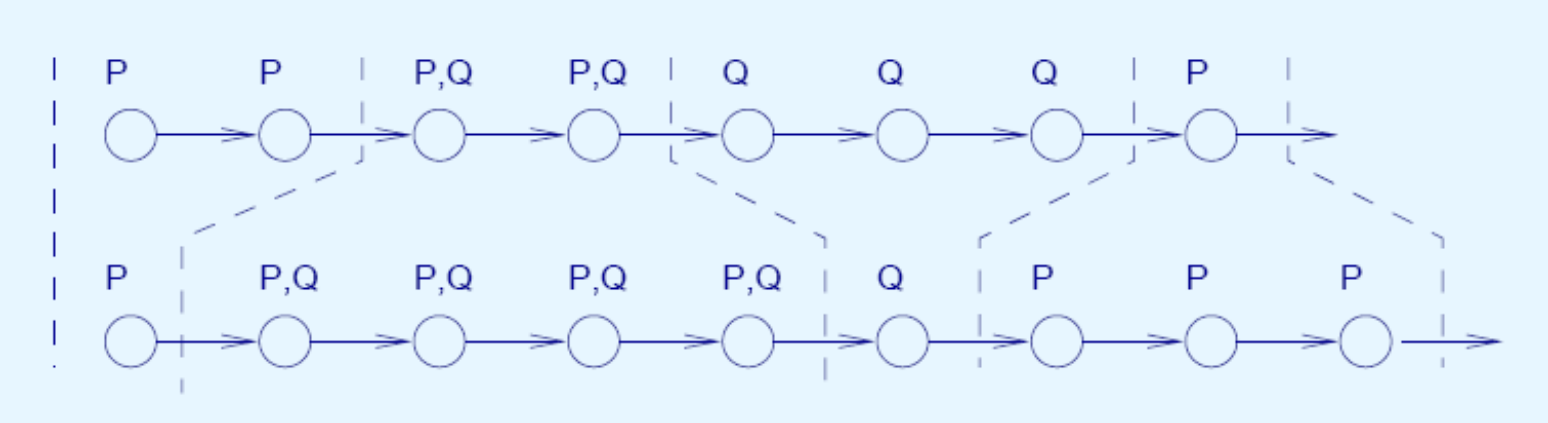

- Az ellenőrizendő követelmény teljesítése függhet az r1 vagy r2 választásától:
  - Ha r1(s) illetve r2(s) más címkéjű
  - Ha r2(s)-ből induló utak kimaradnak, ha r1-et választjuk

# Átmenetek tulajdonságai II.

- Láthatatlan átmenet:
  - r1=(s,s') láthatatlan, ha L(s)=L(s')
  - Az $A_p$ tulajdonság-automata nem „látja" az átmenetet
    (ld. szinkron szorzat képzése a modellellenőrzés során)
- Stuttering (dadogó) blokk:
  - Azonosan címkézett állapotok véges szekvenciája
  - Láthatatlan átmenetek az állapotok között
- Stuttering ekvivalens útvonalak:

  $\pi_1 \sim_{st} \pi_2$ ha azonos stuttering blokkok, azonos sorrendben fordulnak elő a két útvonalon
- Stuttering ekvivalens KS:
  - $M_1 \sim_{st} M_2$ ha a kezdőállapotok azonosan címkézettek, valamint minden $M_1$-beli $\pi_1$ útvonalhoz van olyan $\pi_2$ útvonal $M_2$-ben, hogy $\pi_1 \sim_{st} \pi_2$

# Stuttering ekvivalens útvonalak - példa




- Absztrakció: Elvonatkoztatunk az azonosan címkézett állapotok számától egy-egy blokkban
  - De erre „érzékeny” az X operátor!
  - Ha nem használjuk az X operátort, akkor használható ez az absztrakció

# A követelmények érzékenysége

- PLTL kifejezések tulajdonságai
  - Definíció: Egy p PLTL kifejezés stuttering invariáns (implicit A útvonal kvantorral):
    Ha minden $\pi_1 \sim_{st} \pi_2$ útvonalpárra: $\pi_1 \models p$ a.cs.a. $\pi_2 \models p$
  - Stuttering ekvivalens útvonalak között nem tesz különbséget a stuttering invariáns kifejezés
  - Tehát: Egy útvonalat a tulajdonság ellenőrzése szempontjából reprezentálhat a vele stuttering ekvivalens útvonal
- Bizonyítható állítások:
  - Minden $PLTL_{-X}$ kifejezés stuttering invariáns

