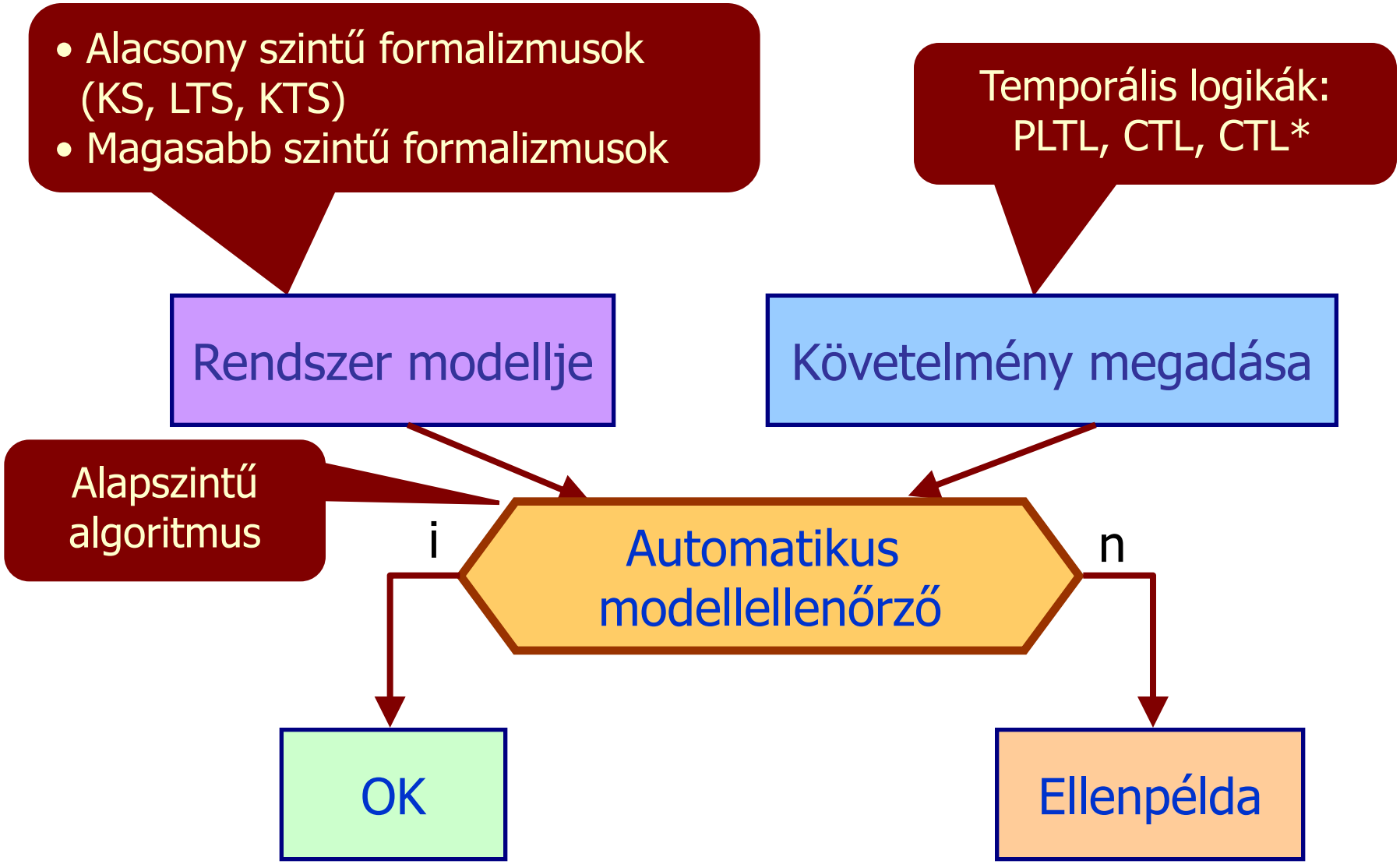
Hatékony technikák modellellenőrzéshez: Szimbolikus technikák (ROBDD)

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

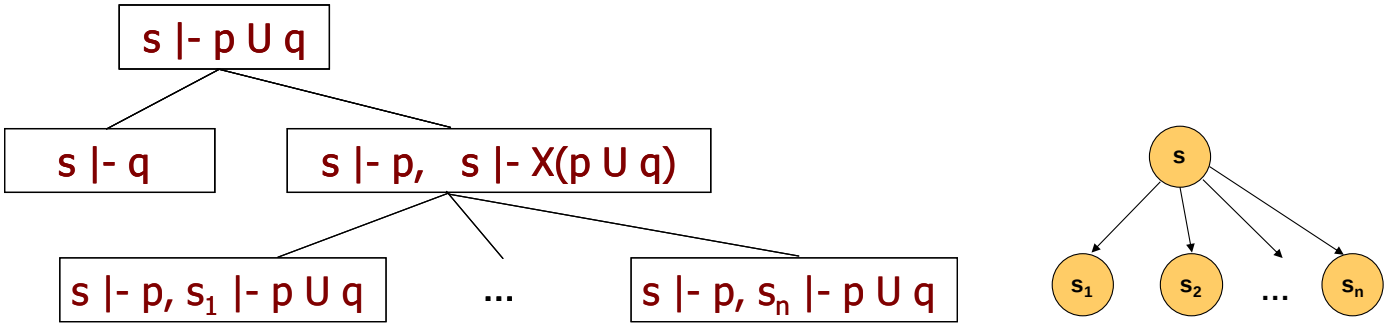
BME Méréstechnika és Információs Rendszerek Tanszék

Hol tartunk?

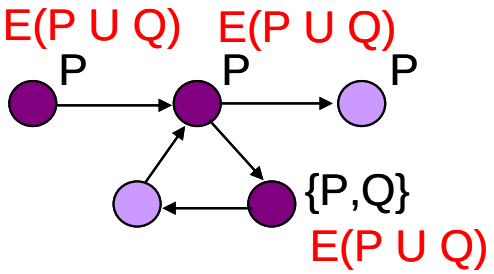

• Alacsony szintű formalizmusok (KS, LTS, KTS)
• Magasabb szintű formalizmusok
Temporális logikák: PLTL, CTL, CTL*
Rendszer modellje
Követelmény megadása
Alapszintű algoritmus
i
Automatikus modellellenőrző
n
OK
Ellenpélda

Ismétlés: A modellellenőrzés tanult technikái

- PLTL modellellenőrzés:
  - Tabló módszer: Kifejezések felbontása a modell mentén

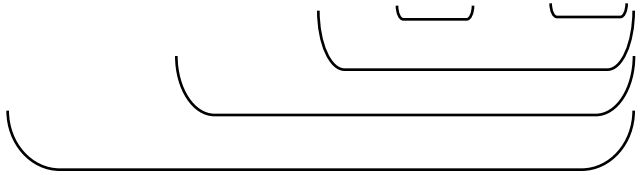


  - Automata alapú megközelítés (kiegészítő anyag)
- CTL modellellenőrzés:
  - Szemantika alapú módszer: Állapotok iteratív címkézése



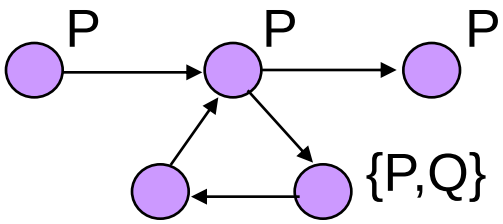
Ismétlés: CTL modellellenőrzés állapot címkézéssel

- Állapotok címkézése részkifejezésekkel Sat(..) állapothalmaz számítása alapján:
  AF (P $\wedge$ E (Q U R))



- Állapotok címkézése: Hol igaz egy adott kifejezés?
  - Kiindulás: KS címkézve van atomi kijelentésekkel
  - Továbblépés: 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 U q), A(p U q) címke
    - Inkrementális címkézési algoritmus az operátorok szemantikája alapján

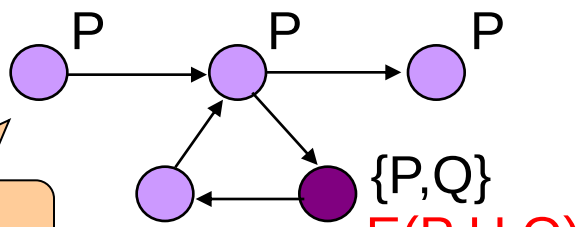
Ismétlés: Az E(P U Q) címkézés iterációja



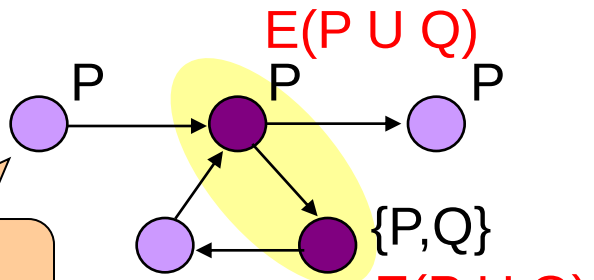
Kripke struktúra a kezdő címkézéssel

- Kihasználható:
  E(P U Q) =
  Q $\vee$ (P $\wedge$ EX E(P U Q))

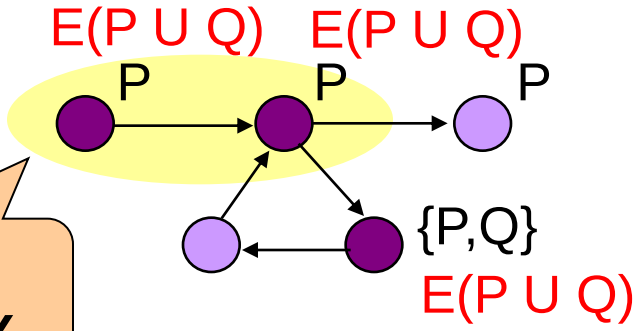
- Az iteráció addig tart, míg nő az állapothalmaz (fixpontot érünk el)



