

Hatékony technikák modellellenőrzéshez: Szimbolikus technikák (ROBDD)

dr. Majzik István
dr. Pataricza András
dr. Bartha Tamás

BME Méréstechnika és Információs Rendszerek Tanszék

Hol tartunk?

- Alacsony szintű formalizmusok (KS, LTS, KTS)
- Magasabb szintű formalizmusok

Temporális logikák:
PLTL, CTL, CTL*

Rendszer modellje

Követelmény megadása

Alapszintű algoritmus

Hogyan lesz hatékony?

i

Automatikus modellellenőrző

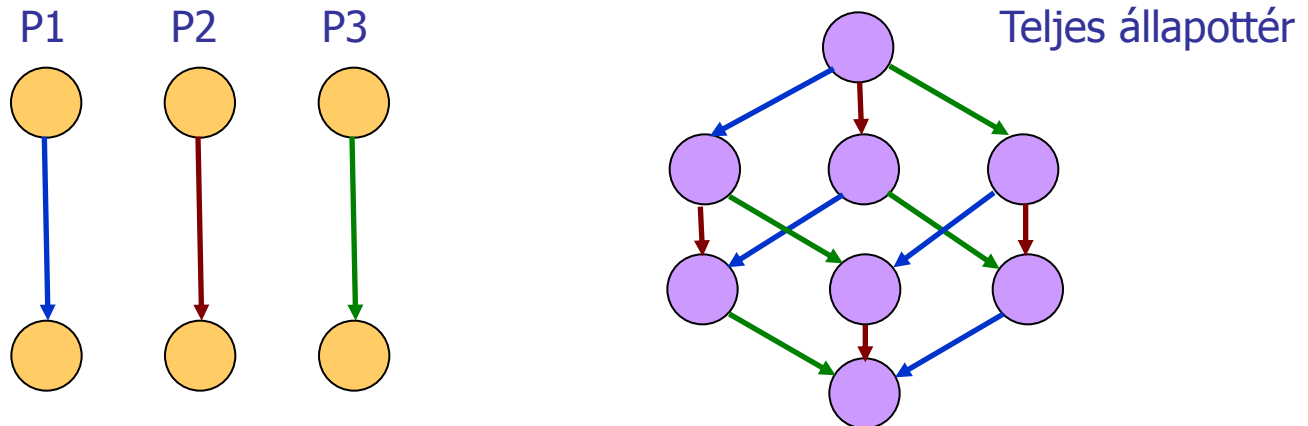
n

OK

Ellenpélda

Problémák

- Nagy méretű állapottér bejárása szükséges
 - Konkurens alkalmazások esetén nagy méretű állapottér adódik: Független állapotátmenetek végrehajtása sokféle (átlapolt) sorrendben történhet



- Hogyan lehet nagy méretű modelleket ellenőrizni?
 - Ígéret: CTL modellellenőrzés: 10^{20} , egyes esetekben 10^{100} állapot
 - Milyen technika tudja ezt biztosítani?

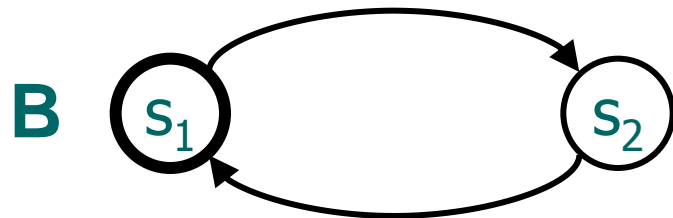
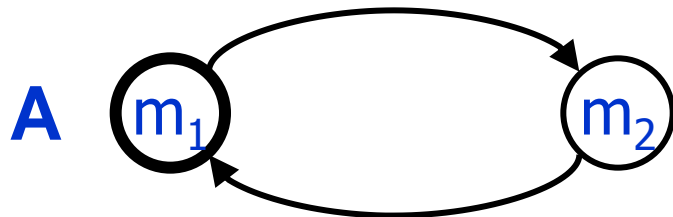
Kitérő: Miért nagy az állapottér?

Konkurens automaták együttes működése

**Automaták direkt szorzata, átlapolás,
szinkronizáció**

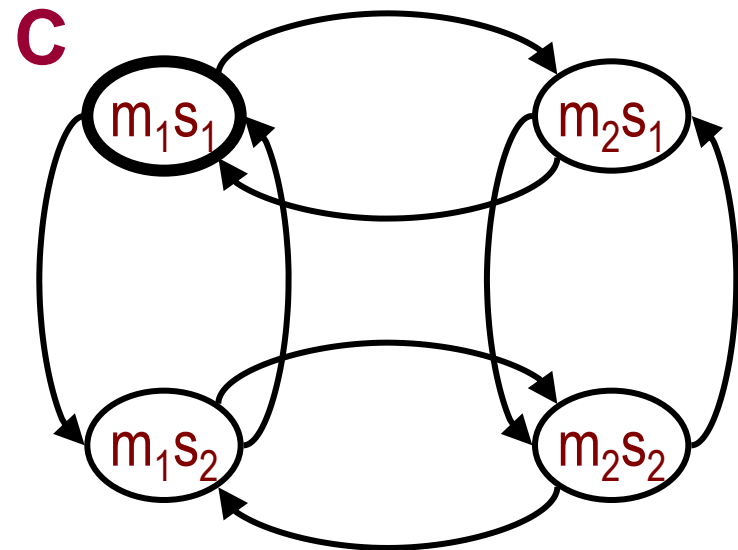
Példa: Aszinkron automaták működése

- Két (független) automatából álló rendszer



- Az automaták állapotai:
 $A = \{m_1, m_2\}$
 $B = \{s_1, s_2\}$

- (Direkt) szorzatautomata: a rendszer állapottere



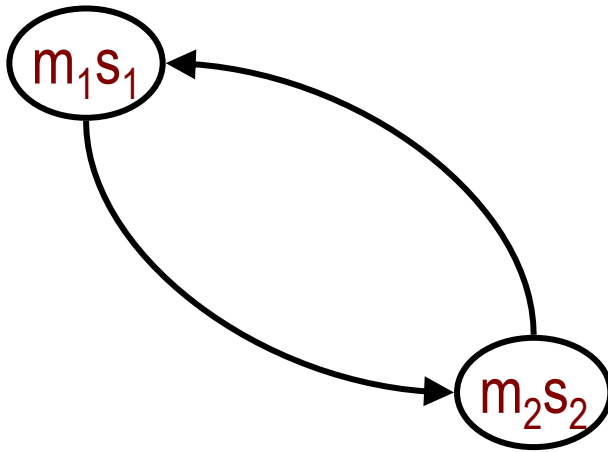
- Az állapotok halmaza:
 $C = A \times B$
 $C = \{m_1s_1, m_1s_2, m_2s_1, m_2s_2\}$

Végrehajtás korlátozása: Szinkronizáció, feltétel

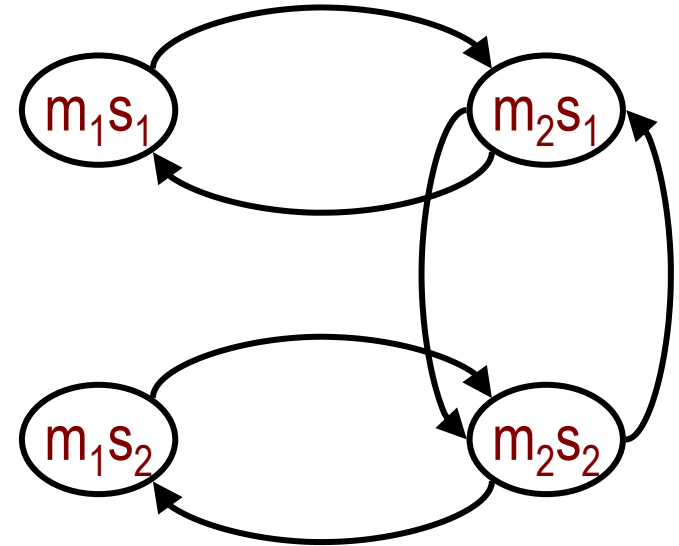
- Szinkronizáció:
Élpárok együttes lépése
- Példa: A és B egyszerre vált állapotot az azonos indexű állapotokból

- Korlátozó feltétel:
állapotátmenetek tiltása
- Példa: B csak akkor vált állapotot, ha A az m_2 állapotban van

