

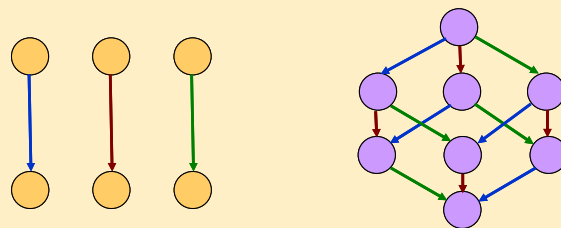
# A modellellenőrzés hatékony technikái (az állapottér kezelése)

Majzik István

BME Méréstechnika és Információs Rendszerek Tanszék

## Ismétlés: A modellellenőrzés tanult technikái

- PLTL modell ellenőrzés:
  - Automata alapú megközelítés:
    - Szinkron szorzat automata konstruálása, elfogadó állapotok keresése
- CTL modell ellenőrzés:
  - Szemantika alapú módszer:
    - Állapotok iteratív címkézése (fixpont iteráció)
- Állapottér robbanás:
  - Egyszerűek az alapszintű modellek (állapot és átmenet)
  - Nagy méretű állapottér adódik (pl. elosztott rendszerek)



- Hogyan lehet nagy méretű modelleket ellenőrizni?
  - Ígéret: CTL modell ellenőrzés:  $10^{20}$  állapot (egyes esetekben  $>10^{100}$  állapot)

# Előzetes áttekintés

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

## Szimbolikus modellellenőrzés a CTL esetén

## Ismétlés: CTL modellellenőrzés állapot címkézéssel

- Az iteráció halmazműveletekkel történik
  - Kezdőhalmaz: Előző lépések alapján
  - Inkrementális bővítés a  $Z$  halmazhoz tartozó (már címkézett) állapotok megelőző állapotain:  
$$\text{pre}_E(Z) = \{s \in S \mid \text{létezik olyan } s', \text{ hogy } (s, s') \in R \text{ és } s' \in Z\}$$
$$\text{pre}_A(Z) = \{s \in S \mid \text{minden } s'\text{-re, ahol } (s, s') \in R: s' \in Z\}$$
- Példa:  $E(P \cup Q)$  iterációja:
  - Kezdőhalmaz:  $X_0 = \{s \mid Q \in L(s)\}$
  - Iteráció:  $X_{i+1} = X_i \cup (\text{pre}_E(X_i) \cap \{s \mid P \in L(s)\})$ 
    - Eddig címkézettek és ..
    - .. megelőző állapotaik közül amelyek ...
    - ... P-vel címkézettek
  - Iteráció vége: Ha  $X_{i+1} = X_i$ , azaz nem bővül a halmaz

## Ismétlés: CTL modellellenőrzés fixpont kereséssel

- $\text{Sat}(EF p)$  számítása:  $\text{lfp } \tau(z)$ 
  - ahol  $\tau(z) = \text{Sat}(p) \cup \text{pre}_E(z)$
  - ahol  $\text{pre}_E(z) = \{s \mid \exists t: (s, s') \in R \text{ és } s' \in z\}$
- Kleene tétele alapján  $\text{lfp } \tau(z)$  számítása:
  - $z_0 = \emptyset$
  - $z_1 = \tau(z_0) = \tau(\emptyset) = \text{Sat}(p) \cup \text{pre}_E(\emptyset)$
  - ...
  - $z_{i+1} = \tau(z_i) = \text{Sat}(p) \cup \text{pre}_E(z_i)$
  - ...
  - folytatás amíg  $z_{i+1} = z_i$  és itt  $z_i = \text{lfp } \tau(z) = \text{Sat}(EF p)$

# A szimbolikus modellellenőrzés alapötlete

- **Állapothalmazok tárolása állapot felsorolás helyett logikai függvények formájában**
  - Állapotok „kódolása” Boole logikai vektorokkal
    - $S$  állapottér kódolásához  $n = \lceil \log_2 |S| \rceil$  bit elég, azaz ha teljesül  $2^n \geq |S|$
  - Állapothalmaz „kódolása”  $n$ -változós Boole függvényekkel
    - Akkor és csak akkor igaz egy-egy bitvektor behelyettesítésére, ha a bitvektor által kódolt állapot az adott állapothalmazban van
- **Karakterisztikus függvény:  $C: \{0,1\}^n \rightarrow \{0,1\}$** 
  - A karakterisztikus függvényekkel fogunk dolgozni a halmazok helyett
- **Elméleti alap: Stone-tétel**
  - Minden véges Boole-algebra izomorf egy véges  $S$  halmaz részalmazainak algebrájával
  - $(2^S, \cap, \cup, \emptyset, S)$  izomorf  $(F_n, \vee, \wedge, 0, 1)$ 
    - $F_n$  az  $n$ -változós logikai függvények  $\{0,1\}$  felett
    - $2^{2^n}$  féle  $n$ -változós logikai függvény van,  $2^{|S|}$  a részalmazok száma