Első lépés: Q



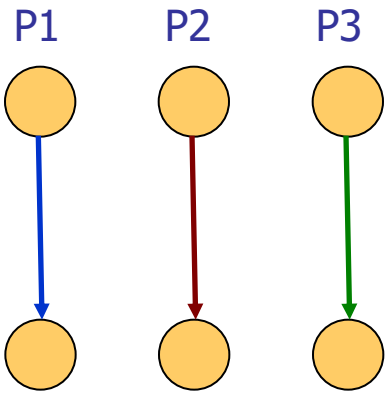
Második lépés: P $\wedge$ EX

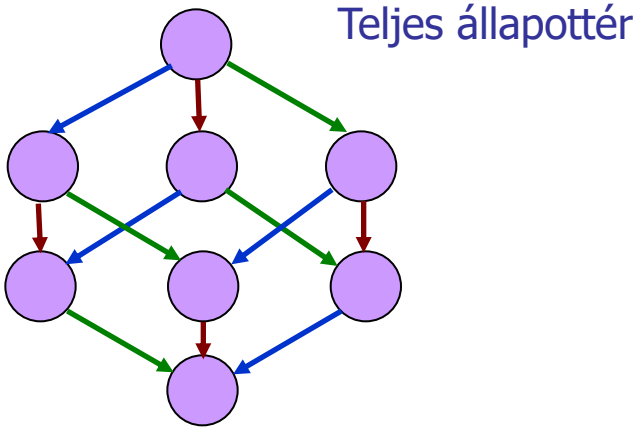


Harmadik lépés: P $\wedge$ EX

Problémák

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





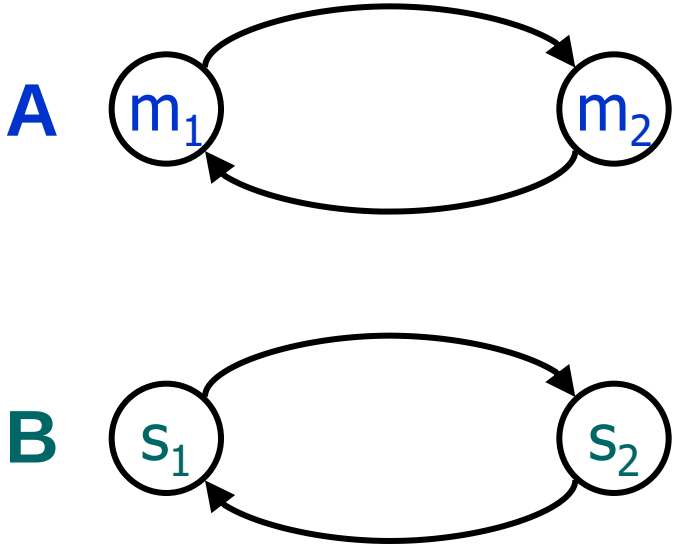
- Hogyan lehet nagy méretű modelleket ellenőrizni?
  - Ígéret: CTL modellellenőrzés: 10^{20}, egyes esetekben 10^{100} állapot
  - Milyen technika tudja ezt biztosítani?

Kitérő: Két automata együttes működése

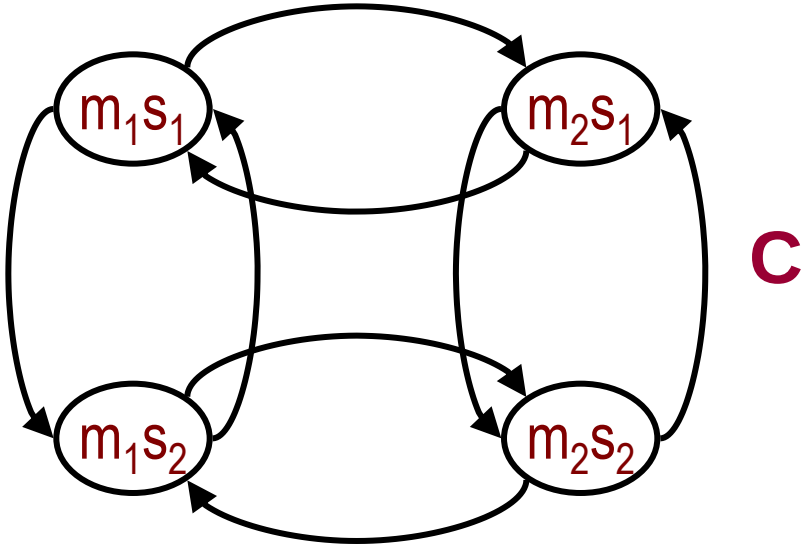
Automaták direkt szorzata, átlapolás, szinkronizáció

Példa: Aszinkron automaták működése

- Két (független) automatából álló rendszer



- (Direkt) szorzatautomata: a rendszer állapottere



- Az automaták állapotai:

  $A = \{m_1, m_2\}$, $B = \{s_1, s_2\}$

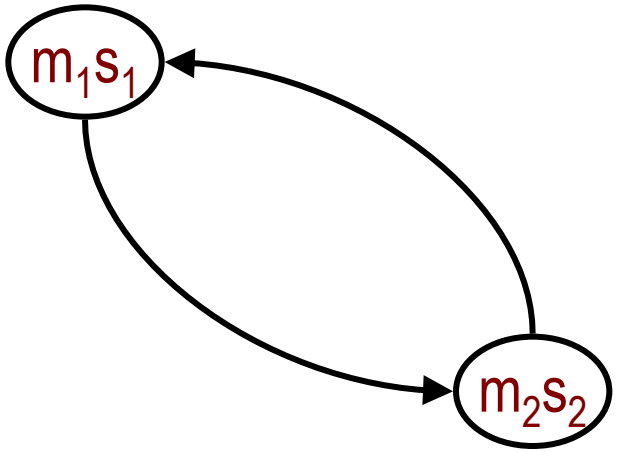
- Az állapotok halmaza:

  $C = A \times B$

  $C = \{m_1s_1, m_1s_2, m_2s_1, m_2s_2\}$

Átlapolás korlátozása: Szinkronizáció, feltétel

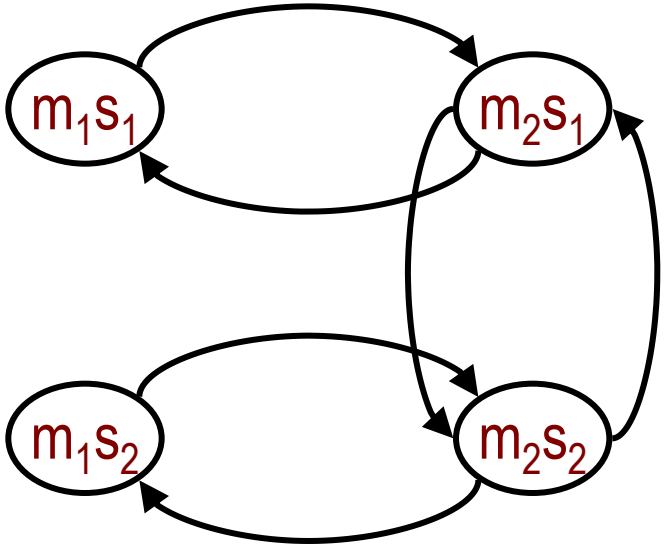
- Egyidejű állapotváltás: szinkronizáció



C’

- pl. „A és B egyszerre vált állapotot az azonos indexű állapotokból”

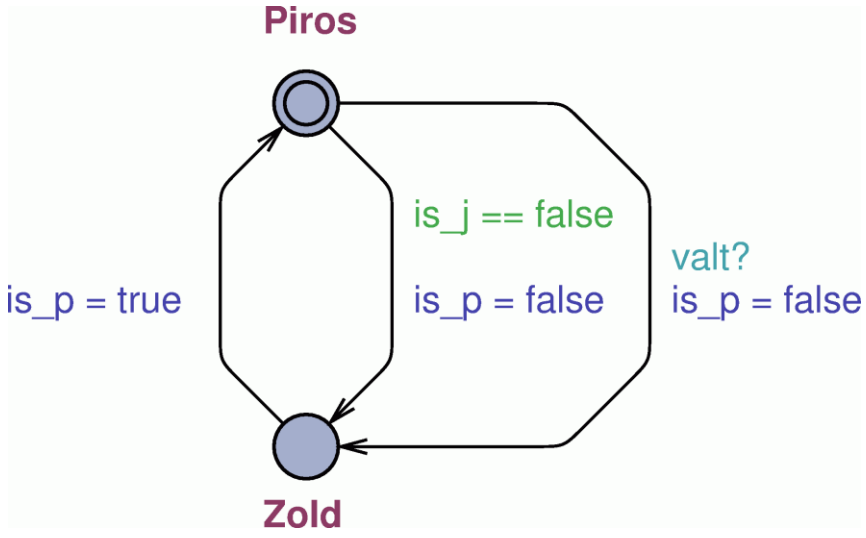
- Korlátozó feltételek: állapotátmenetek tiltása