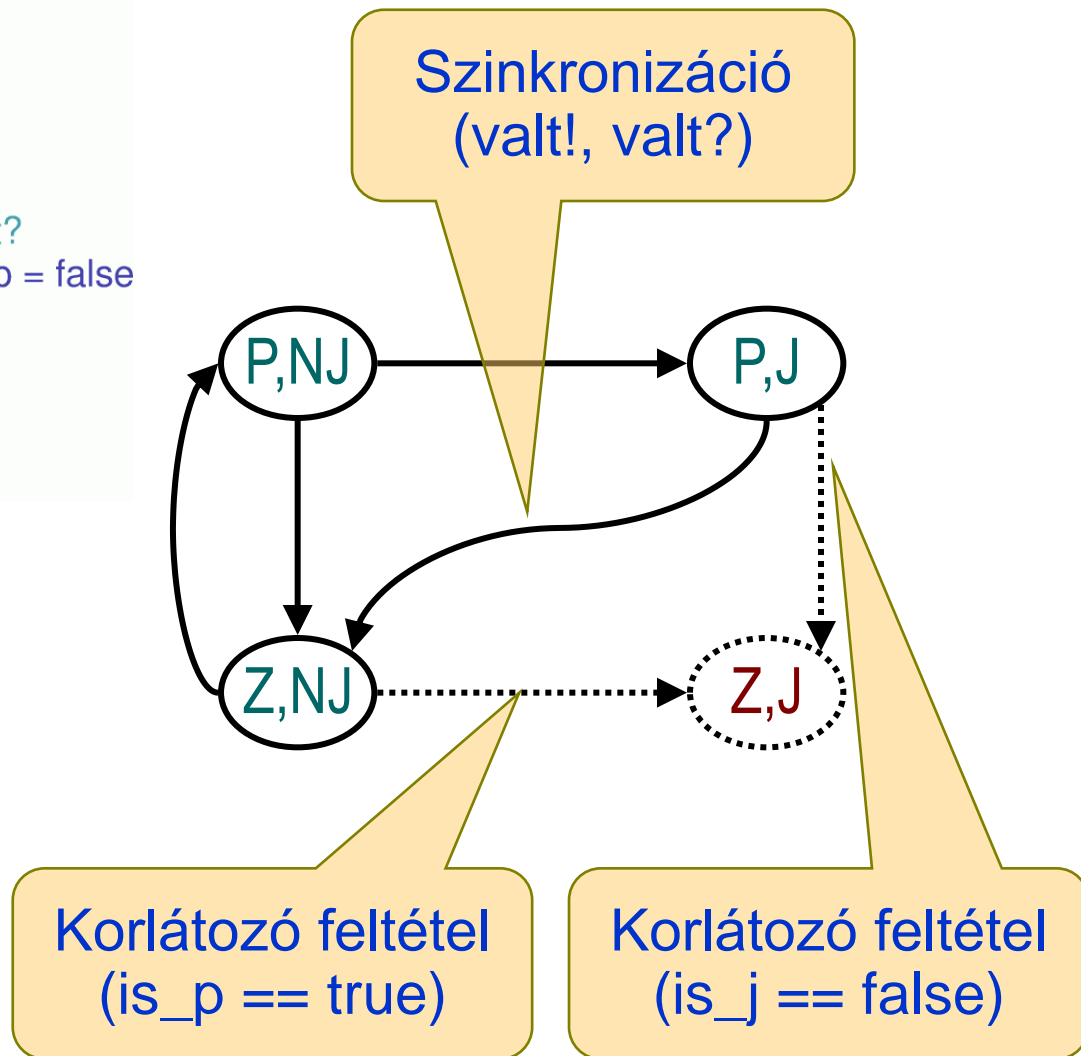
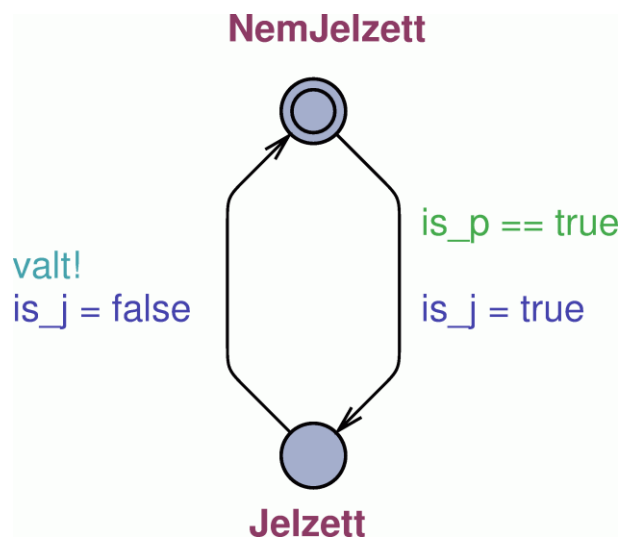
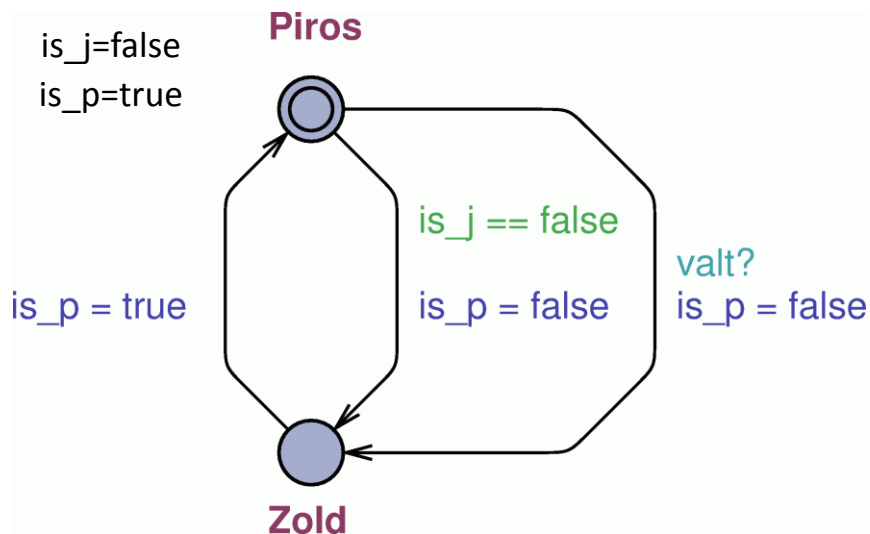
C'



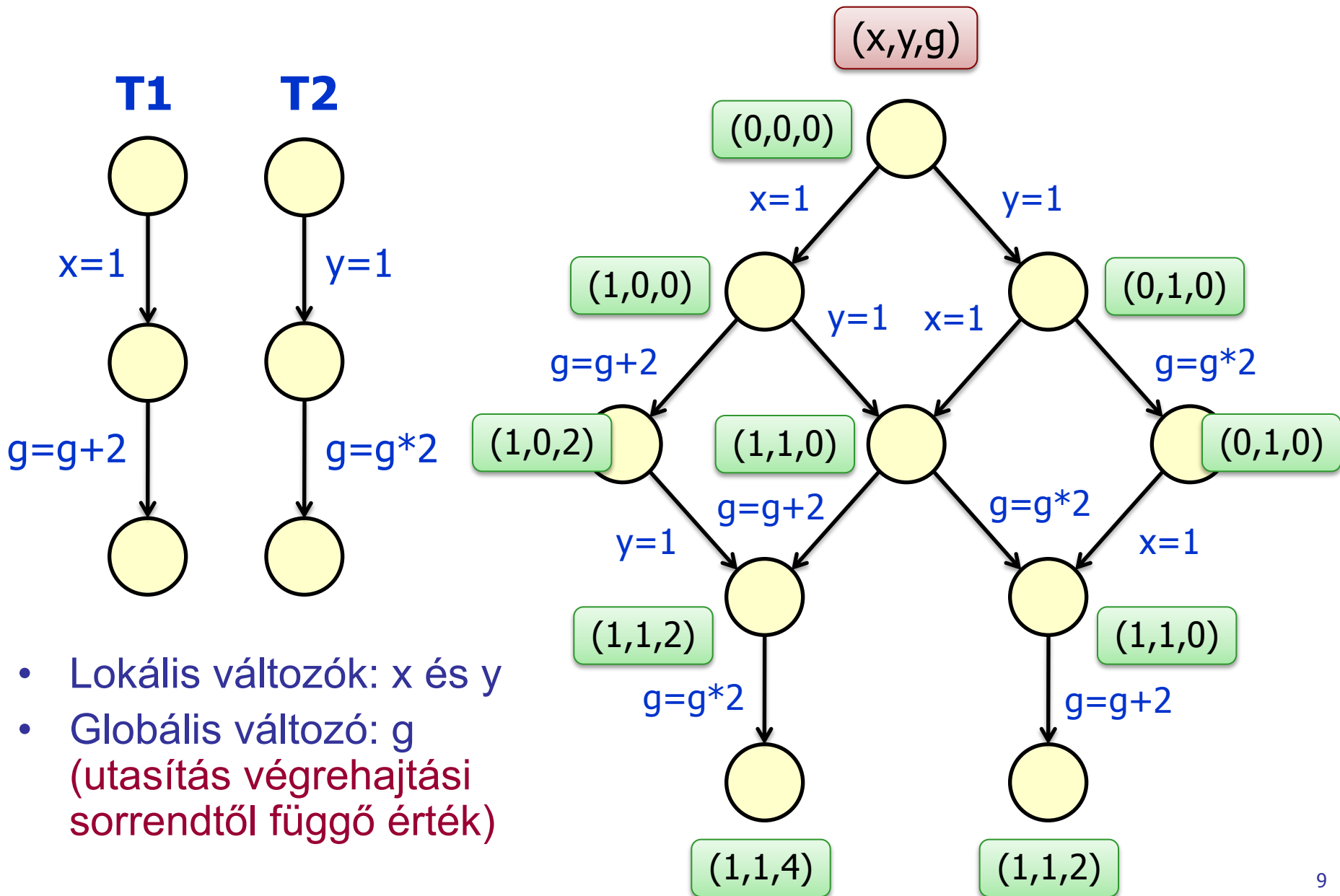
C''



Példa: Gyalogos lámpa jelzőgombbal



Példa: Változók figyelembe vétele



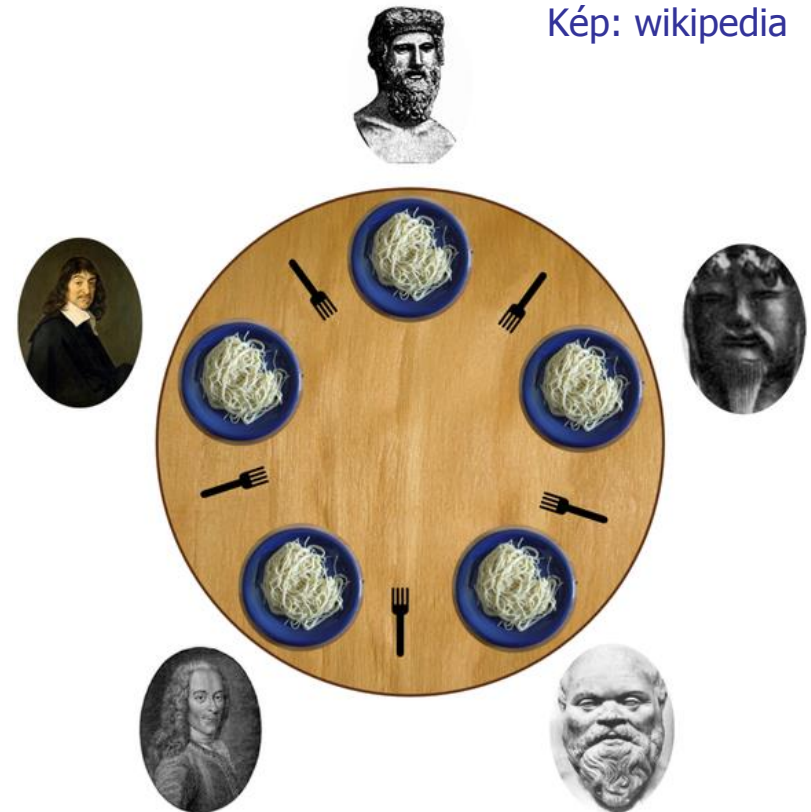
- Lokális változók: x és y
- Globális változó: g
(utasítás végrehajtási sorrendtől függő érték)

Példa nagy állapottérre: Étkező filozófusok

- Konkurens rendszer
 - Holtpont kialakulhat
 - Livelock kialakulhat
- Állapottér mérete gyorsan nő

Filozófusok száma	Állapottér mérete
16	$4,7 \cdot 10^{10}$
28	$4,8 \cdot 10^{18}$
...	...
200	$> 10^{40}$
1000	$> 10^{200}$
...	...

Ha 64 bittel címzek egy tárat, max.
 $2^{64} = 1,8 \cdot 10^{19}$ állapot tárolható



Okos (de nem feladat-specifikus)
állapottér tárolással:
kb. 100 000 filozófus, azaz
 10^{62900} állapottér is ellenőrizhető!