    $PLTL_{-X}$ : az X operátor kimarad az operátorok közül!
  - Minden stuttering invariáns PLTL kifejezés felírható mint $PLTL_{-X}$ kifejezés

# A részleges rendezés célkitűzése

- A $PLTL_{-X}$ formulák azonosan értékelhetők ki a stuttering ekvivalens Kripke struktúrákon
  - Cél: Úgy kell redukálni a Kripke-struktúrát, hogy stuttering ekvivalens legyen az eredeti és a redukált struktúra
  - Ezen a redukált struktúrán $PLTL_{-X}$ követelmények ellenőrizhetők

Ezek után:

- Mik a feltételei olyan ample(s) választásnak, hogy (a reprezentatív útvonalak választásával) stuttering ekvivalens redukált struktúrát kapjunk?
  - 4 kritériumot dolgoztak ki
  - Bizonyítható: betartásukkal optimális redukció elérhető

# Reprezentatív átmenetek választásának kritériumai

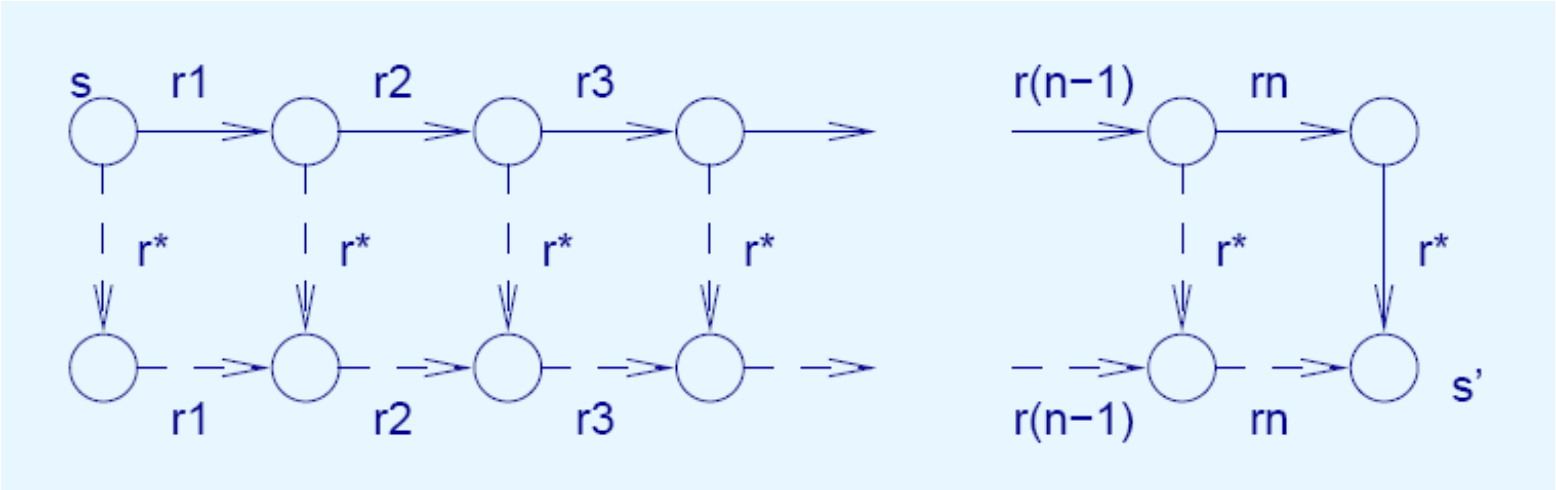
- C(0) ample(s)=0 a.cs.a. ha enabled(s)=0
  - Intuitív: nem akarunk deadlock szituációt, ha van továbblépés
- C(1) A teljes (nem redukált) Kripke-struktúrában minden s-ből induló útvonalra igaz kell legyen:

  Egy ample(s)-belitől *függő* átmenet nem hajtható végre anélkül, hogy előtte egy ample(s)-beli átmenetet végre ne hajtanánk.
- A feltételt a következményein keresztül próbáljuk meg értelmezni:
  - Első észrevétel: C(1) alkalmazásával az enabled(s) \ ample(s) halmazbeli (kimaradó) átmenetek függetlenek kell legyenek az ample(s)-beli átmenetektől

# C(1) következményei

- Ha a reprezentatív átmeneteket C(0) és C(1) szerint választjuk, milyen utak maradnak ki ample(s) választással?
  - $r_1,r_2,...,r_n,r^*$, ... minta szerinti utak,
    ahol $r_1,r_2,...,r_n$ függetlenek ample(s)-től és $r^* \in$ ample(s)
  - $r_1,r_2,...,r_i,...$ minta szerinti végtelen utak,
    ahol $r_1,r_2,...,r_n$ függetlenek ample(s)-től
- Az első eset elemzése:
  - $r_1,r_2,...,r_n,r^*$ és $r^*,r_1,r_2,...,r_n$ utak ugyanazt az állapotot érik el
    - Mivel r* „előrehozható” a függetlenség definícióját alkalmazva
    - Az utóbbi út $r^* \in$ ample(s) átmenettel kezdődik, ezért lesz esély lefedni
  - $r_1,r_2,...,r_n,r^*$ tehát reprezentálható,
    ha $r_1,r_2,...,r_n,r^* \sim_{st} r^*,r_1,r_2,...,r_n$
  - Ez a stuttering ekvivalencia akkor lesz igaz,
    ha r* láthatatlan átmenet
  - Ezt a C(2) kritérium biztosítja majd

# Az első eset elemzése - példa



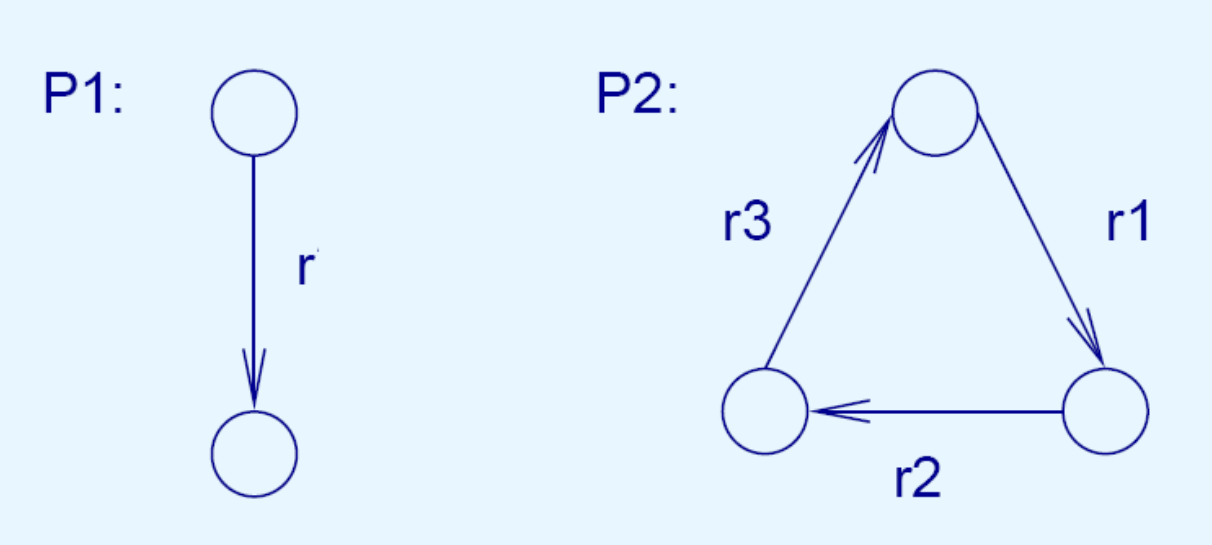

Itt r* lépésenként „előrehozható”

- Mivel $r_1,r_2,...,r_n$ függetlenek r*-tól, a függetlenség definícióját alkalmazhatjuk