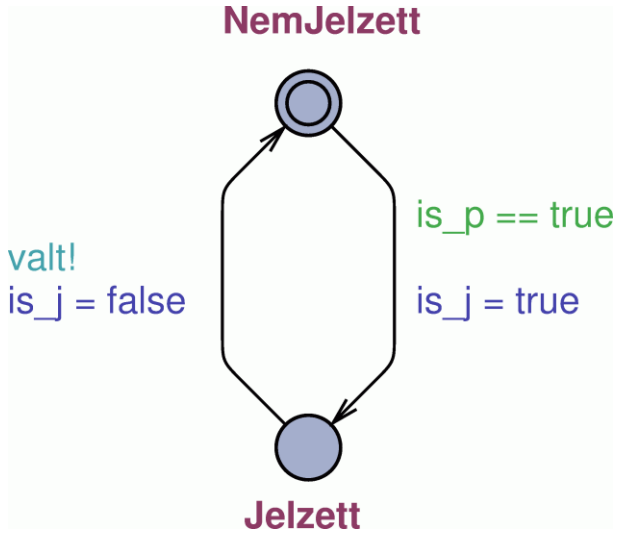


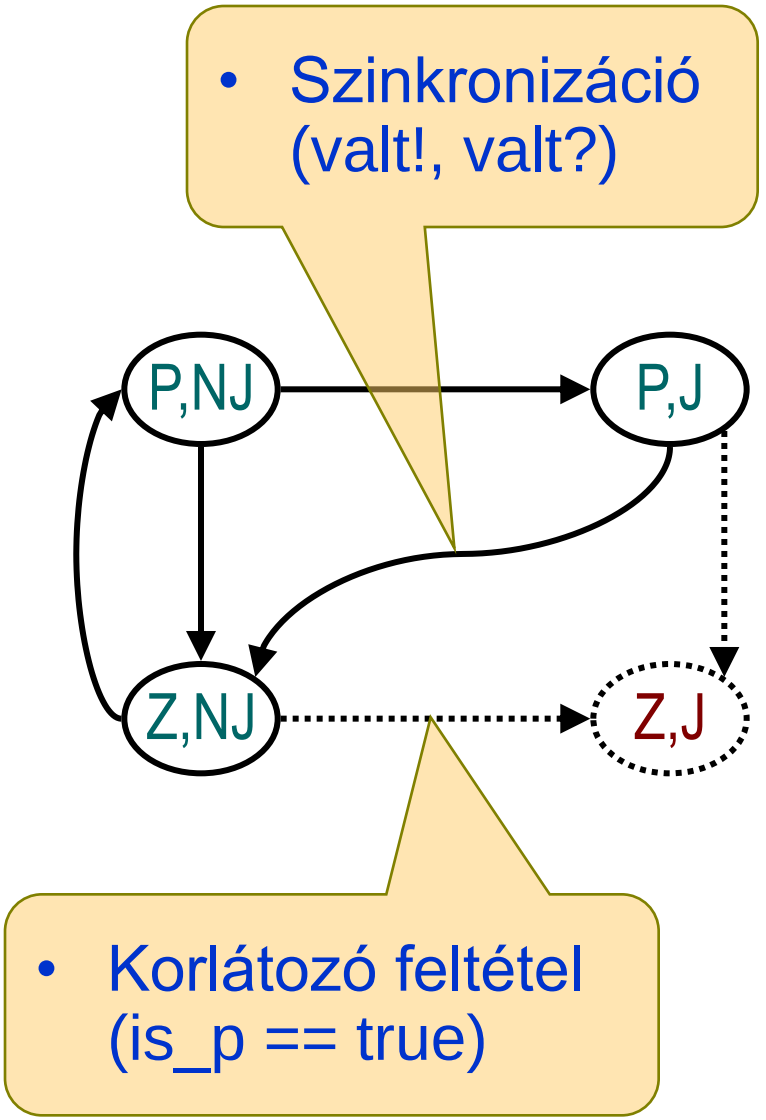
C”

- pl. „B csak akkor vált állapotot, ha A m_2 állapotban van”

