Korlátos modellellenőrzés (Bounded Model Checking)

Majzik István
BME Méréstechnika és Információs Rendszerek Tanszék

Lehetőség: A SAT megoldók fejlődése

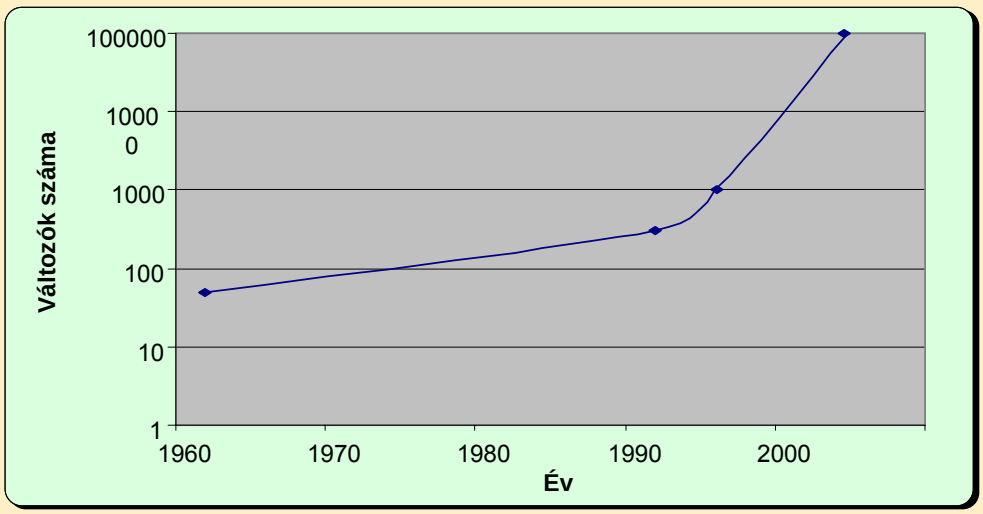
- SAT megoldó:
  - Adott logikai függvényhez olyan helyettesítési értékeket keres, amelyekkel a függvény értéke igaz
- NP-teljes probléma, de hatékony algoritmusok léteznek
  - zChaff, MiniSAT

$\omega_1 = (x_2 \vee x_3)$

$\omega_2 = (\neg x_1 \vee \neg x_4)$

$\omega_3 = (\neg x_2 \vee x_4)$

$A = \{x_1=0, x_2=1, x_3=0, x_4=1\}$

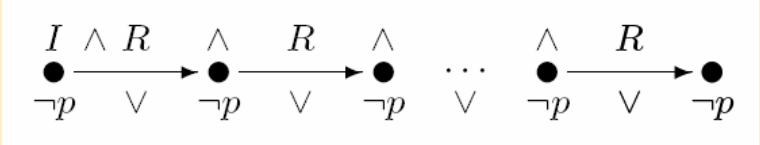


Célkitűzés

- A modellellenőrzési probléma leképzése logikai függvényre
  - Modell + temporális követelmény együttes megadása
    - Tipikusan invariáns követelmény!
- SAT megoldó használata modellellenőrzésre
  - Ha a követelmény teljesül, a SAT megoldó nem talál helyettesítési értéket a függvényhez
  - Ha a követelmény nem teljesül, a SAT megoldó által adott helyettesítési érték egy ellenpéldát jelöl ki
    - Az ellenpélda használható a hibakereséshez
    - Invariáns tulajdonságok esetén jól használható módszer

Informális bevezetés

- Modell leképzése logikai függvénybe:
  - R(s,s') állapotátmeneti reláció használata
    - A modell „széthajtogatása" az állapotátmeneti reláció mentén
    - Új állapotváltozók felvétele az egymást követő állapotokra
  - Kezdőállapotokra vonatkozó predikátum I(s)
- Kritérium (invariáns) leképzése logikai függvénybe:
  - Állapotok címkézése: p(s) állapot-predikátum (az állapotváltozókkal kifejezve)
- A SAT probléma: Konjunkció
  - Kezdőállapotra I(s) predikátum
  - Széthajtogatott átmenetek az R(s,s') reláció alkalmazásával
  - p(s) követelmény mint állapot-predikátum



Korlátos modellellenőrzés

- Az állapottér mérete kezelhetetlenül nagy lehet
  - Állapottér robbanás:
    Konkurens, lazán csatolt rendszerek
- Az útvonalak hosszát korlátozva végezzük az ellenőrzést
  - Teljes állapottér bejárás csak m útvonalhossz korlátig
  - Egyes esetekben van „átmérője” az állapottérnek: ez a leghosszabb útvonal, ami bejárható
- A m korlát becslése:
  - Intuíció a probléma méretéről
  - WCET algoritmusok használata