## Karakterisztikus függvények

- **$s$  állapotra:  $C_s(x_1, x_2, \dots, x_n)$** 

Legyen az  $s$  „kódolása”  $(u_1, u_2, \dots, u_n)$  vektor, itt  $u_i \in \{0,1\}$   
Ekkor  $C_s(x_1, x_2, \dots, x_n)$  csak az  $(u_1, u_2, \dots, u_n)$  esetén adjon 1 értéket  
 $C_s(x_1, x_2, \dots, x_n)$  konstruálása:  $\wedge$  operátorral

  - $x_i$  szerepel, ha  $u_i=1$
  - $\neg x_i$  szerepel, ha  $u_i=0$

Példa:  $(0,1)$  kódolású  $s$  állapotra:  $C_s(x_1, x_2) = \neg x_1 \wedge x_2$
- **$Y \subseteq S$  állapothalmazra:  $C_Y(x_1, x_2, \dots, x_n)$** 

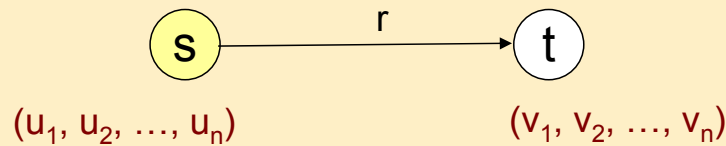
$C_Y(x_1, x_2, \dots, x_n)$  a.cs.a. legyen igaz egy  $(u_1, u_2, \dots, u_n)$  behelyettesítésre, ha  $(u_1, u_2, \dots, u_n) \in Y$

$C_Y(x_1, x_2, \dots, x_n)$  konstruálása:  $C_Y(x_1, x_2, \dots, x_n) = \bigvee_{s \in Y} C_s(x_1, x_2, \dots, x_n)$
- **Állapothalmazokra általában:**

$C_{Y \cup W} = C_Y \vee C_W, \quad C_{Y \cap W} = C_Y \wedge C_W$

## Karakterisztikus függvények (folytatás)

- Állapotátmenetekre:  $C_r$



$r=(s,t)$  állapotátmenet, ahol  $s=(u_1, u_2, \dots, u_n)$  és  $t=(v_1, v_2, \dots, v_n)$

Karakterisztikus függvény  $C_r(x_1, x_2, \dots, x_n, x'_1, x'_2, \dots, x'_n)$  alakban; „vesszős” változók jelzik a cél állapotot

$C_r$  a.cs.a. legyen igaz, ha  $x_i=u_i$  és  $x'_i=v_i$  a behelyettesítés

$C_r$  konstruálása:

$$C_r = C_s(x_1, x_2, \dots, x_n) \wedge C_t(x'_1, x'_2, \dots, x'_n)$$

## Karakterisztikus függvények (folytatás)

- $\text{pre}_E(z)$  képzése:  $\text{pre}_E(z)=\{s \mid \exists t: (s,t) \in R \text{ és } t \in z\}$   
z reprezentációja:  $C_z$