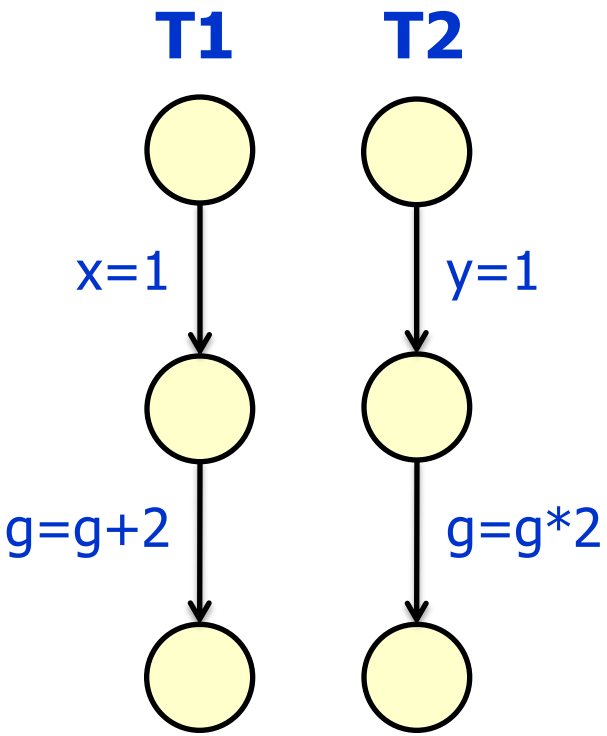
Példa: Gyalogos lámpa jelzőgombbal





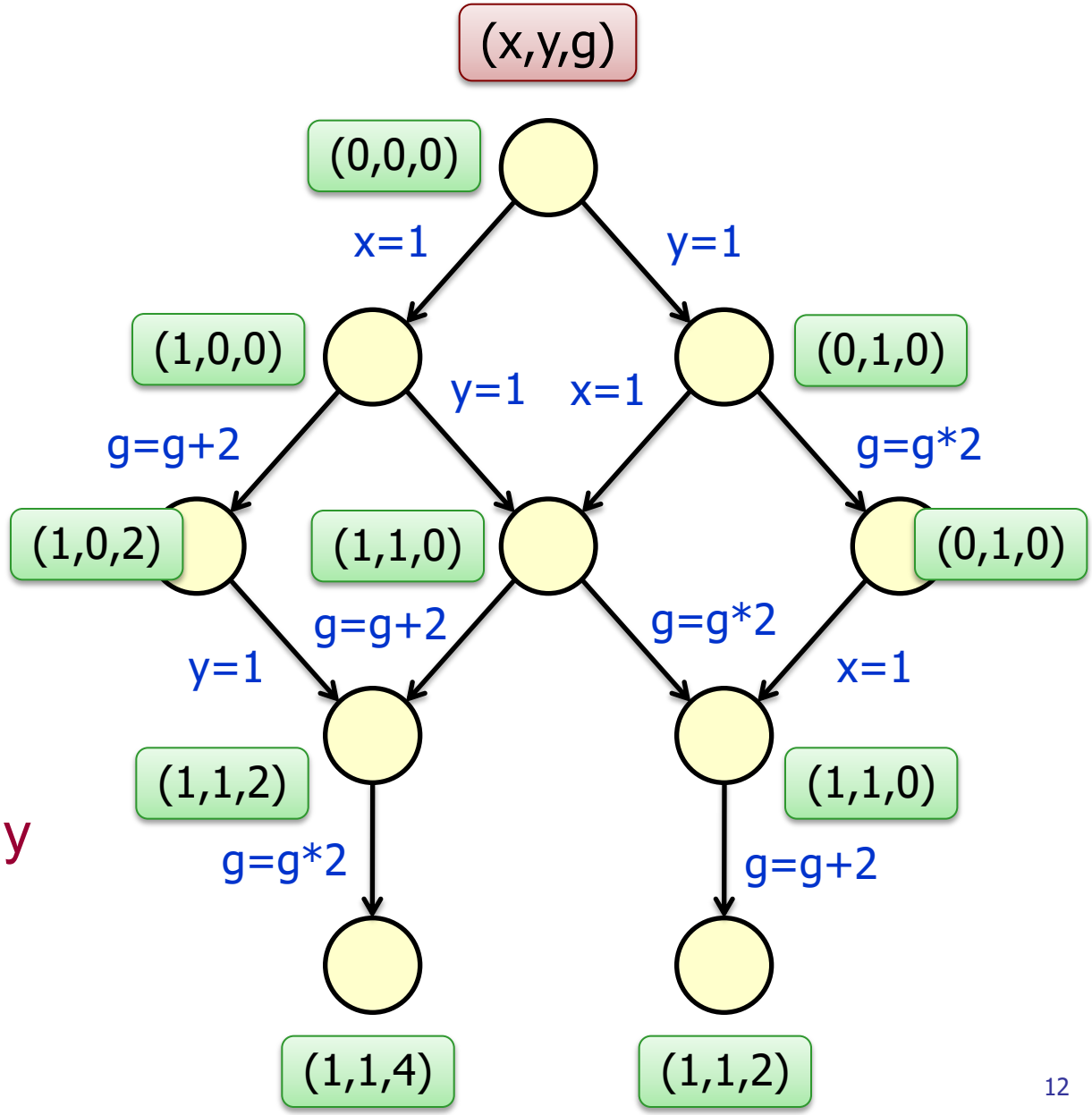


Példa: Különböző utak hatása



Lokális változók: x és y

Globális változó: g



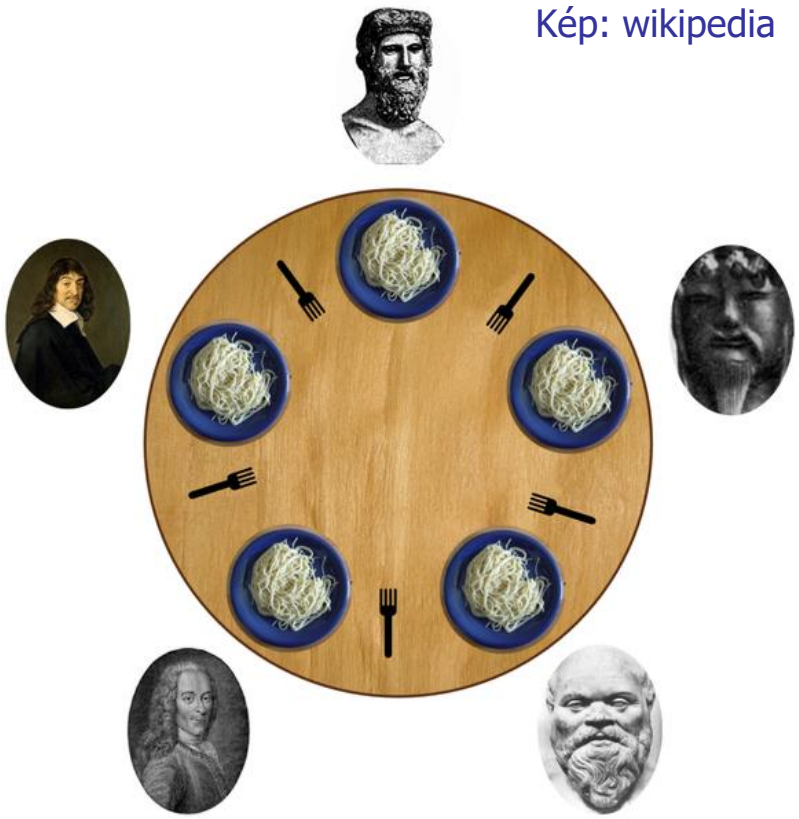
Példa nagy állapottérre: Étkező filozófusok

- 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	$> 10^{200}$
…	…

$2^{64} = 1{,}8 \cdot 10^{19}$

Kép: wikipedia



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

Szemantika 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: Iteráció halmazműveletekkel

- A címkézés bővítése halmazműveletekkel történik
  - Kezdőhalmaz: Rész-kifejezésekkel már címkézett állapotok
  - Címkézés bővítése:
    - E(p U q) esetén: „Legalább egy rákövetkező állapota már címkézett …”
    - A(p U q) esetén: „Minden rákövetkező állapota már címkézett …”
  - Ez alapján a megelőző állapotok valamelyikére tehető az új címke
- Megelőző állapotok jelölése már címkézett Z halmazhoz:

  $pre_E(Z) = \{s \in S \mid$ létezik olyan s', hogy $(s,s') \in R$ és $s' \in Z\}$

  $pre_A(Z) = \{s \in S \mid$ minden s'-re, ahol $(s,s') \in R$: $s' \in Z\}$
- Példa: E(P U Q) iterációja:
  - Kezdőhalmaz: $Z_0 = \{s \mid Q \in L(s)\}$
  - Iteráció: $Z_{i+1} = Z_i \cup (pre_E(Z_i) \cap \{s \mid P \in L(s)\})$

Eddig címkézettek plusz …

… az eddig címkézettek olyan megelőző állapotai, amelyek ...

... P-vel címkézettek

  - Iteráció vége: Ha $Z_{i+1} = Z_i$, azaz nem bővül a halmaz

Alapötlet

- Állapothalmazok tárolása és műveletei: Az állapotok felsorolása helyett logikai függvényekkel
  - Állapot „kódolása” 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
    - A logikai függvény akkor legyen 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 műveleteket végezni a halmazok helyett

Karakterisztikus függvények

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

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

  Cél: $C_s(x_1, x_2, \ldots, x_n)$ csak az $(u_1, u_2, \ldots, u_n)$ esetén adjon 1 értéket

  $C_s(x_1, x_2, \ldots, x_n)$ 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: $C_Y(x_1, x_2, \ldots, x_n)$

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

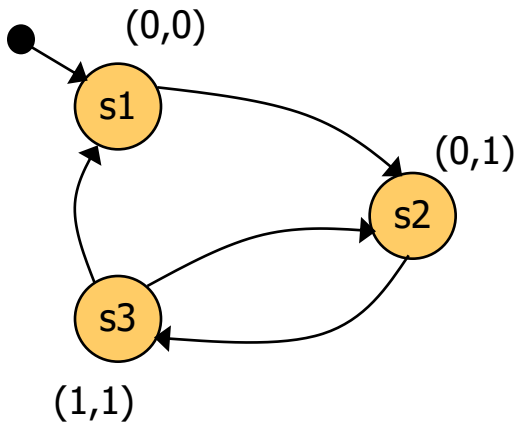
  $C_Y(x_1, x_2, \ldots, x_n)$ konstruálása:

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

- Állapothalmazokra általában:

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

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:

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

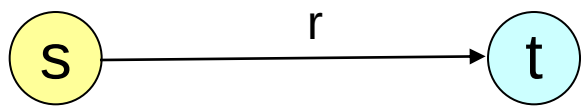
Állapothalmaz karakterisztikus függvénye:

{s1,s2} állapothalmaz:

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

Karakterisztikus függvények (folytatás)

- Állapotátmenetekre: C_r



$(u_1, u_2, \ldots, u_n)$ $(v_1, v_2, \ldots, v_n)$

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

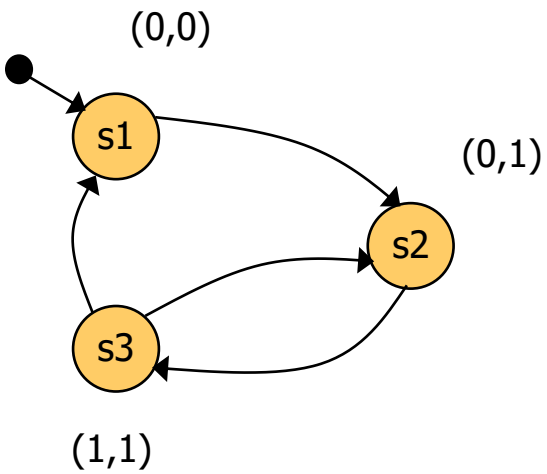
- Karakterisztikus függvény $C_r(x_1, x_2, \ldots, x_n, x'_1, x'_2, \ldots, x'_n)$ alakban
  - „Vesszős” változók jelzik a cél állapotot

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

C_r konstruálása:

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

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:

$$R(x,y,x',y') = (\neg x \wedge \neg y \wedge \neg x' \wedge y') \vee (\neg x \wedge y \wedge x' \wedge y') \vee (x \wedge y \wedge \neg x' \wedge y') \vee (x \wedge y \wedge \neg x' \wedge \neg y')$$

Karakterisztikus függvények (folytatás)

- $pre_E(Z)$ képzése: $pre_E(Z)=\{s \mid \exists t: (s,t)\in R \text{ és } t\in Z\}$

  Z reprezentációja: C_Z

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

  $pre_E(Z)$: kikeresni a Z-beli állapotokra az előzőeket

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

  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
  - $pre_E(Z)$ képzése: Összetett művelet (egzisztenciális absztrakció)

Logikai függvények reprezentációja

Kanonikus forma: ROBDD

Reduced, Ordered Binary Decision Diagram

Redukált, rendezett, bináris döntési diagram

Lépések (áttekintés):

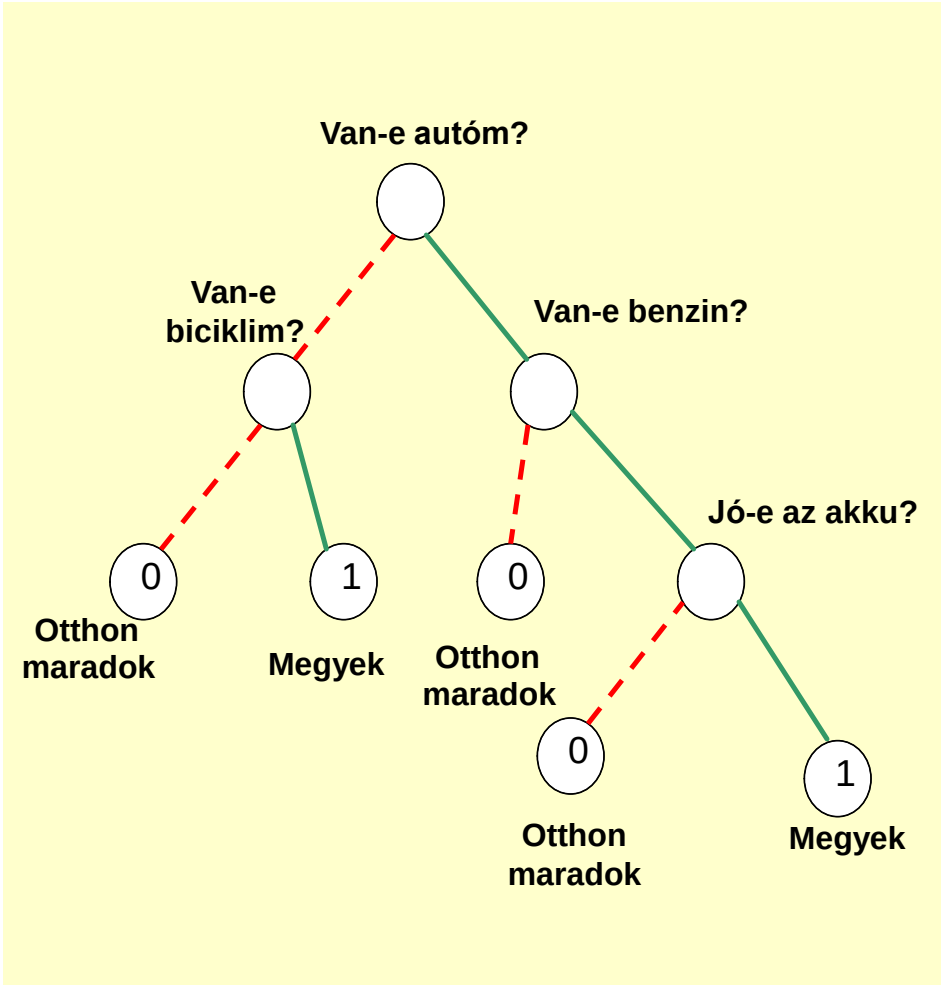
- Bináris döntési fa: bináris döntések ábrázolása
- BDD: azonos részfák összevonva
- OBDD: minden ágon azonos változósorrendezés
- ROBDD: szükségtelen csomópontok redukálása
  - Ha mindkét ág ugyanahhoz a csomóponthoz vezet