Áttekintés a megismerendő technikákról

- CTL modellellenőrzés: Szimbolikus technika

Állapothalmaz alapú technika	Szimbolikus technika
Címkézett állapothalmazok	Karakterisztikus függvények (Boole logikai függvények): ROBDD reprezentáció
Műveletek állapothalmazokon	Hatékony műveletek ROBDD-ken

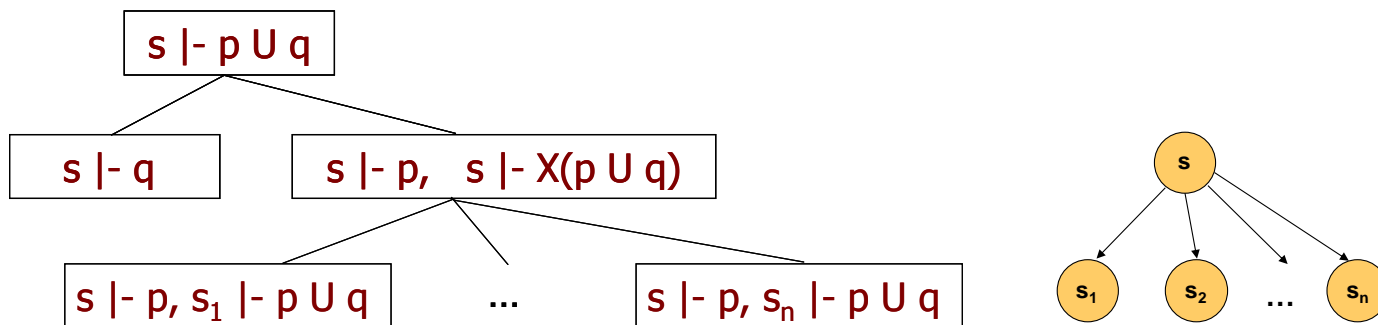
- Invariánsok modellellenőrzése: Korlátos modellellenőrzés
 - Logikai függvények igazságának keresése SAT technikával
 - Adott mélységig folytatható modellellenőrzés: Korlátos hosszúságú ellenpéldák keresése
 - Egy megtalált ellenpélda mindenképpen érvényes ellenpélda
 - Ha nincs ellenpélda, az nem végleges eredmény

Szimbolikus modellellenőrzés

Ismétlés: A modellellenőrzés tanult technikái

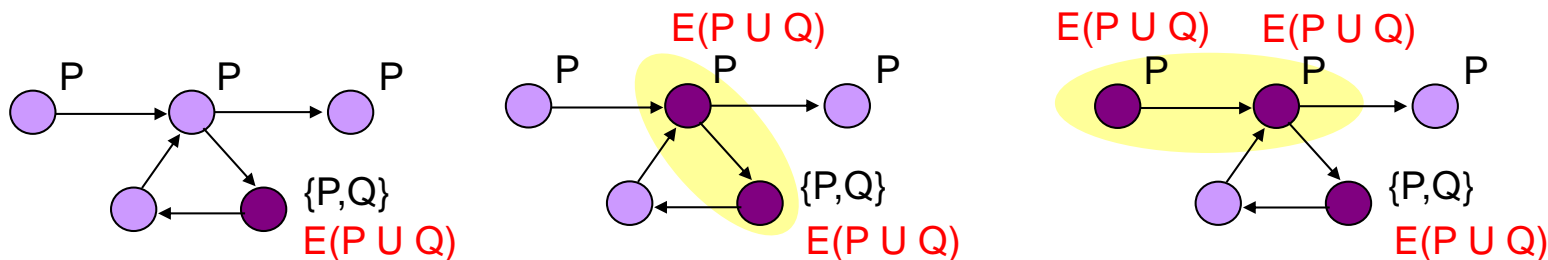
- PLTL modellellenőrzés:

- Tabló módszer: Kifejezések felbontása a modell mentén



- CTL modellellenőrzés:

- Szemantika alapú módszer: Állapotok iteratív címkézése



Ismétlés: CTL modellellenőrzés állapot címkézéssel

- Állapotok címkézése rész kifejezésekkel $Sat(..)$ állapothalmaz számítása alapján:

$$AF (P \wedge E (Q \cup R))$$

- Állapotok címkézése: Hol igaz egy adott kifejezés?
 - Kiindulás: KS címkézve van **atomi** kijelentésekkel
 - Tovább lépés: Címkézés az **összetettebb** kifejezésekkel
 - Ha p illetve q címkék már vannak, akkor megadható, hol lehet $\neg p$, $p \wedge q$, $EX p$, $AX p$, $E(p \cup q)$, $A(p \cup q)$ címke
 - Inkrementális címkézési algoritmus az operátorok szemantikája alapján

Ismétlés: $E(p \cup q)$ modellellenőrzés

$$\underline{E(p \cup q)} = \underline{q} \vee (\underline{p} \wedge \underline{EX E(p \cup q)})$$

Hová
rakható
 $E(p \cup q)$
címke?

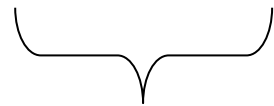
Ahol q
címke
már
van ...

vagy

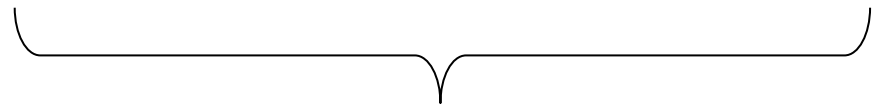
ahol p
címke
már
van ...

és

van legalább egy
rákövetkezője, ahol
már van $E(p \cup q)$
címke

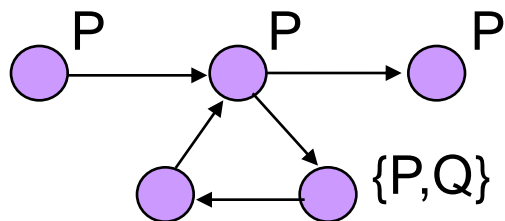


Címke
ránhelyezés
első lépése



Iteratívan bővíthető
a címkehalmaz
(amíg bővül)

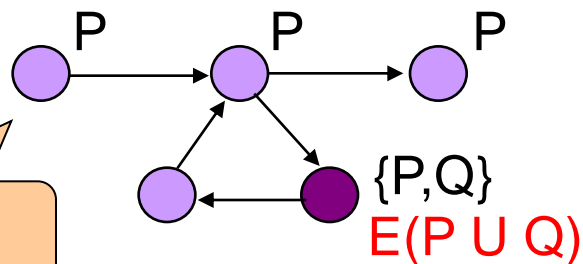
Ismétlés: Az $E(P \cup Q)$ címkézés iterációja



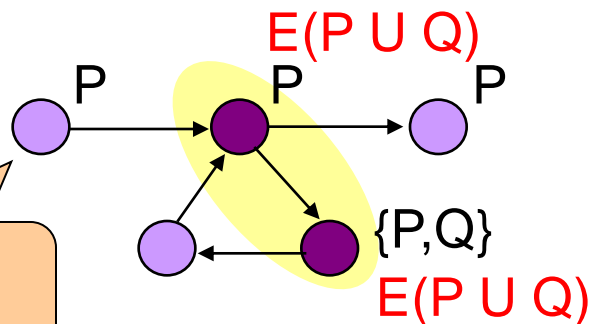
Kripke struktúra a kezdő címkézéssel

- Kihasználható:
 $E(P \cup Q) = Q \vee (P \wedge EX E(P \cup Q))$
- Az iteráció addig tart, míg nő az állapothalmaz (fixpontot érünk el)

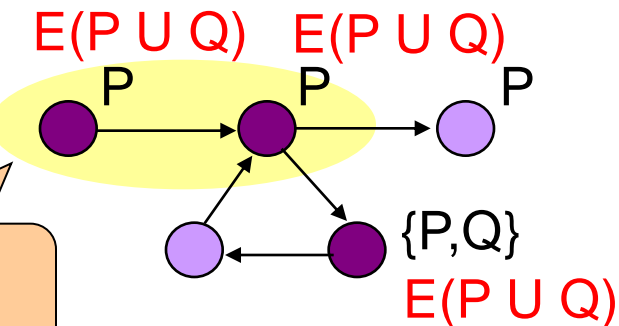
Első lépés: Q



Második lépés: $P \wedge EX$



Harmadik lépés: $P \wedge EX$



Ismétlés: Leírás halmazműveletekkel

- A címkézés bővítése halmazműveletekkel történik
 - Kezdőhalmaz: Rész-kifejezésekkel már címkézett állapotok
 - Címkézés bővítése:
 - $E(p \cup q)$ esetén: „p-vel címkézettek és legalább egy rákövetkező állapota már címkézett ...”
 - $A(p \cup q)$ esetén: „p-vel címkézettek és minden rákövetkező állapota már címkézett ...”
 - Ez alapján a megelőző állapotok valamelyikére tehető az új címke
- Jelölések:
 - Legalább egy rákövetkező állapota a címkézett Z halmazban van:
$$\text{pre}_E(Z) = \{s \in S \mid \text{létezik olyan } s', \text{ hogy } (s, s') \in R \text{ és } s' \in Z\}$$
 - Minden rákövetkező állapota a címkézett Z halmazban van:
$$\text{pre}_A(Z) = \{s \in S \mid \text{minden } s'\text{-re, ahol } (s, s') \in R: s' \in Z\}$$

Ismétlés: $E(p \cup q)$ modellellenőrzés algoritmus

$$\underline{E(p \cup q)} = \underline{q} \vee (\underline{p} \wedge \underline{EX E(p \cup q)})$$

Hová
rakható
 $E(p \cup q)$
címke?

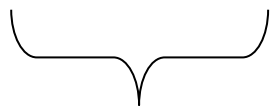
Ahol q
címke
már
van ...

vagy

ahol p
címke
már
van ...

és

van legalább egy
rákövetkezője, ahol
már van $E(p \cup q)$
címke



Címke
ránhelyezés
első lépése



Iteratívan bővíthető
a címkehalmaz
(amíg bővül)

$$Z_0 = \{s \mid q \in L(s)\}$$

$$Z_{i+1} = Z_i \cup (\{s \mid p \in L(s)\} \cap \text{pre}_E(Z_i))$$

A szimbolikus modellellenőrzés alapötlete

- Állapothalmazok tárolása: Az állapotok felsorolása helyett logikai függvényekkel megadva
 - Állapot „kódolása” n bites bitvektorral
 - S állapothalmaz kódolásához $n = \lceil \log_2 |S| \rceil$ bit elég, azaz válasszunk olyan n értéket, hogy legyen $2^n \geq |S|$
 - Állapothalmaz „kódolása” n -változós logikai (Boole) függvénnyel: Karakterisztikus függvény
 - Karakterisztikus függvény: $C: \{0,1\}^n \rightarrow \{0,1\}$
 - Egy állapothalmaz karakterisztikus függvénye akkor és csak akkor legyen igaz egy-egy bitvektor behelyettesítésére, ha a bitvektor által kódolt állapot az adott állapothalmazban van
- Állapothalmazokon végzett műveletek
 - Halmazok helyett a karakterisztikus függvényeken

Precízen: Karakterisztikus függvények

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

Legyen az s „kódolása” (u_1, u_2, \dots, u_n) bitvektor, itt $u_i \in \{0, 1\}$

Cél: $C_s(x_1, x_2, \dots, x_n)$ 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: $C_s(x_1, x_2) = \neg x_1 \wedge x_2$

- Egy $Y \subseteq S$ állapotthalmazra: $C_Y(x_1, x_2, \dots, x_n)$

Cél: $C_Y(x_1, x_2, \dots, x_n)$ akkor legyen igaz egy (u_1, u_2, \dots, u_n) behelyettesítésre, ha $(u_1, u_2, \dots, u_n) \in Y$

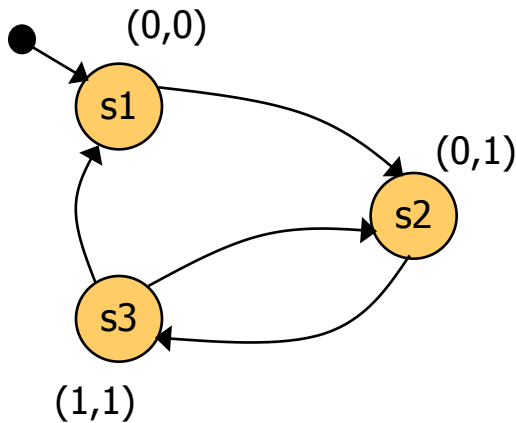
$C_Y(x_1, x_2, \dots, x_n)$ konstruálása:

$$C_Y(x_1, x_2, \dots, x_n) = \bigvee_{s \in Y} C_s(x_1, x_2, \dots, x_n)$$

- Állapothalmazokra általában:

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

Példa: Állapotok karakterisztikus függvénye



Változók: x, y

Állapotok karakterisztikus függvényei:

s1 állapot:

$$C_{s1}(x,y) = (\neg x \wedge \neg y)$$

s2 állapot:

$$C_{s2}(x,y) = (\neg x \wedge y)$$

s3 állapot:

$$C_{s3}(x,y) = (x \wedge y)$$

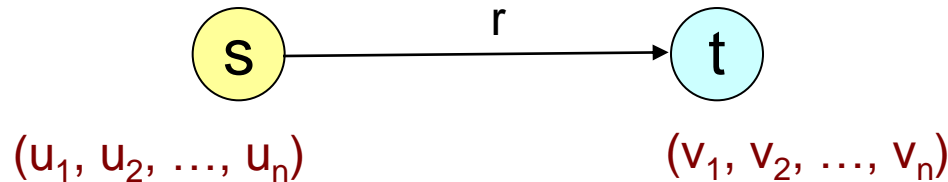
Állapothalmaz karakterisztikus függvénye:

{s1,s2} állapothalmaz:

$$C_{\{s1,s2\}} = C_{s1} \vee C_{s2} = (\neg x \wedge \neg y) \vee (\neg x \wedge y)$$

Állapotátmenetek karakterisztikus függvénye

- Egy r állapotátmenetre: C_r karakterisztikus függvény

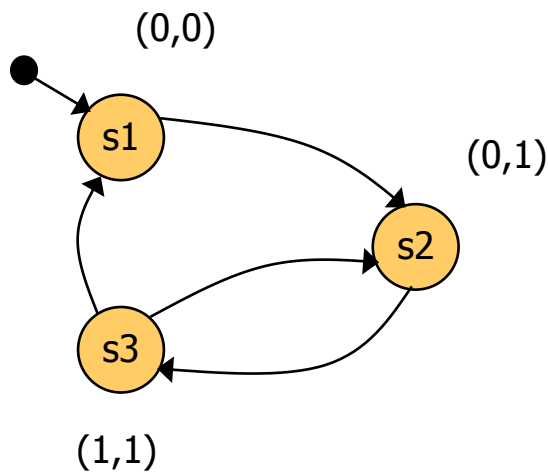


$r=(s,t)$ állapotátmenet, ahol $s=(u_1, u_2, \dots, u_n)$ és $t=(v_1, v_2, \dots, v_n)$

- Karakterisztikus függvény $C_r(x_1, x_2, \dots, x_n, x'_1, x'_2, \dots, x'_n)$ alakban
 - „Eredeti” n változó a forrás állapothoz
 - „Vesszős” n változó a cél állapothoz
- Cél: C_r a.cs.a. legyen igaz, ha a forrás és a cél állapot bitvektorát helyettesítjük be, azaz ha $x_i=u_i$ és $x'_i=v_i$ a behelyettesítés
 - C_r konstruálása:

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

Példa: Állapotátmenetek karakterisztikus függvénye



s1 állapot:

$$C_{s1}(x,y) = (\neg x \wedge \neg y)$$

s2 állapot:

$$C_{s2}(x,y) = (\neg x \wedge y)$$

$(s1,s2) \in R$ állapotátmenet:

$$C_{(s1,s2)} = (\neg x \wedge \neg y) \wedge (\neg x' \wedge y')$$

Állapotátmeneti reláció: Összes átmenet

$$\begin{aligned} R(x,y,x',y') = & (\neg x \wedge \neg y \wedge \neg x' \wedge y') \vee \\ & \vee (\neg x \wedge y \wedge x' \wedge y') \vee \\ & \vee (x \wedge y \wedge \neg x' \wedge y') \vee \\ & \vee (x \wedge y \wedge \neg x' \wedge \neg y') \end{aligned}$$

$\text{pre}_E(Z)$ halmaz karakterisztikus függvénye

- $\text{pre}_E(Z)$ 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

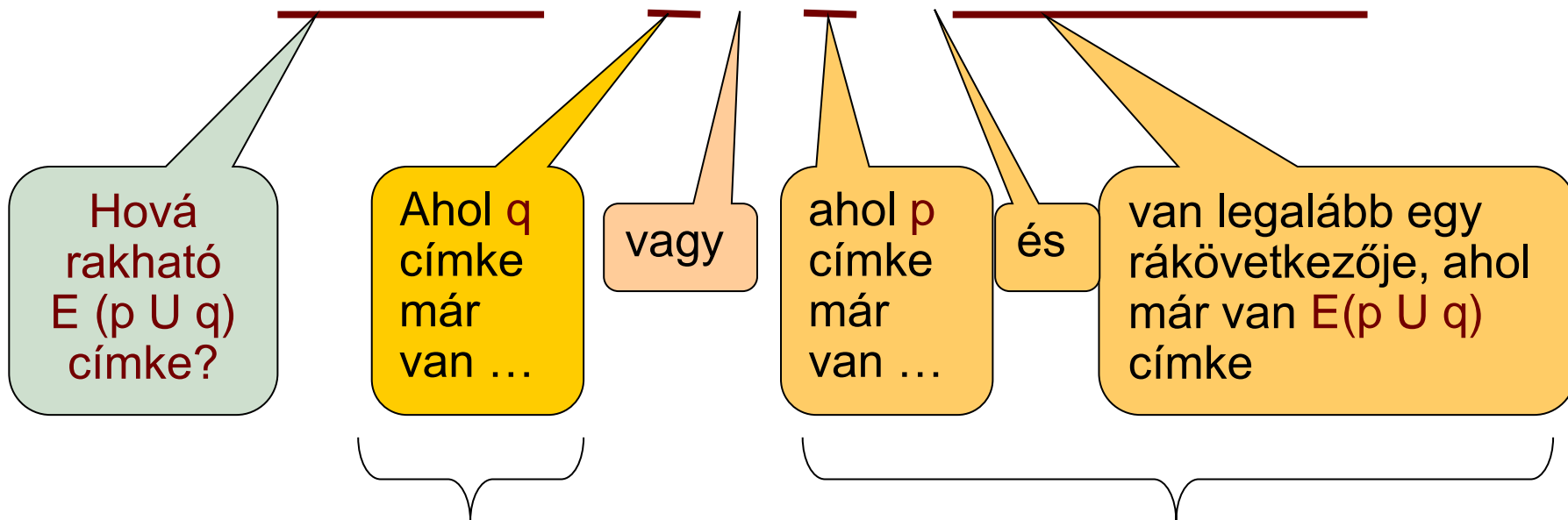
$$C_{\text{pre}_E(Z)} = \exists_{x'_1, x'_2, \dots, x'_n} C_R \wedge C'_Z$$

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

- Modellellenőrzés állapothalmaz műveletekkel:
visszavezetve logikai függvényeken végzett műveletekre!
 - Halmazok uniója: Függvények \vee kapcsolata
 - Halmazok metszete: Függvények \wedge kapcsolata
 - $\text{pre}_E(Z)$ képzése: Összetett művelet (egzisztenciális absztrakció)

E(p U q) modellellenőrzés algoritmus

$$\underline{E(p \cup q)} = \underline{q} \vee (\underline{p} \wedge \underline{EX E(p \cup q)})$$



Címke
ráhelyezés
első lépése

$$Z_0 = \{s \mid q \in L(s)\}$$

$$C_{Z_0} = C_q$$

Iteratíván bővíthető
a címkehalmaz
(amíg bővül)

$$Z_{i+1} = Z_i \cup (\{s \mid p \in L(s)\} \cap \text{pre}_E(Z_i))$$

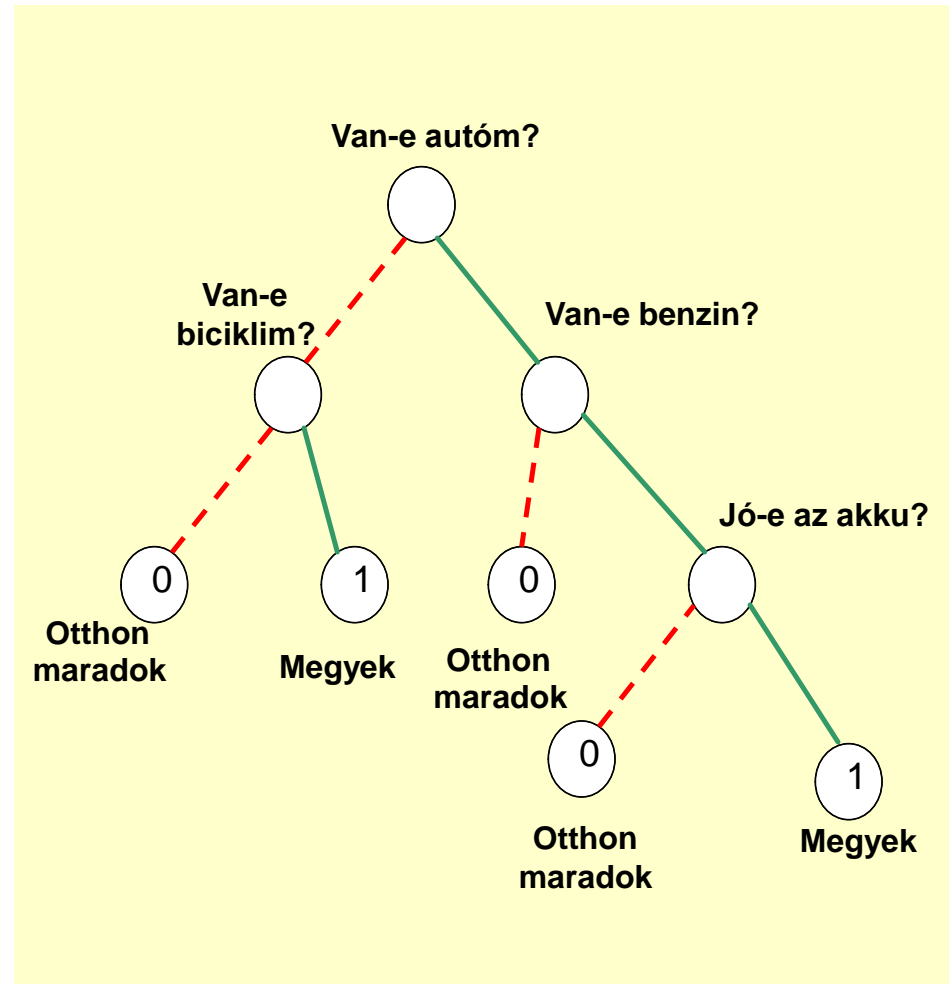
$$C_{Z_{i+1}} = C_{Z_i} \vee (C_p \wedge C_{\text{pre}_E(Z_i)})$$

Logikai függvények reprezentációja: Az ROBDD bevezetése

A bináris döntési fa

- Egy cél elérését több döntés befolyásolja
- Csomópontokban bináris döntések
 - Igen/Nem ágak
- Eredmény: Válasz a cél elérésére egy döntéssorozat után:
 - Igen (1) / nem (0)