R reprezentációja:  $C_R = \bigvee_{r \in R} C_r$

$\text{pre}_E(z)$ : kikeresni a z-beli állapotokra az előzőeket

$$C_{\text{pre}_E(z)} = \exists_{x'_1, x'_2, \dots, x'_n} C_R \wedge C'_z$$

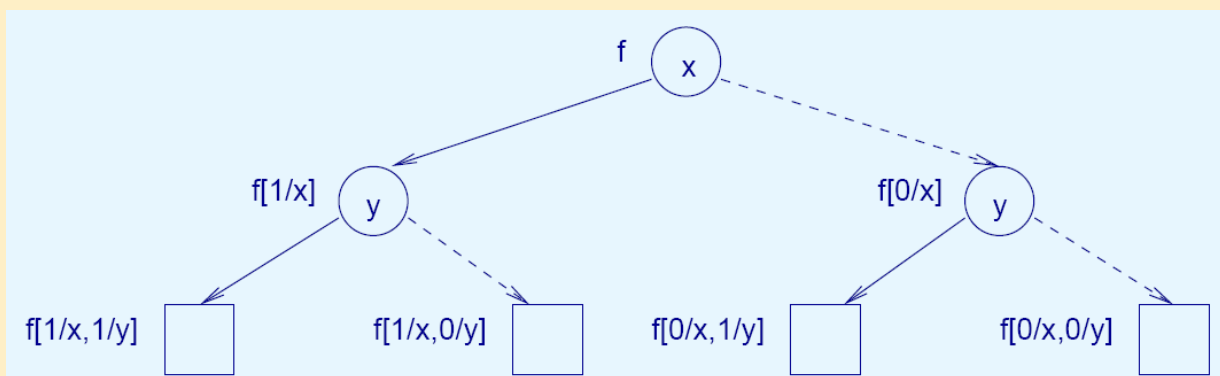
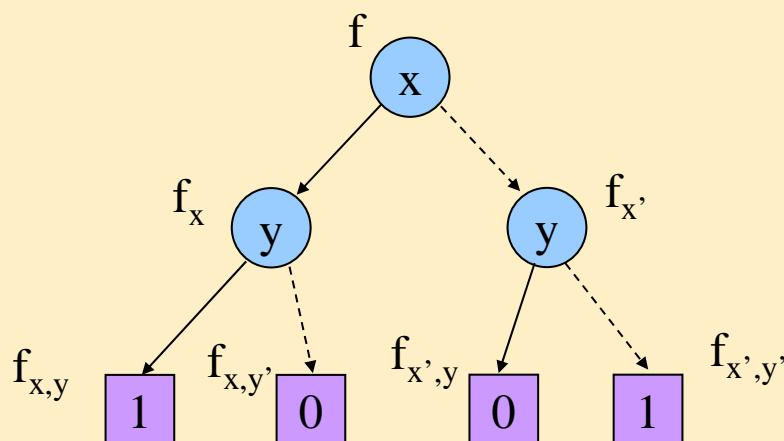
ahol  $\exists_x C = C[1/x] \vee C[0/x]$  „egzisztenciális absztrakció”

- Modellellenőrzés állapotalmaz műveletekkel:  
visszavezetve logikai függvényeken végzett műveletekre!