# C(2) kritérium

- C(2) Ha ample(s) $\neq$ enabled(s), akkor
  minden r$\in$ample(s) láthatatlan kell legyen.
  - Ez alapján az előbbi $r_1,r_2,...,r_n,r^* \sim_{st} r^*,r_1,r_2,...,r_n$ fennáll
    - Az első minta szerinti kimaradó utak reprezentálhatók
  - Azt is biztosítja, hogy $r_1,r_2,...,r_i,... \sim_{st} r^*,r_1,r_2,...,r_i,...$
    - A második minta szerinti kimaradó utak reprezentálhatók
- Még marad lehetőség, hogy kihagyjunk útvonalat:
  - A lokális választási kritériumok miatt ha ciklus van láthatatlan átmenetekből egy résztvevő modelljében, akkor egy másik résztvevő átmenete kimaradhat
  - Példa: ld. következő dia
  - Ennek elkerülése lesz: C(3) kritérium

# Kimaradó eset - példa




- r1,r2,r3 *ciklus* az egyik résztvevő KS-ében
- Itt r független az r1,r2,r3 átmenetektől és nem láthatatlan, ugyanakkor r1,r2,r3 láthatatlanok és egymástól függnek
- Rendre r1,r2,r3 átmeneteket választva az ample() halmazba, a ciklus záródik és a másik résztvevő r átmenete kimarad
  - „Késleltetett” marad az átmenet választás lokális döntései miatt

# C(3) kritérium

- C(3) Ciklus nem megengedett, ha olyan állapotot tartalmazna, amelyben egy r nem láthatatlan átmenet engedélyezett, de nincs benne a ciklus egy állapotának ample() halmazában sem.
  - Így nem lehetnek „kifelejtett” átmenetek
  - Gond: Egy adott s állapotban lokálisan nem eldönthető a kritérium teljesítése!
    - Csak globálisan (a teljes állapottérben) látható, hogy van-e ilyen ciklus
    - Közelítő megoldás javasolható

# A gyakorlati alkalmazás szempontjai

- A kritériumok betartásának számításigénye:
  - C(0) lokálisan egyszerűen ellenőrizhető
  - C(2) lokálisan egyszerűen ellenőrizhető
  - C(1) az egész állapottér vizsgálatát igényelné!
    - Kompromisszum: Olyan ample(s) keresése, ami betartja C(1)-et, de nem a lehető legkisebb ample(s) (nem optimális a redukció)
      - Példa: Konkurens processzek átmeneteinek vizsgálata; ha $P_i$-ben lévő $T_i$ átmenetektől csak független engedélyezett átmenetek vannak más processzekben, akkor ample(s)=$T_i$, ha ilyen nem található, akkor minden processzt ki kell bontani
      - A függőséget a nyelvi primitívek (pl. kommunikáció) kijelölik
  - C(3) esetén cikluskeresés kellene a teljes állapottérben
    - Helyette C(3)' kritérium: Ha ample(s)≠enabled(s) akkor ample(s)-beli átmenet nem léphet a DFS során már érintett állapotba (azaz nem záródhat ciklus)
    - Így legalább egy állapot teljesen ki lesz bontva a ciklusban

# A redukció hatékonysága

- Legjobb eset: Exponenciális állapottér-robbanás lineáris növekedéssé redukálható
  - Pl. protokoll sok azonos viselkedésű résztvevővel
- Tipikus: A redukció lineáris a modell méretével
- Memória és futási idő nyereség: 10..90%
- Példa: Vezetőválasztási protokoll gyűrű struktúrában

| Processzek száma | Redukció nélkül | | Redukcióval | |
|---|---|---|---|---|
| | Állapotok | Időigény | Állapotok | Időigény |
| 3 | 15 929 | 13,8 s | 1 435 | 0,6 s |
| 4 | 522 255 | 9,3 perc | 8 475 | 3,5 s |
| 5 | | >40 óra | 57 555 | 28,7 s |
| 6 | | | 434 083 | 4,1 perc |

- Előny: A részleges rendezés nem igényel paraméterezést
  - Pl. ROBDD esetén kritikus az állapotváltozók sorrendezése