Többértékű kiterjesztés is létezik



Boole függvények bináris döntési fa alakban

- Döntés: Egy x változó behelyettesítése a függvényben
 - „1” ág: A függvény $x=1$ behelyettesítéssel (jelölés: legyen f_x)
 - „0” ág: A függvény $x=0$ behelyettesítéssel (jelölés: legyen $f_{\underline{x}}$)
- Ez a Boole függvények Shannon felbontása:

$$f = x \rightarrow f_x, f_{\underline{x}}$$

- Jelölés: Az if-then-else szerkezet

$$x \rightarrow f_x, f_{\underline{x}} \equiv (x \wedge f_x) \vee (\neg x \wedge f_{\underline{x}})$$

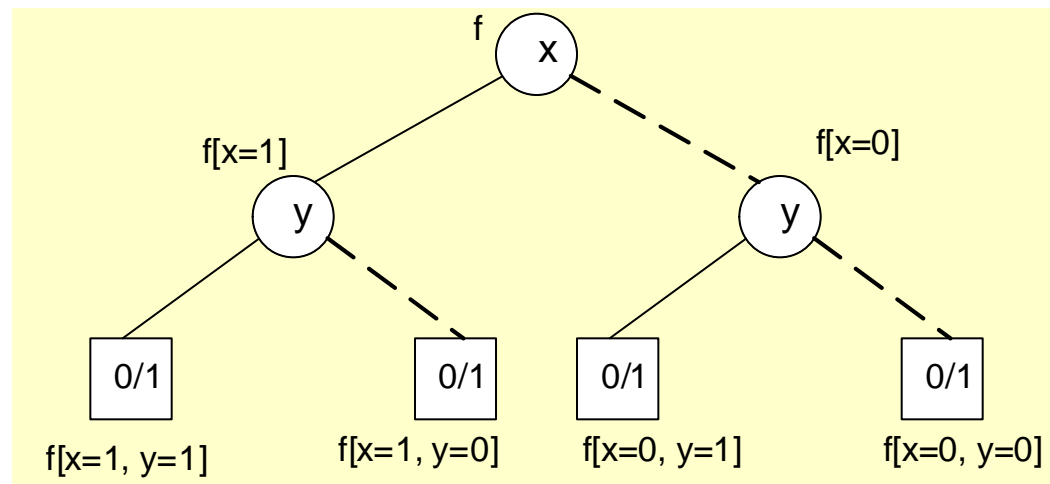
- x szokásos neve **tesztváltozó**, értékének vizsgálata a **teszt**
- Tehát a függvényt az if-then-else alapján felbonthatjuk
- A then-else ágakban ezzel egy változóját eltüntettük, redukáltuk
- Ciklikusan ismételjük, amíg van változó

Döntési fák típusai

Példa:

$f(x,y)$

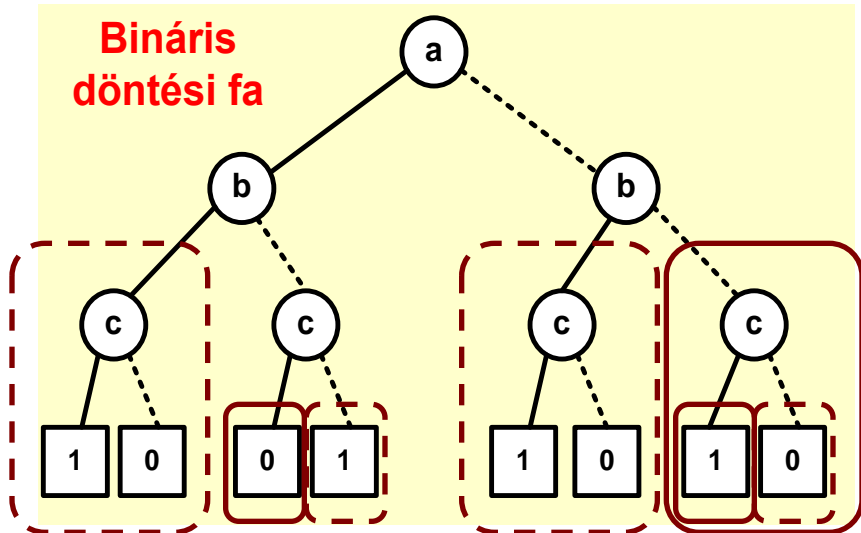
Ekkor a fán a négyzetekben határozhatók meg $f(x,y)$ lehetséges értékei



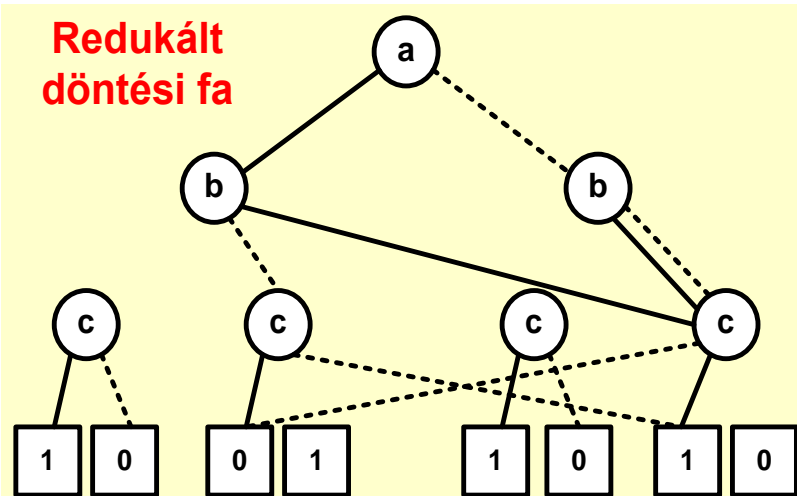
- Bináris döntési diagramot (BDD) kapunk, ha az azonos részfákat összevonjuk
- Rendezett bináris döntési diagramot (OBDD) kapunk, ha a felbontás során minden ágon azonos sorrendben vesszük fel a teszt változókat
- Redukált rendezett bináris döntési diagramot (ROBDD) kapunk, ha a szükségtelen csomópontokat töröljük

Példa: Egy bináris döntési fa átalakítása

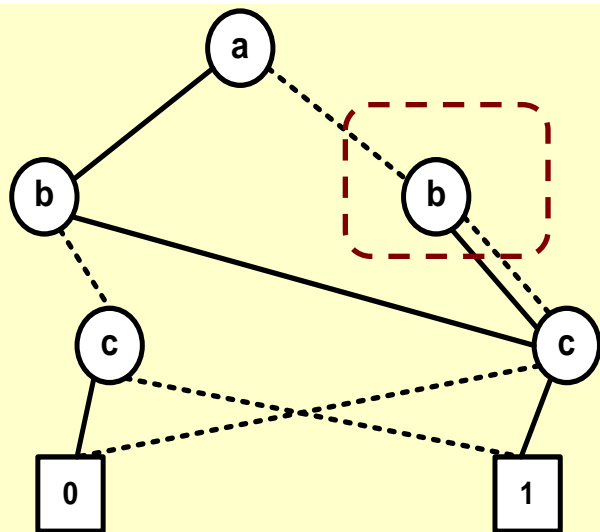
**Bináris
döntési fa**



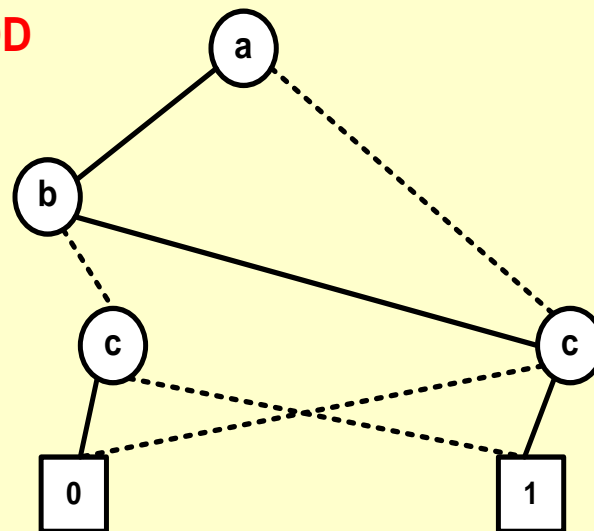
**Redukált
döntési fa**



BDD



ROBDD



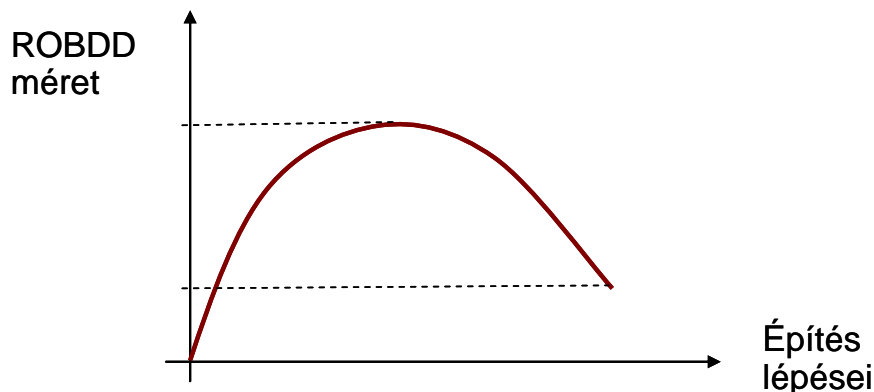
ROBDD tulajdonságok

- Irányított, aciklikus gráf, egy gyökérrel és két levéllel
 - A két levél értéke **1** vagy **0** (true vagy false)
 - Minden csomópontban egy-egy teszt változó van
- Minden csomópontból két él indul ki
 - Egyik a **0** behelyettesítésre (jelölés: szaggatott él)
 - Másik az **1** behelyettesítésre (jelölés: folytonos él)
- Minden útvonalon a teszt változók azonos sorrendben
- Az izomorf részfák egyetlen részfává összevonva
- Azon csomópontok, ahonnan kimenő él ugyanahhoz a csomóponthoz vezet, redukálva vannak

Egy adott függvény esetén két,
azonos változósorrendezésű ROBDD **izomorf**

ROBDD változók sorrendezése

- ROBDD mérete
 - Egyes függvények (pl. páros paritás) esetén nagyon kompakt
 - Más függvények esetén (pl. XOR) exponenciális méret
- A méret szempontjából nagyon fontos a változók sorrendjének megválasztása!
 - Más sorrend nagyságrendbeli különbséget is jelenthet
 - Optimális sorrend megtalálása NP-teljes probléma (→ heurisztika)
- Tárigény: Ha folyamatosan építjük a ROBDD-t, akkor az építés közben ideiglenes csomópontokat kell tárolnunk, amiket le lehet redukálni