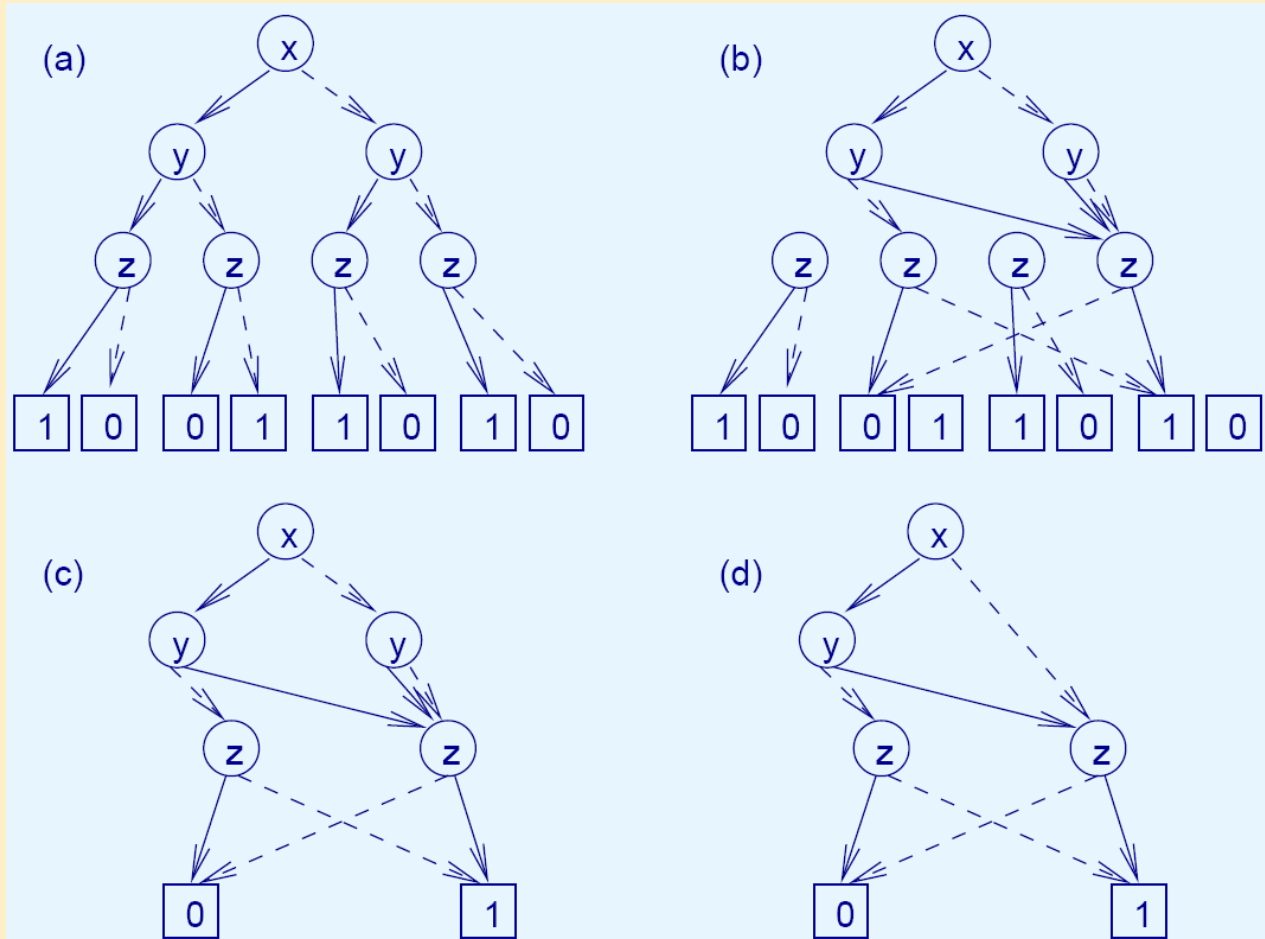
# Logikai függvények reprezentációja

- Kanonikus forma: ROBDD
  - redukált, rendezett, bináris döntési diagram
- Lépések: Bináris döntési fa, BDD, OBDD, ROBDD
  - Bináris döntési fa: Shannon-felbontás
    - „if-then-else” operátor:  $x \rightarrow x_1, x_0 = (x \wedge x_1) \vee (\neg x \wedge x_0)$
    - felbontás:  $f = x \rightarrow f[1/x], f[0/x]$
    - más jelölés:  $f(x, y, z, \dots) = x \rightarrow f_x(y, z, \dots), f_{x'}(y, z, \dots)$
  - BDD: azonos részfák összevonva
  - OBDD: minden ágon azonos változó sorrendezés
  - ROBDD: szükségtelen csomópontok redukálása
    - Ha mindkét ág ugyanahhoz a csomóponthoz vezet

## Bináris döntési fa

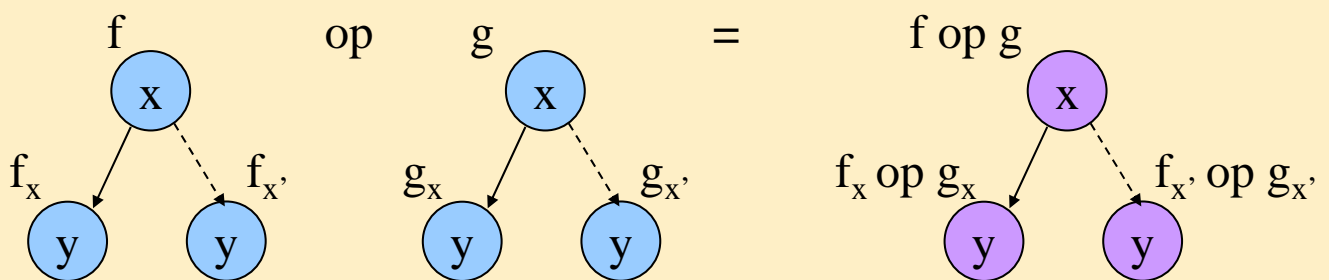


## Bináris döntési fa redukálása



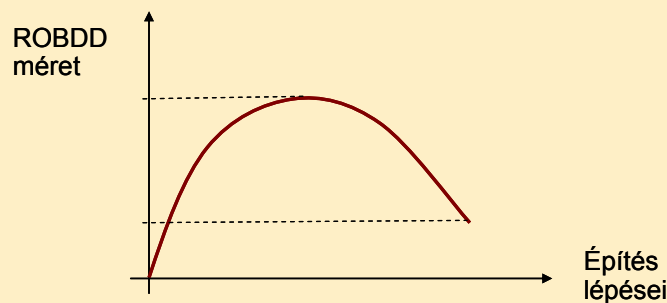
## ROBDD használat

- Generálás: Id. jegyzet (+ Formális módszerek tárgya)
  - Rekurzív felbontás és azonos részfák figyelése
- Műveletek ROBDD által reprezentált fv-ekkel:
  - $f \text{ op } g = (x \rightarrow f_x, f_{x'}) \text{ op } (x \rightarrow g_x, g_{x'}) = x \rightarrow (f_x \text{ op } g_x), (f_{x'} \text{ op } g_{x'})$
  - $f \text{ op } g = (x \rightarrow f_x, f_{x'}) \text{ op } g = x \rightarrow (f_x \text{ op } g), (f_{x'} \text{ op } g)$
  - $f \text{ op } g = f \text{ op } (x \rightarrow g_x, g_{x'}) = x \rightarrow (f \text{ op } g_x), (f \text{ op } g_{x'})$