Az ROBDD-ről részletesen

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

- Mindegyik változó behelyettesítése egy-egy döntés
- Jelölés bevezetése: Az if-then-else szerkezet

$$x \rightarrow f_1, f_0 = (x \wedge f_1) \vee (\neg x \wedge f_0)$$

  - f_1 értéket veszi fel a kifejezés ha x igaz (true, 1)
  - f_0 értéket veszi fel a kifejezés ha x nem igaz (false, 0)
  - x szokásos neve tesztváltozó, értékének vizsgálata a teszt
- Boole függvények Shannon felbontása:

$$f = x \rightarrow f\,[1/x],\ f\,[0/x]$$

legyen $f_x = f\,[1/x]$; $f_{\underline{x}} = f\,[0/x]$

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

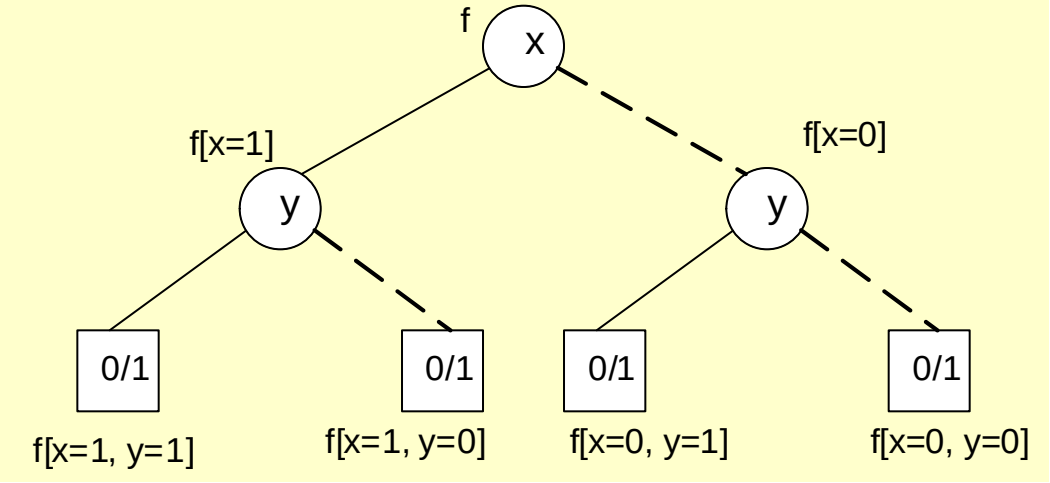
  - Tehát a függvényt az if-then-else alapján felbonthatjuk
  - A then-else ágakban ezzel egy változóját eltüntettük, redukáltuk
  - Ciklikusan ismételjük, amíg van változó

Döntési fák típusai

Példa:

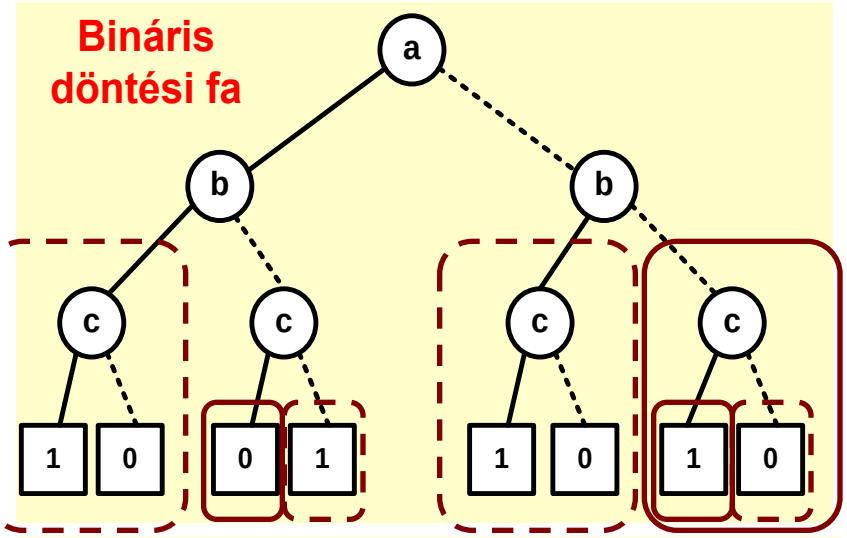
f(x,y)

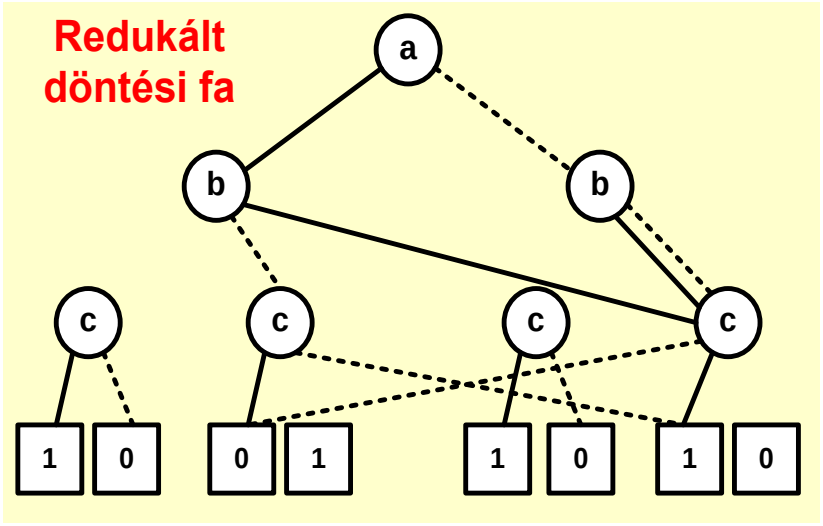
Ekkor a fán a négyzetekben határozhatók meg f(x,y) lehetséges értékei

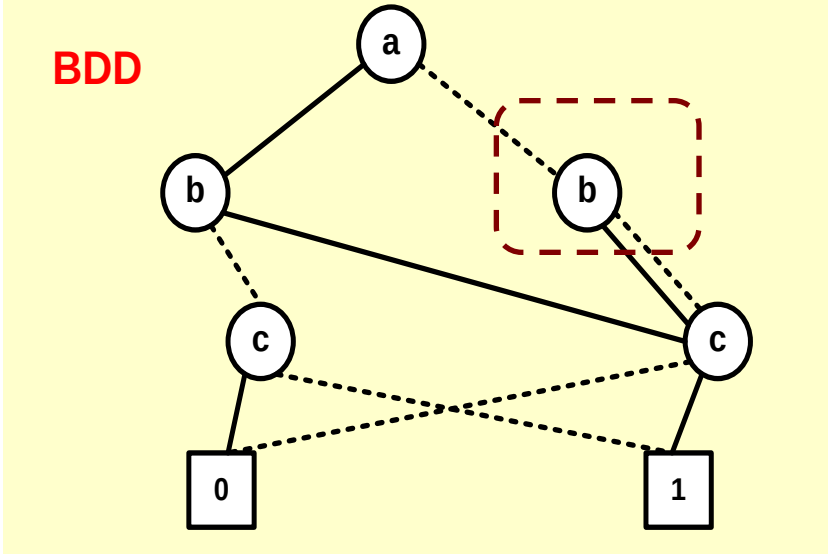


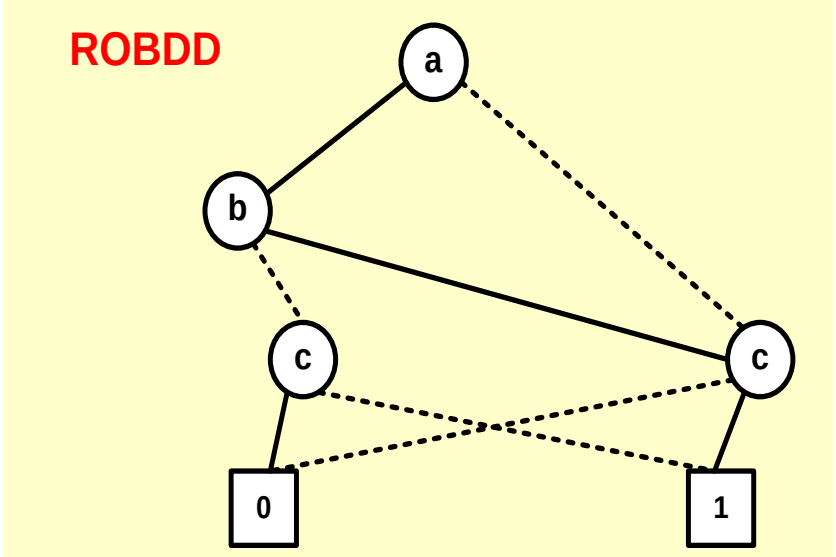
- Bináris döntési diagramot (BDD) kapunk, ha az azonos részfákat összevonjuk
- Rendezett bináris döntési diagramot (OBDD) kapunk, ha a felbontás során minden ágon azonos sorrendben vesszük fel a teszt változókat
- Redukált rendezett bináris döntési diagramot (ROBDD) kapunk, ha a szükségtelen csomópontokat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