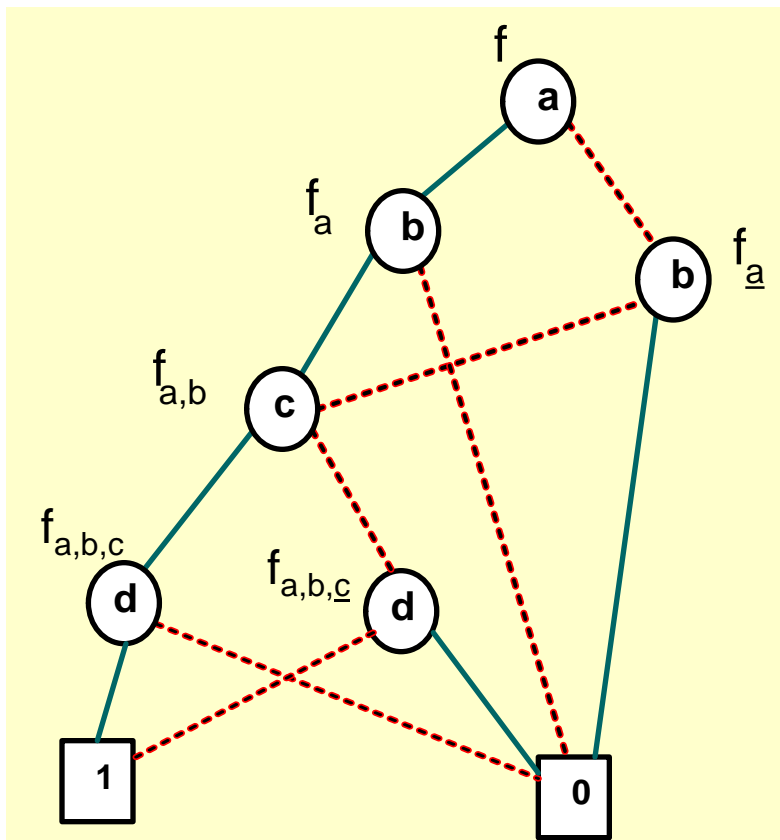


Példa: ROBDD kézi előállítása

Legyen az

$$f = (a \Leftrightarrow b) \wedge (c \Leftrightarrow d)$$

Sorrend legyen: a, b, c, d

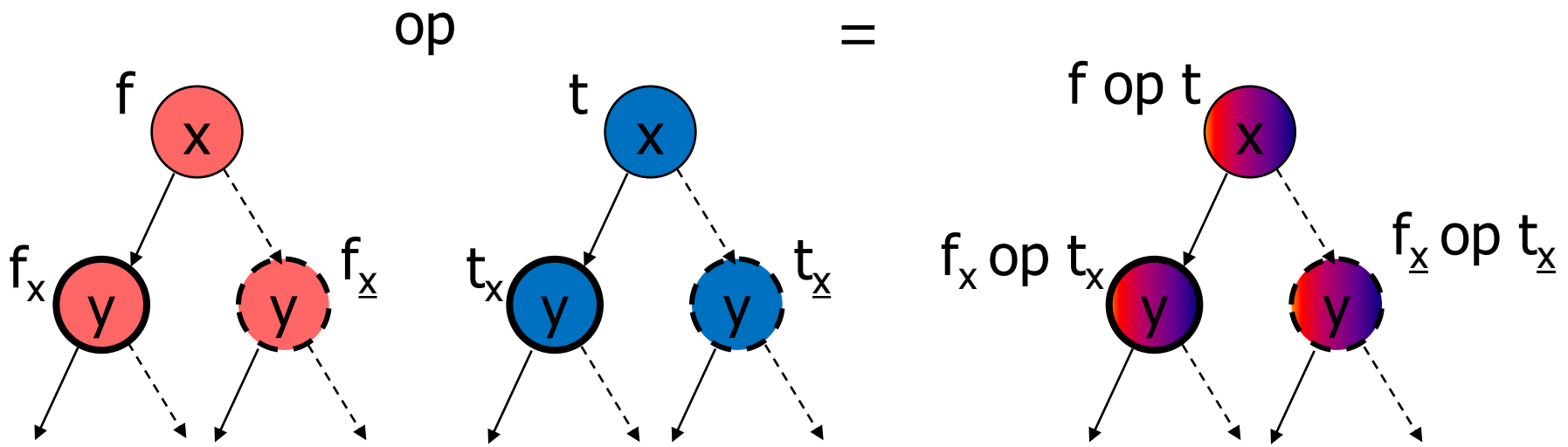


- $f = a \rightarrow f_a, \underline{f_a}$
 $f_a = (1 \Leftrightarrow b) \wedge (c \Leftrightarrow d), \underline{f_a} = (0 \Leftrightarrow b) \wedge (c \Leftrightarrow d)$
- $f_a = b \rightarrow f_{a,b}, \underline{f_{a,b}}$
 $f_{a,b} = (1 \Leftrightarrow 1) \wedge (c \Leftrightarrow d) = (c \Leftrightarrow d)$
 $\underline{f_{a,b}} = (1 \Leftrightarrow 0) \wedge (c \Leftrightarrow d) = 0$
- $f_a = b \rightarrow \underline{f_{a,b}}, \underline{\underline{f_{a,b}}}$
 $\underline{f_{a,b}} = (0 \Leftrightarrow 1) \wedge (c \Leftrightarrow d) = 0$
 $\underline{\underline{f_{a,b}}} = (0 \Leftrightarrow 0) \wedge (c \Leftrightarrow d) = (c \Leftrightarrow d)$
- $f_{a,b} = c \rightarrow f_{a,b,c}, \underline{f_{a,b,c}}$
 $f_{a,b,c} = (1 \Leftrightarrow d), \underline{f_{a,b,c}} = (0 \Leftrightarrow d)$
- $f_{a,b,c} = d \rightarrow f_{a,b,c,d}, \underline{f_{a,b,c,d}}$
 $f_{a,b,c,d} = (1 \Leftrightarrow 1) = 1,$
 $\underline{f_{a,b,c,d}} = (1 \Leftrightarrow 0) = 0$
- $f_{a,b,c} = d \rightarrow \underline{f_{a,b,c,d}}, \underline{\underline{f_{a,b,c,d}}}$
 $\underline{f_{a,b,c,d}} = (0 \Leftrightarrow 1) = 0, \underline{\underline{f_{a,b,c,d}}} = (0 \Leftrightarrow 0) = 1$

$f_{a,b}$ és $\underline{\underline{f_{a,b}}}$
izomorf!

Műveletek ROBDD-ken

- Boole operátorokat közvetlenül ROBDD-ken hajtjuk végre
 - A két függvény változói azonosak legyenek, azonos sorrendben
- Alap azonosság f , t függvényekre (itt op Boole operátor):
 - $f \text{ op } t = (x \rightarrow f_x, f_{\underline{x}}) \text{ op } (x \rightarrow t_x, t_{\underline{x}}) = x \rightarrow (f_x \text{ op } t_x), (f_{\underline{x}} \text{ op } t_{\underline{x}})$
Itt „1” ágat az „1” ággal, „0” ágat a „0” ággal kell bevonni



Műveletek ROBDD-ken (folytatás)

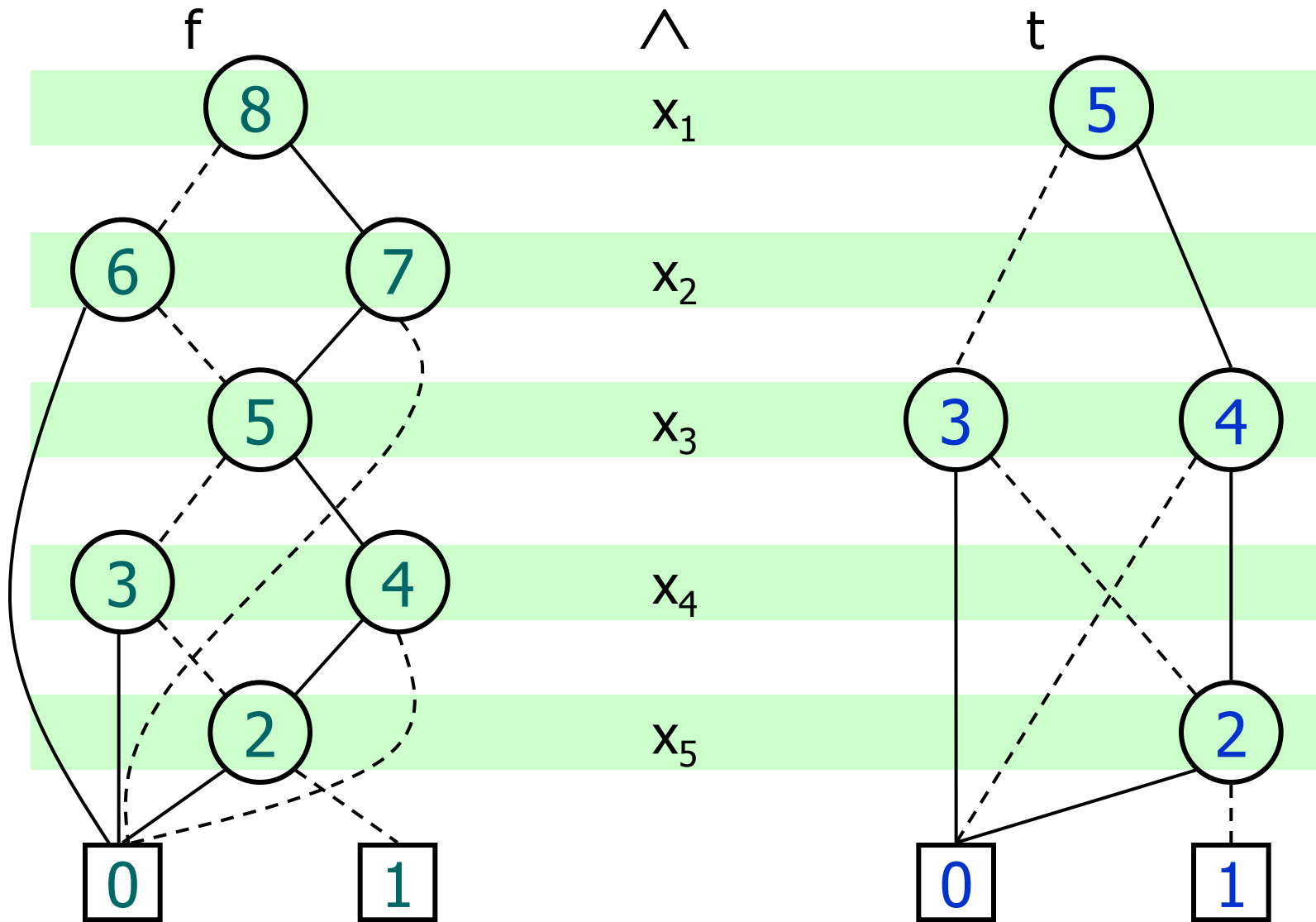
- Boole operátorokat közvetlenül ROBDD-ken hajtjuk végre
 - A két függvény változói azonosak legyenek, azonos sorrendben
- Alap azonosság f , t függvényekre (itt op Boole operátor):
 1. $f \text{ op } t = (x \rightarrow f_x, f_{\underline{x}}) \text{ op } (x \rightarrow t_x, t_{\underline{x}}) = x \rightarrow (f_x \text{ op } t_x), (f_{\underline{x}} \text{ op } t_{\underline{x}})$

Itt „1” ágat az „1” ággal, „0” ágat a „0” ággal kell bevonni
- További szabályok (redukálás miatt hiányzó változó esete):
 2. $f \text{ op } t = (x \rightarrow f_x, f_{\underline{x}}) \text{ op } t = x \rightarrow (f_x \text{ op } t), (f_{\underline{x}} \text{ op } t)$