## ROBDD mérete

- Tömör ábrázolásmód, pl. étkező filozófusok problémája:
  - 16 filozófus:  $4,7 \cdot 10^{10}$  állapot, 747 ROBDD csomópont
  - 28 filozófus:  $4,8 \cdot 10^{18}$  állapot, 1347 ROBDD csomópont itt 63 bites állapottér címzés helyett ~21kB elég!
- Fontos a helyes sorrend megválasztása
  - Más sorrend nagyságrendbeli méretkülönbséget is jelenthet
  - Optimális sorrend „próba-szerencse” alapon derülhet ki
- Tárigény:
  - Ha egy konkurens rendszer modellje alapján folyamatosan építjük a ROBDD-t, akkor az építés közben sok ideiglenes csomópontot kell tárolnunk, amit később lehet redukálni



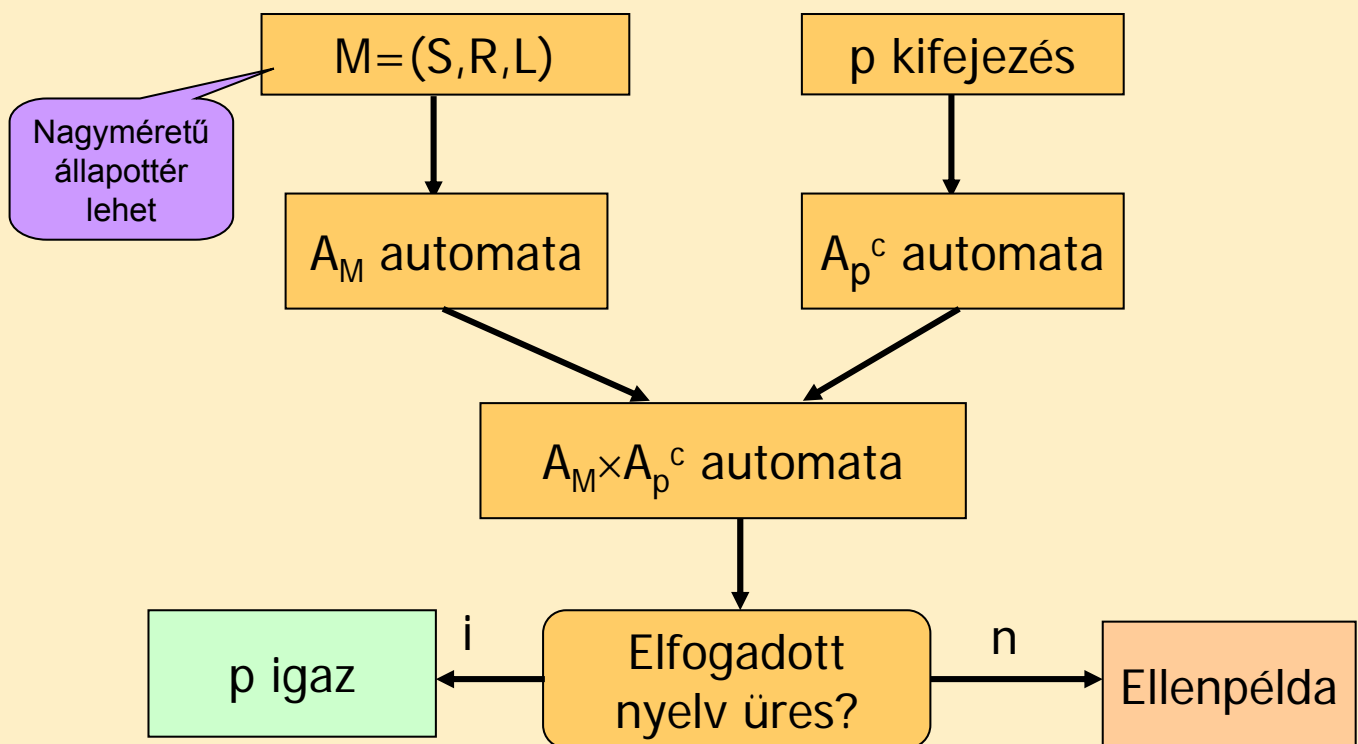
## ROBDD és modellellenőrzés (összefoglalás)