Példa: Egy bináris döntési fa átalakítása









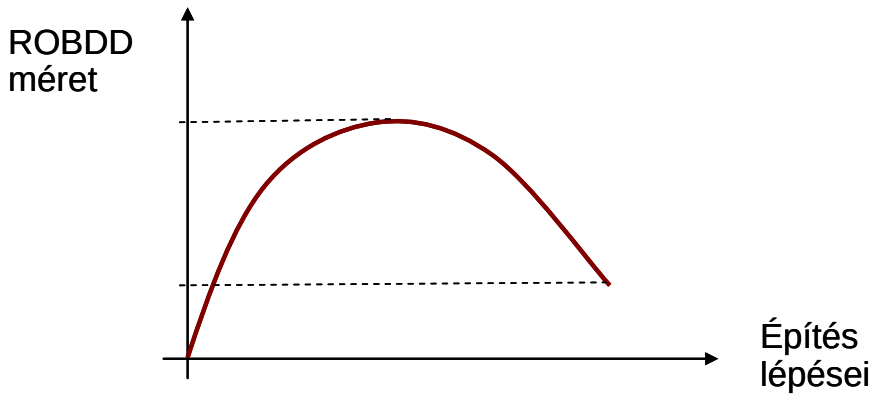
ROBDD tulajdonságok

- 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 ($\rightarrow$ heurisztika)
- Tárigény: Ha folyamatosan építjük a ROBDD-t, akkor az építés közben ideiglenes csomópontokat kell tárolnunk, amiket le lehet redukálni

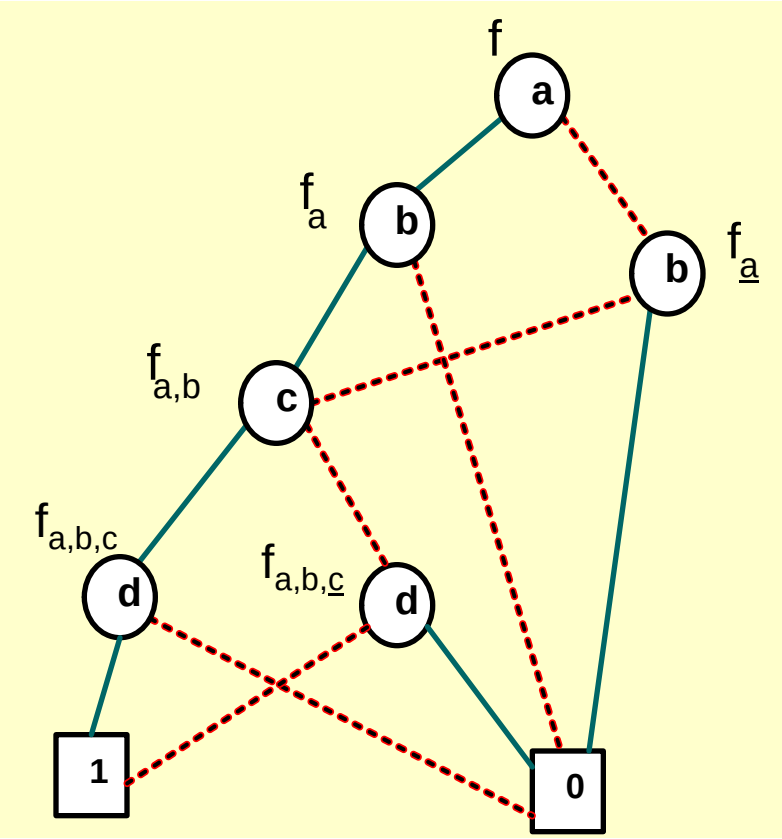


Példa: ROBDD kézi előállítása

Legyen az

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

Sorrend legyen: a, b, c, d

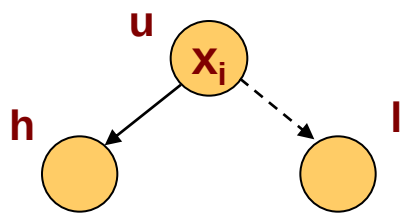


- $f = a \rightarrow f_a, f_{\underline{a}}$
  $f_a=(1\Leftrightarrow b)\wedge(c\Leftrightarrow d)$, $f_{\underline{a}}=(0\Leftrightarrow b)\wedge(c\Leftrightarrow d)$
- $f_a = b \rightarrow f_{a,b}, f_{a,\underline{b}}$
  $f_{a,b}= (1\Leftrightarrow 1)\wedge(c\Leftrightarrow d) = (c\Leftrightarrow d)$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

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

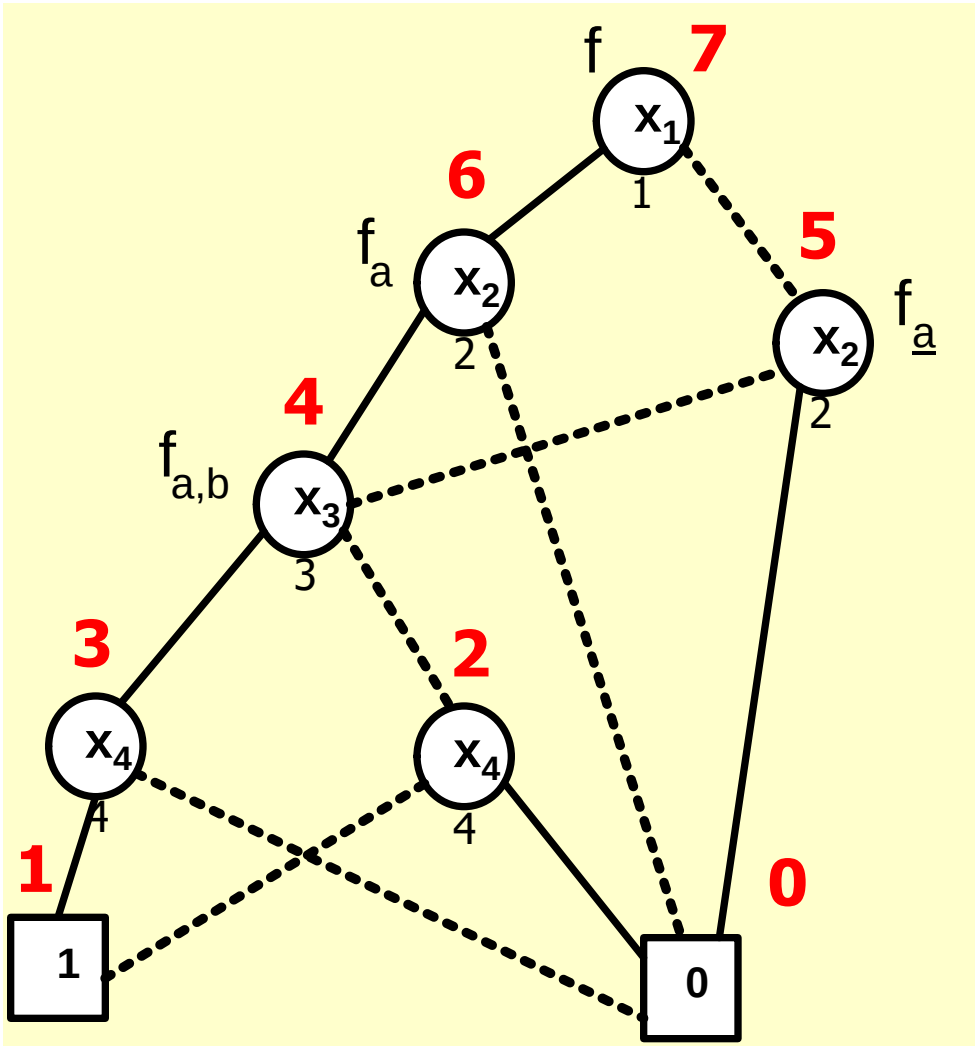
ROBDD gépi tárolása

- A ROBDD csomópontjait indexekkel azonosítjuk
- A ROBDD-t egy
  T: u → (i,l,h)
  táblázatban tároljuk:
  - u: csomópont indexe
  - i: a változó indexe (x_i, i=1…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:
  - init(T)
    - T kezdeti állapotát állítja be
    - csak a 0 és 1 csomópontok vannak a táblázatban
  - add(T,i,l,h):u
    - egy új csomópontot készít T-ben az adott paraméterekkel
    - ennek u indexét adja vissza
  - var(T,u):i
    - visszaadja T-ből az u csomópont változójának i indexét
  - low(T,u):l és 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 táblázatot is nyilvántartunk
- Műveletei:
  - init(H)
    - egy üres H táblázatot állít elő
  - member(H,i,l,h):t
    - ellenőrzi, hogy az (i,l,h) hármas szerepel-e H-ban; t Boole érték
  - lookup(H,i,l,h):u
    - kikeresi az (i,l,h) hármast a H táblázatban
    - visszaadja a hozzá tartozó u csomópont indexét
  - insert(H,i,l,h,u)
    - beilleszt egy új sort a táblázatba