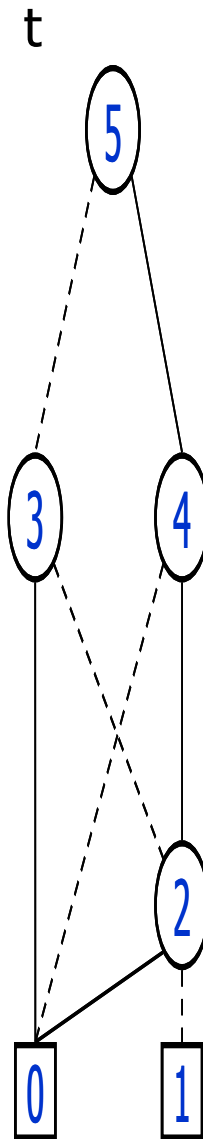
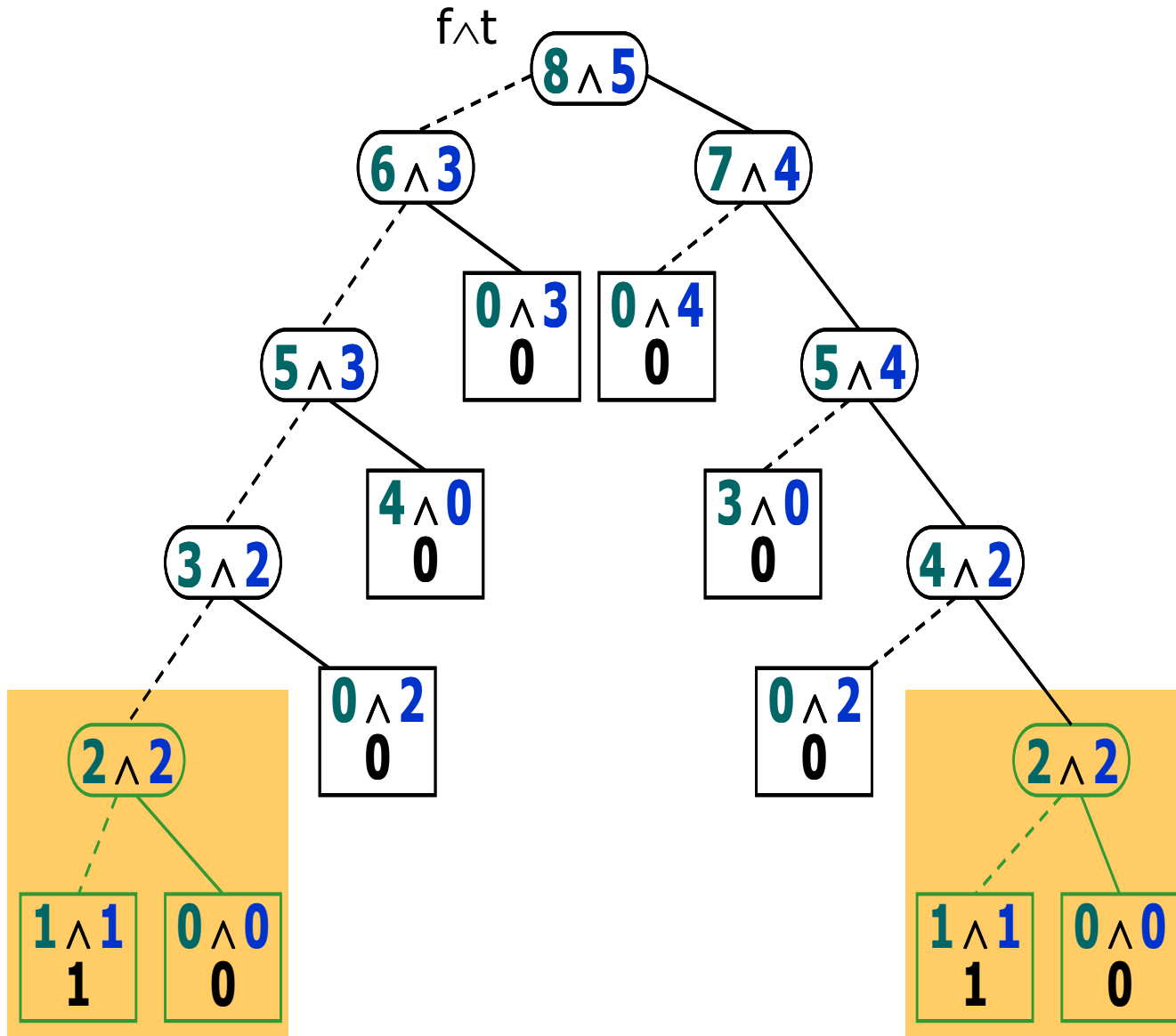
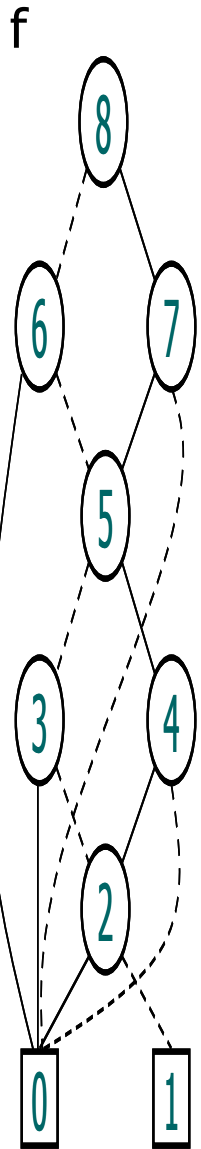
Itt t -ből hiányzik az x változó, nincsenek új ágak t -ben
 3. $f \text{ op } t = f \text{ op } (x \rightarrow t_x, t_{\underline{x}}) = x \rightarrow (f \text{ op } t_x), (f \text{ op } t_{\underline{x}})$

Itt f -ből hiányzik az x változó, nincsenek új ágak f -ben

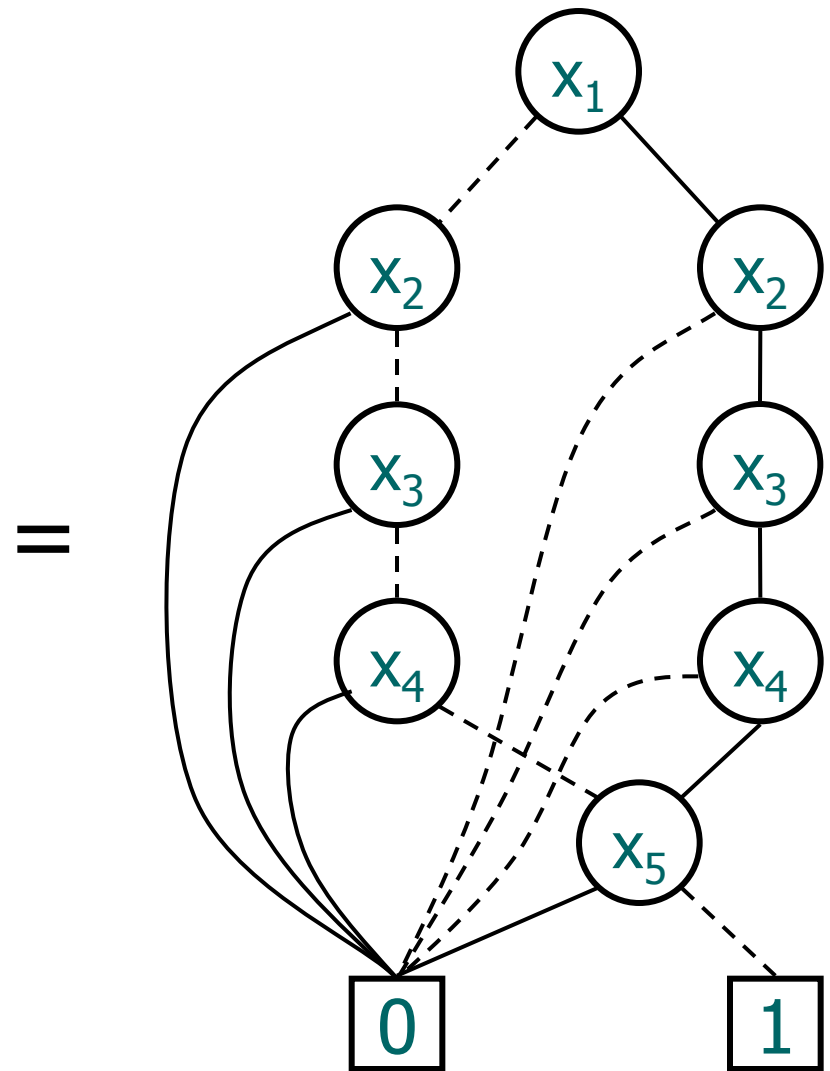
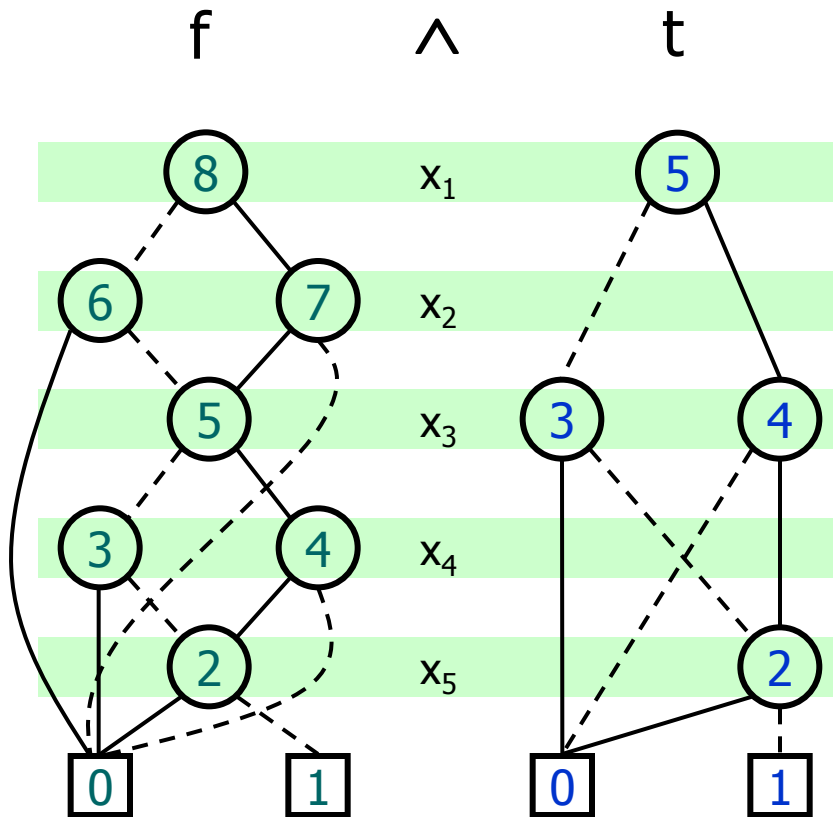
Példa: Művelet elvégzése ($f \wedge t$)



Példa: Művelet elvégzése ($f \wedge t$)



Példa: Művelet eredménye ($f \wedge t$)



Összefoglalás: Modellellenőrzés ROBDD-vel

- A modellellenőrzés megvalósítása:
 - Modellellenőrzés algoritmus: Műveletek állapotalmazokon (címkézés)
 - Szimbolikus technika: Állapothalmazok helyett logikai függvények
 - Hatékony megvalósítás: Logikai függvények ROBDD alakban kezelve
- Előnyök
 - A ROBDD kanonikus alak (függvények ekvivalenciája jól vizsgálható)
 - Algoritmusok hatékonyan gyorsíthatók
 - Tárigény csökken (változósorrend megválasztásától függ!)