- Modellellenőrzés: Halmazműveletek
  - Iteratív  $Sat(p)$  számítás
- Halmazműveletek leképezése karakterisztikus függvények műveleteire
  - „Állapotok kódolása”: Boole-függvények
- Karakterisztikus függvény műveleteinek leképezése ROBDD-ken végzett műveletekre
  - Közvetlenül az ROBDD reprezentáción



# Részleges rendezés redukció a PLTL modellellenőrzés során

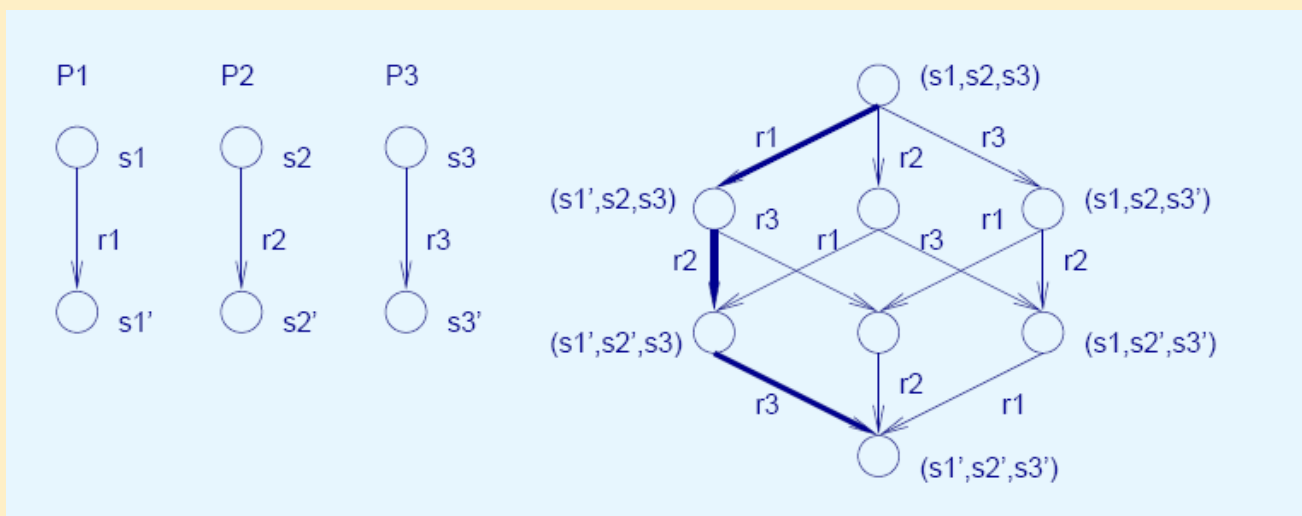
## Ismétlés: Automata alapú PLTL modellellenőrzés



## 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 azonos állapotba vezet minden sorrendű végrehajtás
    - 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: PLTL modellellenőrzés a reprezentatív útvonalak alapján  $M=(S, R, L)$  KS-en AP mellett
- 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)

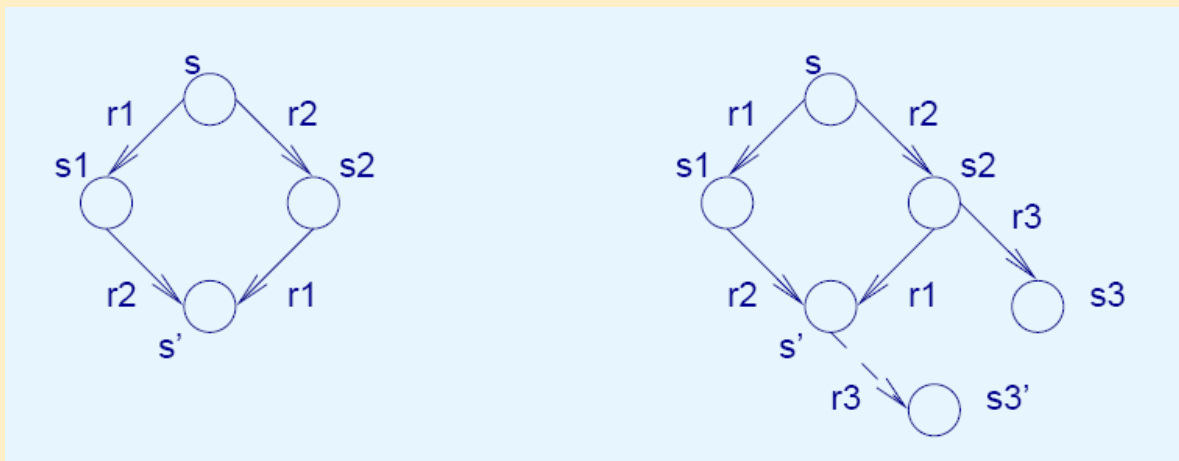
## Kritériumok $ample(s)$ kiválasztásához