ROBDD gépi előállítása 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);
}

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

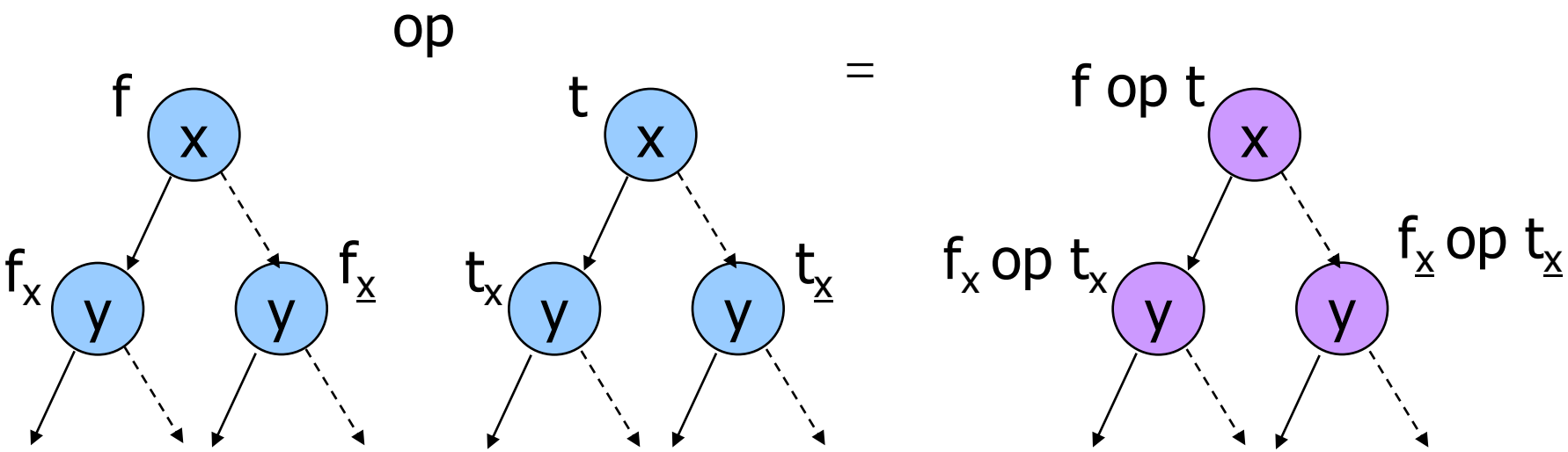
Rekurzívan végigmegy majd a változókon

Terminális csomóponthoz értünk (minden változó behelyettesítve)

Rekurzív építés; Mk() ellenőrzi az izomorf részfákat

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. $f \text{ op } t = (x \rightarrow f_x, f_{\underline{x}}) \text{ op } (x \rightarrow t_x, t_{\underline{x}}) = x \rightarrow (f_x \text{ op } t_x), (f_{\underline{x}} \text{ op } t_{\underline{x}})$



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. $f \text{ op } t = (x \rightarrow f_x, f_{\underline{x}}) \text{ op } (x \rightarrow t_x, t_{\underline{x}}) \quad = \quad x \rightarrow (f_x \text{ op } t_x), (f_{\underline{x}} \text{ op } t_{\underline{x}})$

- 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

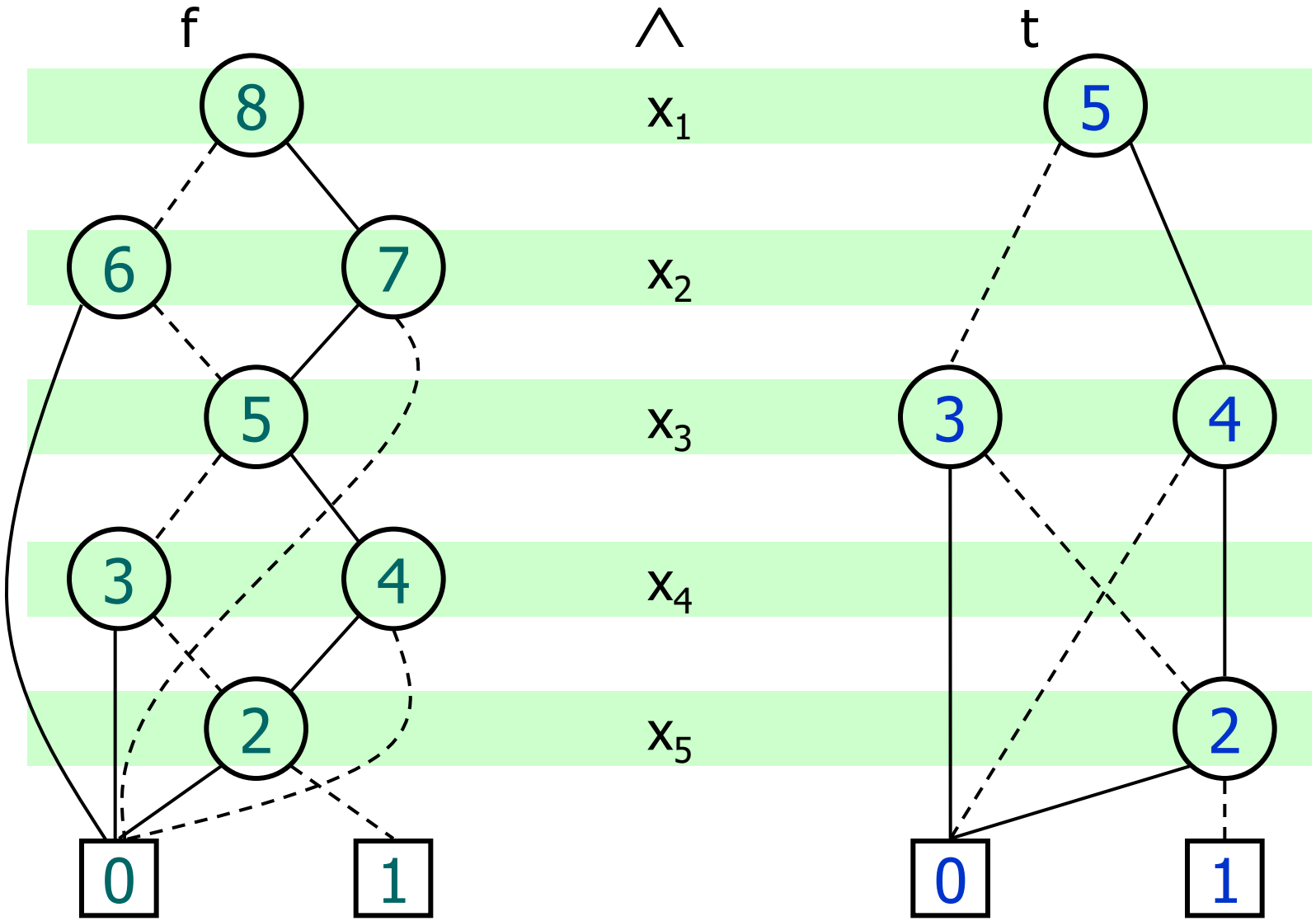
- Legyen G(op,i,j) egy gyorsítótáblázat, amely App(op,i,j) eredményét tartalmazza (csomópont)
- Az algoritmus négy esete:
  - Mindkét csomópont terminális: ekkor egy új terminális hozható létre az operátorral végzett eredmény alapján
  - Ha a csomópontokhoz tartozó változó indexe azonos, akkor a 0 és 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

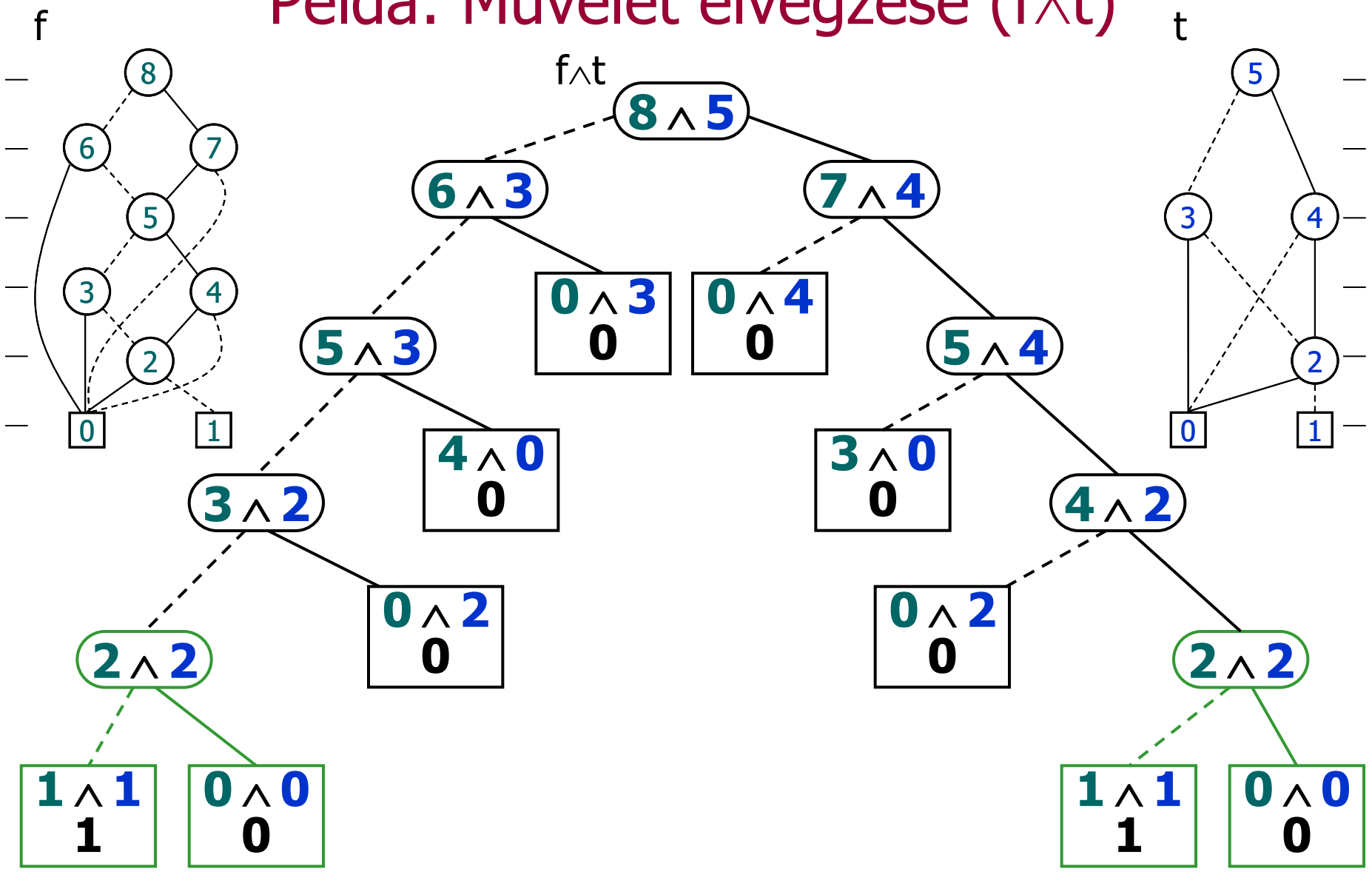
```
Apply(op,f,t){
  init(G);
  return App(op,f,t);
}

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;
}
```