Étkező filozófusok problémája:

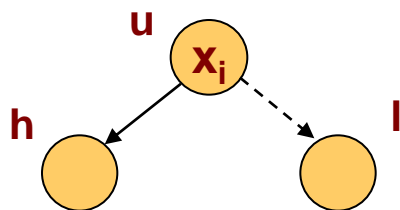
Filozófusok száma	Állapottér mérete	ROBDD csomópontok
16	$4,7 \cdot 10^{10}$	747
28	$4,8 \cdot 10^{18}$	1347

10^{18} méretű állapottér tárolása helyett kb. 21kB elég!

Az ROBDD gépi előállítása és műveletei (kiegészítő anyag)

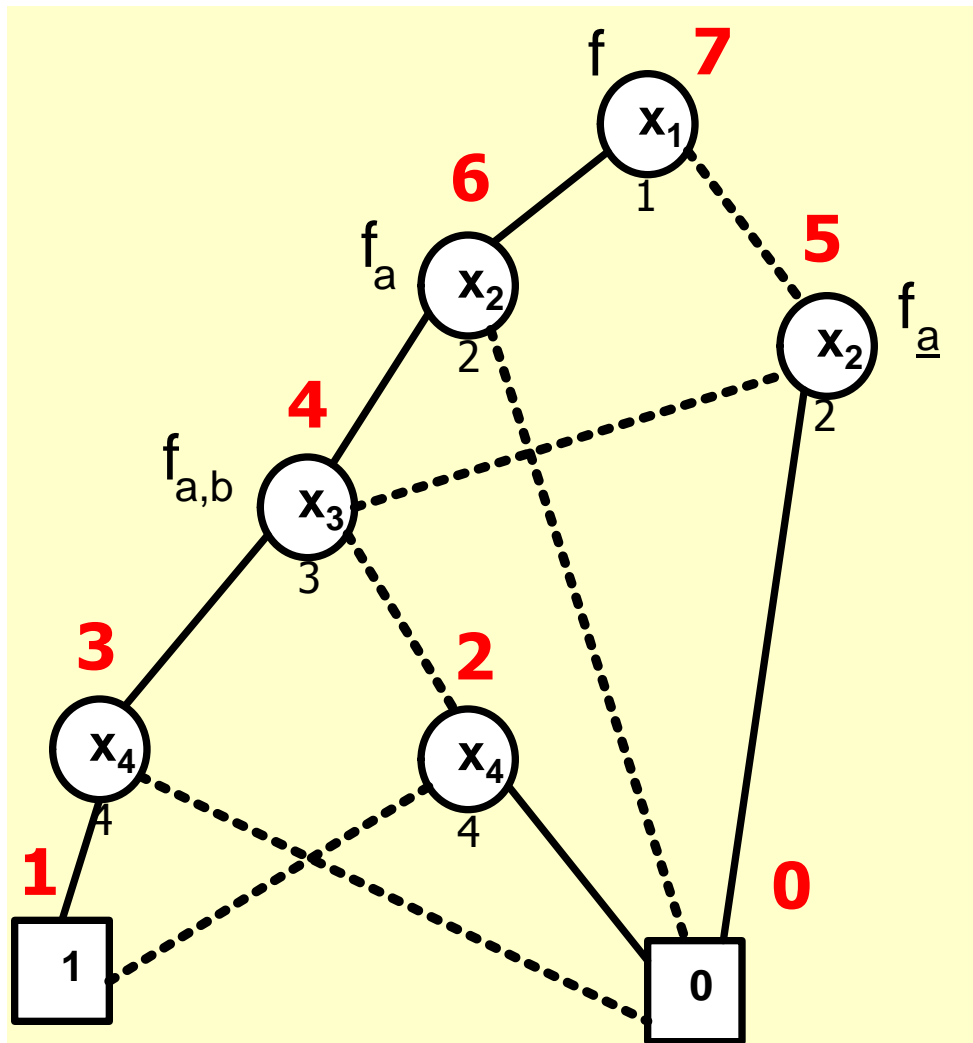
ROBDD gépi tárolása

- A ROBDD csomópontjait indexekkel azonosítjuk
- A ROBDD-t egy $T: u \rightarrow (i,l,h)$ táblázatban tároljuk:
 - u : csomópont indexe
 - i : a változó indexe ($x_i, i=1\dots n$)
 - l : a 0 behelyettesítési ágon elérhető csomópont indexe
 - h : az 1 behelyettesítési ágon elérhető csomópont indexe



u	i	l	h
0			
1			
2	4	1	0
3	4	0	1
4	3	2	3
5	2	4	0
6	2	0	4
7	1	5	6

ROBDD gépi tárolása



u	i	l	h
0			
1			
2	4	1	0
3	4	0	1
4	3	2	3
5	2	4	0
6	2	0	4
7	1	5	6

ROBDD gépi előállítása 1.

- Értelmezett műveletek:

- $\text{init}(T)$

- T kezdeti állapotát állítja be
- csak a 0 és 1 csomópontok vannak a táblázatban

- $\text{add}(T, i, l, h): u$

- egy új csomópontot készít T -ben az adott paraméterekkel
- ennek u indexét adja vissza

- $\text{var}(T, u): i$

- visszaadja T -ből az u csomópont változójának i indexét

- $\text{low}(T, u): l$ és $\text{high}(T, u): h$

- T -ben az u két behelyettesítési ágán levő csomópont l illetve h indexét adja vissza

ROBDD gépi előállítása 2.

- ROBDD csomópontjainak visszakereséséhez egy $H: (i,l,h) \rightarrow u$ segédtáblázatot is nyilvántartunk
- Műveletei:
 - $\text{init}(H)$
 - egy üres H táblázatot állít elő
 - $\text{member}(H,i,l,h):t$
 - ellenőrzi, hogy az (i,l,h) hármas szerepel-e H -ban; t Boole érték
 - $\text{lookup}(H,i,l,h):u$
 - kikeresi az (i,l,h) hármost a H táblázatban
 - visszaadja a hozzá tartozó u csomópont indexét
 - $\text{insert}(H,i,l,h,u)$
 - beilleszt egy új sort a táblázatba

ROBDD gépi előállítás 3.

Csomópont építése: $Mk(i,l,h)$

- Itt i a változó index,
 l és h az ágak
- Ha $l=h$, azaz azonos csomópontba vezetne a két él
 - akkor nem kell csomópontot létrehozni
 - bármelyik ágat vissza lehet adni
- Ha H -ban már van egy (i,l,h) hármas
 - akkor sem kell újat létrehozni
⇒ létezik izomorf részfa, ennek indexét kell visszaadni
- Ha nincsen H -ban ilyen (i,l,h)
 - akkor létre kell hozni, és visszaadni indexét

```
Mk(i,l,h) {  
    if l=h then  
        return l;  
    else if member(H,i,l,h) then  
        return lookup(H,i,l,h);  
    else {  
        u=add(T,i,l,h);  
        insert(H,i,l,h,u);  
        return u;  
    }  
}
```

ROBDD gépi előállítása 4.

ROBDD építése: `Build(f)` és a `Build'(t,i)` rekurzív segédfüggvény

```
Build(f) {  
  init(T); init(H);  
  return Build'(f,1);  
}
```

Rekurzívan végigmegy
majd a változókon

```
Build'(t,i) {  
  if i>n then  
    if t==false then return 0 else return 1  
  else {v0 = Build'(t[0/xi],i+1);  
        v1 = Build'(t[1/xi],i+1);  
        return Mk(i,v0,v1)}  
}
```

Terminális csomóponthoz értünk
(minden változó behelyettesítve)

Rekurzív építés;
Mk() ellenőrzi az
izomorf részfákat

Gépi műveletek ROBDD-ken

- Boole operátorokat közvetlenül ROBDD-ken hajtjuk végre
 - A két függvény változói azonosak legyenek, azonos sorrendben
- Alap azonosság f , t függvényekre (itt op Boole operátor):
 1. $f \text{ op } t = (x \rightarrow f_x, f_{\underline{x}}) \text{ op } (x \rightarrow t_x, t_{\underline{x}}) = x \rightarrow (f_x \text{ op } t_x), (f_{\underline{x}} \text{ op } t_{\underline{x}})$
- További szabályok (redukálás miatt hiányzó változó):
 2. $f \text{ op } t = (x \rightarrow f_x, f_{\underline{x}}) \text{ op } t = x \rightarrow (f_x \text{ op } t), (f_{\underline{x}} \text{ op } t)$
 3. $f \text{ op } t = f \text{ op } (x \rightarrow t_x, t_{\underline{x}}) = x \rightarrow (f \text{ op } t_x), (f \text{ op } t_{\underline{x}})$
- Ezen szabályok alapján definiálható $App(op, i, j)$ rekurzívan
 - ahol i, j : a művelet operandusainak ROBDD-jében a csomópontok
- Hátrány: lassú
 - worst-case 2^n exponenciális

Gyorsított műveletvégzés

- Legyen $G(op,i,j)$ egy gyorsítótáblázat, amely $App(op,i,j)$ eredményét tartalmazza (csomópont)
- Az algoritmus négy esete:
 - Mindkét csomópont **terminális**: ekkor egy új terminális hozható létre az operátorral végzett eredmény alapján
 - Ha a csomópontokhoz tartozó változó **indexe azonos**, akkor a **0** és az **1** behelyettesítésű ágak párosíthatóak az $App(op,i,j)$ alkalmazásából, az **1. azonosság szerint**
 - Ha az egyik csomóponthoz tartozó változó **indexe nagyobb**, akkor ezt párosítjuk a kisebb változó-indexű csomópont **0** és **1** ágaival a **2. vagy 3. azonosság szerint**

A műveletvégzés pszeudo-kódja

```
Apply(op, f, t) {
```

```
  init(G);
```

```
  return App(op, f, t);
```

```
}
```

```
App(op, u1, u2) {
```

```
  if (G(op, u1, u2) <> empty) then return G(op, u1, u2);
```

```
  else if (u1 in {0,1} and u2 in {0,1}) then u = op(u1, u2);
```

```
  else if (var(u1) = var(u2)) then
```

```
    u=Mk(var(u1), App(op, low(u1), low(u2)),
```

```
          App(op, high(u1), high(u2)));
```

```
  else if (var(u1) < var(u2)) then
```

```
    u=Mk(var(u1), App(op, low(u1), u2), App(op, high(u1), u2));
```

```
  else (* if (var(u1) > var(u2)) then *)
```

```
    u=Mk(var(u2), App(op, u1, low(u2)), App(op, u1, high(u2)));
```

```
  G(op, u1, u2)=u;
```

```
  return u;
```

```
}
```


Behelyettesítés ROBDD alakjának előállítása

Változók konstans behelyettesítése (ld. $\text{pre}_E(Z)$ is):

Itt az u alatti ROBDD-ben az x_j változó értéke b legyen

```
Restrict(u,j,b) {  
    return Res(u,j,b);  
}  
  
Res(u,j,b) {  
    if var(u) > j then return u;  
    else if var(u) < j then  
        return Mk(var(u),  
                  Res(low(u),j,b),  
                  Res(high(u),j,b));  
    else  
        if b=0 then  
            return Res(low(u),j,b)  
        else  
            return Res(high(u),j,b);  
}
```

Ha lejjebb vagyok a behelyettesítendő változónál, marad az eredeti részfa

Ha feljebb vagyok a behelyettesítendő változónál, rekurzív építés kell

Ha ott vagyok a behelyettesítendő változónál, behelyettesítés kell