- Lokális választás kellene
  - a teljes állapottér vizsgálata nélkül
  - a mélységi keresés (automata előállítás) során (DFS)
- Szükséges feltételek:
  - a modellellenőrzés helyessége
  - az állapottér méretének csökkentése
  - kis számítási igény

## Átmenetek tulajdonságai I.

- **Független átmenetek:  $I \subseteq R \times R$  reláció**
  - szimmetrikus és antireflexív reláció
  - ha  $(r1, r2) \in I$  akkor az átmenetek egymást nem tiltják le
    - ha  $r1, r2 \in \text{enabled}(s)$  akkor  $r1 \in \text{enabled}(r2(s))$ , valamint  $r2 \in \text{enabled}(r1(s))$
  - 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 csak egyet kell kiválasztani**
  - Problémák: Utak elhagyása, címkézés

## Független átmenetek - példa

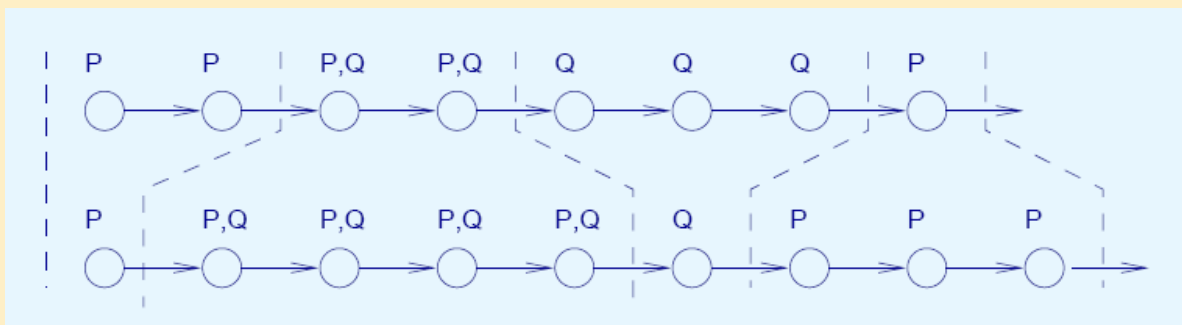


- Az ellenőrizendő követelmény teljesítése függhet az  $r1$  vagy  $r2$  választásától:
  - ha  $r1(s)$  illetve  $r2(s)$  más címkéjű
  - ha  $r2(s)$ -ből induló utak kimaradnak, ha  $r1$ -et választjuk

## Átmenetek tulajdonságai II.

- Láthatatlan átmenet:
  - $r1=(s,s')$  láthatatlan, ha  $L(s)=L(s')$
  - az  $A_p$  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, használható az absztrakció

## A követelmények érzékenysége

- Stuttering invariáns  $p$  PLTL kifejezés (implicit  $A$  útvonal kvantorral):  
Ha minden  $\pi_1 \sim_{st} \pi_2$  útvonalpárra:  $\pi_1 \models p$  a.cs.a.  $\pi_2 \models p$ ,  
azaz stuttering ekvivalens útvonalak között nem tesz különbséget a kifejezés
- 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 nem különböztetik meg a stuttering ekvivalens Kripke struktúrákat
  - 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

- $C(0)$   $\text{ample}(s)=0$  a.cs.a.  $\text{enabled}(s)=0$ 
  - Intuitív: mindenképpen van egy továbblépés
- $C(1)$  A teljes Kripke-struktúrán minden  $s$ -ből induló útvonalra igaz kell legyen:  
Egy  $\text{ample}(s)$ -belitől *függő* átmenet nem hajtható végre anélkül, hogy előtte egy  $\text{ample}(s)$ -beli átmenetet végre ne hajtánánk.