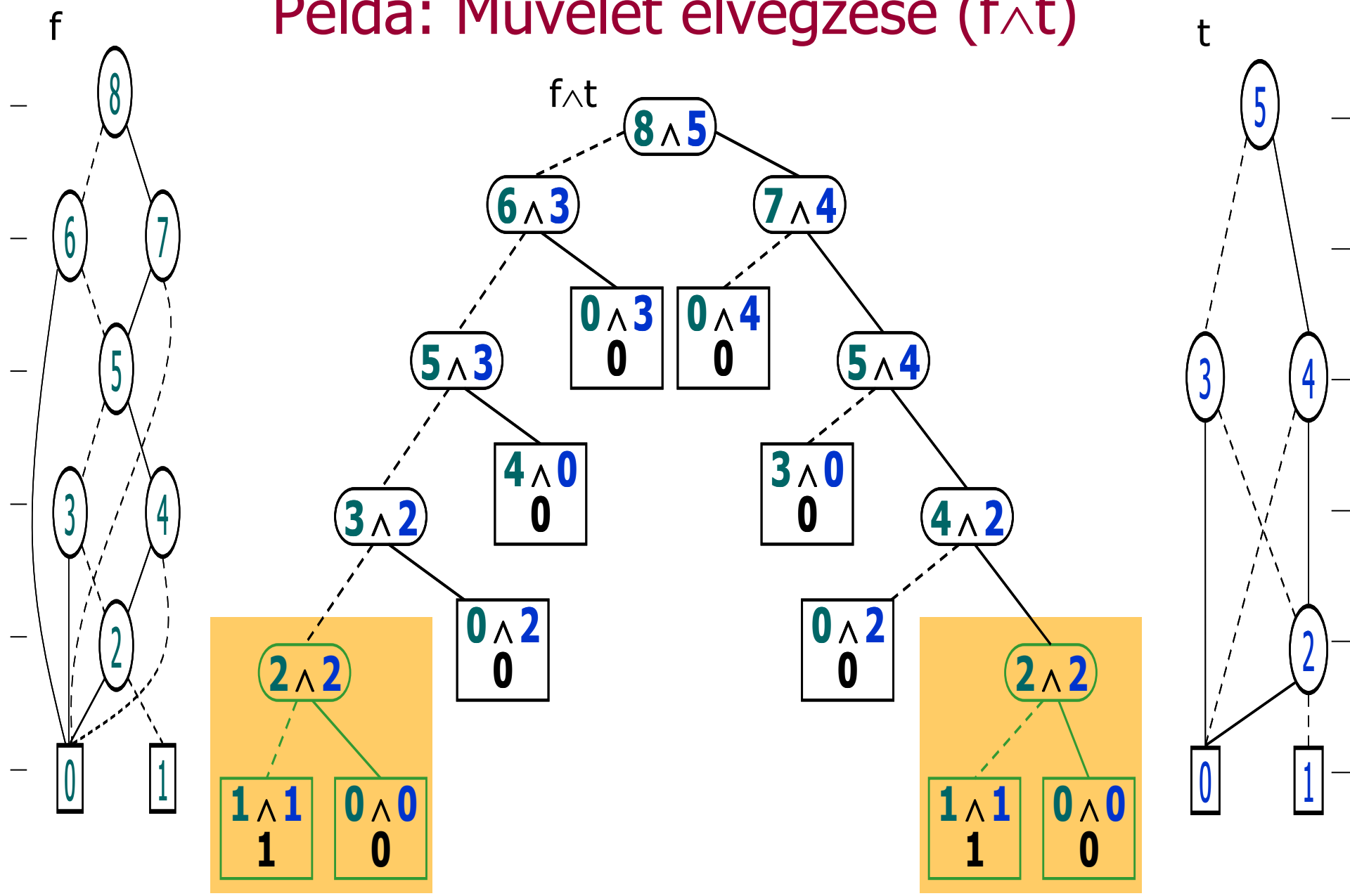
Példa: Művelet elvégzése (f∧t)



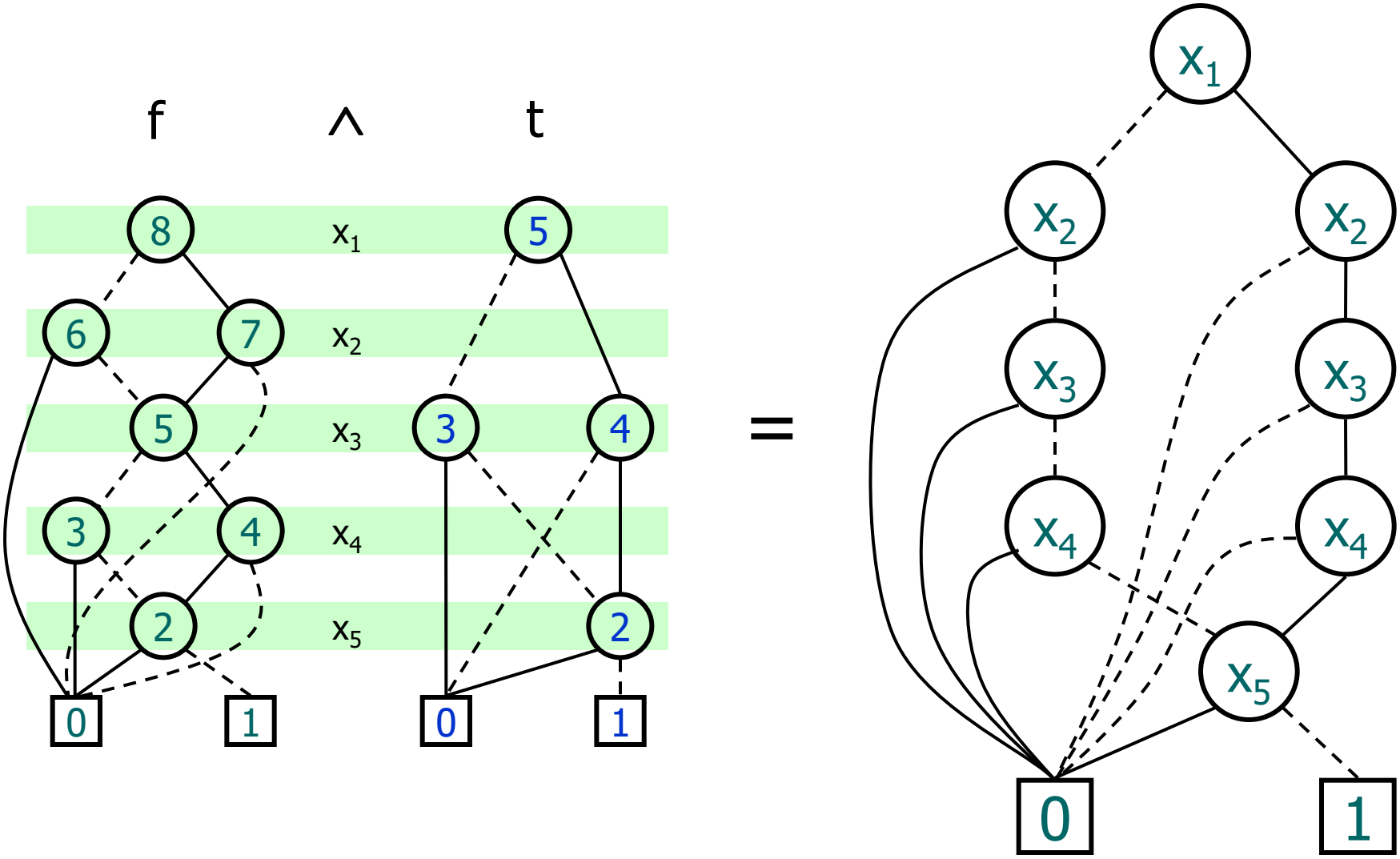
Példa: Művelet elvégzése (f∧t)



Példa: Művelet elvégzése (f∧t)



Példa: Művelet eredménye (f∧t)



Behelyettesítés ROBDD alakjának előállítása

Változók konstans behelyettesítése (ld. $pre_E(Z)$ 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

Összefoglalás: Modellellenőrzés ROBDD-vel

- A modellellenőrzés megvalósítása:
  - Modellellenőrzés algoritmusa: Műveletek állapothalmazokon (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!