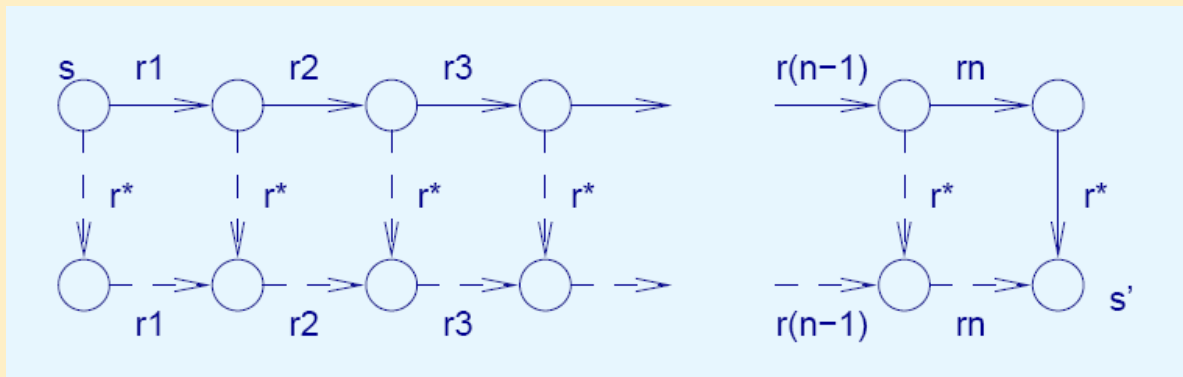
A feltételt a következményein keresztül próbáljuk meg megérteni:

- Következmény: Az  $\text{enabled}(s) \setminus \text{ample}(s)$  halmazbeli (kimaradó) átmenetek függetlenek az  $\text{ample}(s)$ -beli átmenetektől

## $C(1)$ következményei

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

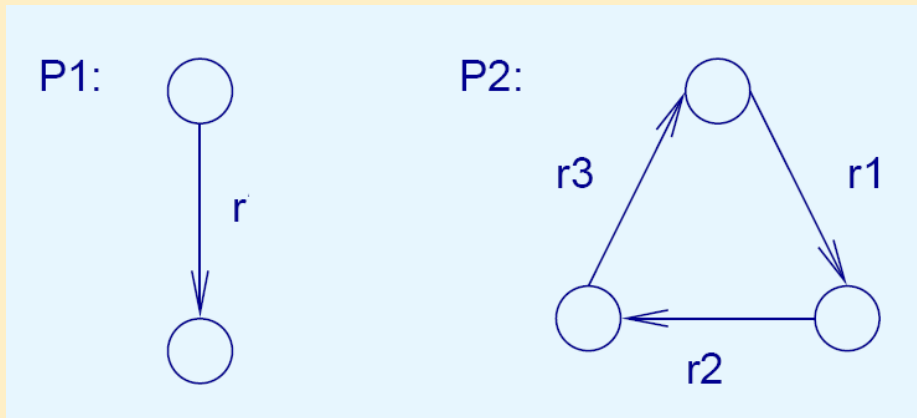
- mivel  $r_1, r_2, \dots, r_n$  függetlenek  $r^*$ -tól, a függetlenség definícióját alkalmazhatjuk

## C(2) kritérium

- C(2) Ha  $\text{ample}(s) \neq \text{enabled}(s)$ , akkor minden  $r \in \text{ample}(s)$  láthatatlan kell legyen.
  - Ez alapján az előbbi  $r_1, r_2, \dots, r_n, r^* \sim_{st} r^*, r_1, r_2, \dots, r_n$  fennáll
    - az első minta szerinti kimaradó utak reprezentálhatók
  - Azt is biztosítja, hogy  $r_1, r_2, \dots, r_i, \dots \sim_{st} r^*, r_1, r_2, \dots, r_i, \dots$ 
    - a második minta szerinti kimaradó utak reprezentálhatók
- Még marad lehetőség, hogy kihagyjunk útvonalat:
  - Ha *ciklus* van láthatatlan átmenetektől egy résztvevő KS-ében, egy másik résztvevő átmenete kimaradhat
  - Példa: ld. következő fólia
  - 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 függnék
- Rendre  $r1, r2, r3$  átmeneteket választva az  $\text{ample}()$  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  $r$  átmenet engedélyezett, de nincs benne a ciklus egy állapotának  $\text{ample}()$  halmazában sem.
  - Így nem lehetnek „kifelejtett” átmenetek
  - Gond: Egy adott  $s$  állapotban lokálisan nem eldönthető a kritérium teljesítése!
    - Globálisan látható, hogy van-e ilyen ciklus
    - Közelítő megoldás javasolt

## 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 redukció)
      - Példa: Konkurens processzek átmeneteinek vizsgálata; ha nincs a  $P_i$ -ben lévő  $T_i$  átmenektől függő engedélyezett átmenet más processzekben, akkor **ample(s)= $T_i$** , ha ilyen nem található, akkor minden processzt ki kell bontani
      - A függőséget a nyelvi primitívek (kommunikáció) kijelölik
  - C(3) esetén bonyolult cikluskeresés a teljes KS-ben
    - 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	
	Allapotok	Időigény	Allapotok	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

- Egy előny: A részleges rendezés nem igényel paraméterezést (pl. állapotváltozók sorrendezését)