Jelölések

- M=(S,R,L) Kripke-struktúra, $R \subseteq S \times S$
- Logikai függvények:
  - $C_s(s)$ állapotok „kódolása” n hosszú bitvektorokkal
  - $C_R(s,s')$ állapotátmenetek 2n változós karakterisztikus függvénye
    - „Vesszős” állapotváltozók a cél állapot számára
  - L címkézés: Állapot-predikátumok, pl. P(s), Q(s)
  - I(s) a kezdőállapotok predikátuma (több is lehet)
  - Útvonal: k hosszú útvonal (k+1)n változóval

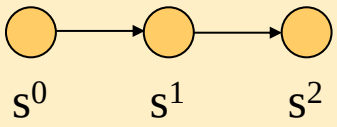
$$path(s^0, s^1, ..., s^k) = \bigwedge_{0 \le i < k} C_R(s^i, s^{i+1})$$

Vesszős változók helyett felső index

  - Adott végpontok között k hosszúságú útvonal létezése

$$path_k(s^0, s^k) = \exists_{s_1,...,s_{k-1}} path(s^0, s^1, ..., s^k)$$

Útvonalhoz tartozó logikai függvény



- Állapotok karakterisztikus függvénye:
  $C_s(x,y)$, itt x és y az állapotváltozók (bitek)
- Egy $r=(s^0,s^1)$ átmenet karakterisztikus függvénye:
  $C_r(s^0,s^1) = C_r(x^0,y^0,x^1,y^1)$
- Egy (s^0,s^1,s^2) útvonal karakterisztikus függvénye:
  $C_p(s^0, s^1, s^2) = C_p(x^0,y^0,x^1,y^1,x^2,y^2)$
- Összes 2 átmenet hosszú útvonal karakterisztikus függvénye:
  $path(x^0,y^0,x^1,y^1,x^2,y^2) = C_R(s^0,s^1) \wedge C_R(s^1,s^2)$
- Az s^0 kezdőállapotból induló 2 hosszú útvonalak :

$$I(s^0) \wedge C_R(s^0,s^1) \wedge C_R(s^1,s^2)$$

A probléma formalizálása

- Bizonyítandó P(s) invariáns: Minden útvonal, ami a kezdőállapotból indul, olyan állapotba jut, ahol P(s) igaz

$$\forall i: \ \forall s^0,s^1,...,s^i: \ (I(s^0) \wedge path(s^0,s^1,...,s^i) \Rightarrow P(s^i))$$

- Ha P(s) nem igaz valahol, akkor lesz olyan i, amire a következő függvény igaz értéket vesz fel:

$$I(s^0) \wedge path(s^0,s^1,...,s^i) \wedge \neg P(s^i)$$

- A fv. igaz értékéhez tartozó behelyettesítést adja a SAT megoldó!
  - Azaz az $(s^0,s^1,...,s^i)$ útvonalat meghatározó (i+1)n változó értéket

- Első ötlet: i=0,1,2,...-ra rendre megvizsgálni, hogy i hosszú útvonalon igaz lehet-e a következő függvény (ha igaz, akkor van ellenpélda):

$$\forall s^0,s^1,...,s^i: \ (I(s^0) \wedge path(s^0,s^1,...,s^i) \wedge \neg P(s^i))$$

Az algoritmus elemei

- Iteráció: i=0,1,2,... az útvonalak hosszára
- Ciklusmentes utakat vizsgálunk: *lfpath*

$$lfpath(s^0, s^1, ..., s^k) = path(s^0, s^1, ..., s^k) \wedge \bigwedge_{0 \le i < j \le k} s^i \neq s^j$$

- Megállási feltétel az iteráció során:
  - Nincs i hosszú ciklusmentes út kezdőállapotból, azaz nem lehet igaz

$$I(s^0) \wedge lfpath(s^0, s^1, ..., s^i)$$

  - „Rossz” állapothoz (ahol P(s) nem igaz) nem is vezethet i hosszú ciklusmentes út (kezdőállapottól függetlenül), azaz nem lehet igaz

$$lfpath(s^0, s^1, ..., s^i) \wedge \neg P(s^i)$$

- Ha megáll az iteráció, akkor P(s) mindenütt igaz

Az algoritmus

$i = 0$
while True do
  if not SAT$(I(s^0) \wedge lfpath(s^0, s^1, ..., s^i))$
    or not SAT$((lfpath(s^0, s^1, ..., s^i) \wedge \neg P(s^i))$
    then return True
  if SAT$(I(s^0) \wedge path(s^0, s^1, ..., s^i) \wedge \neg P(s^i))$
    then return $(s^0, s^1, ..., s^i)$
  $i = i + 1$
end

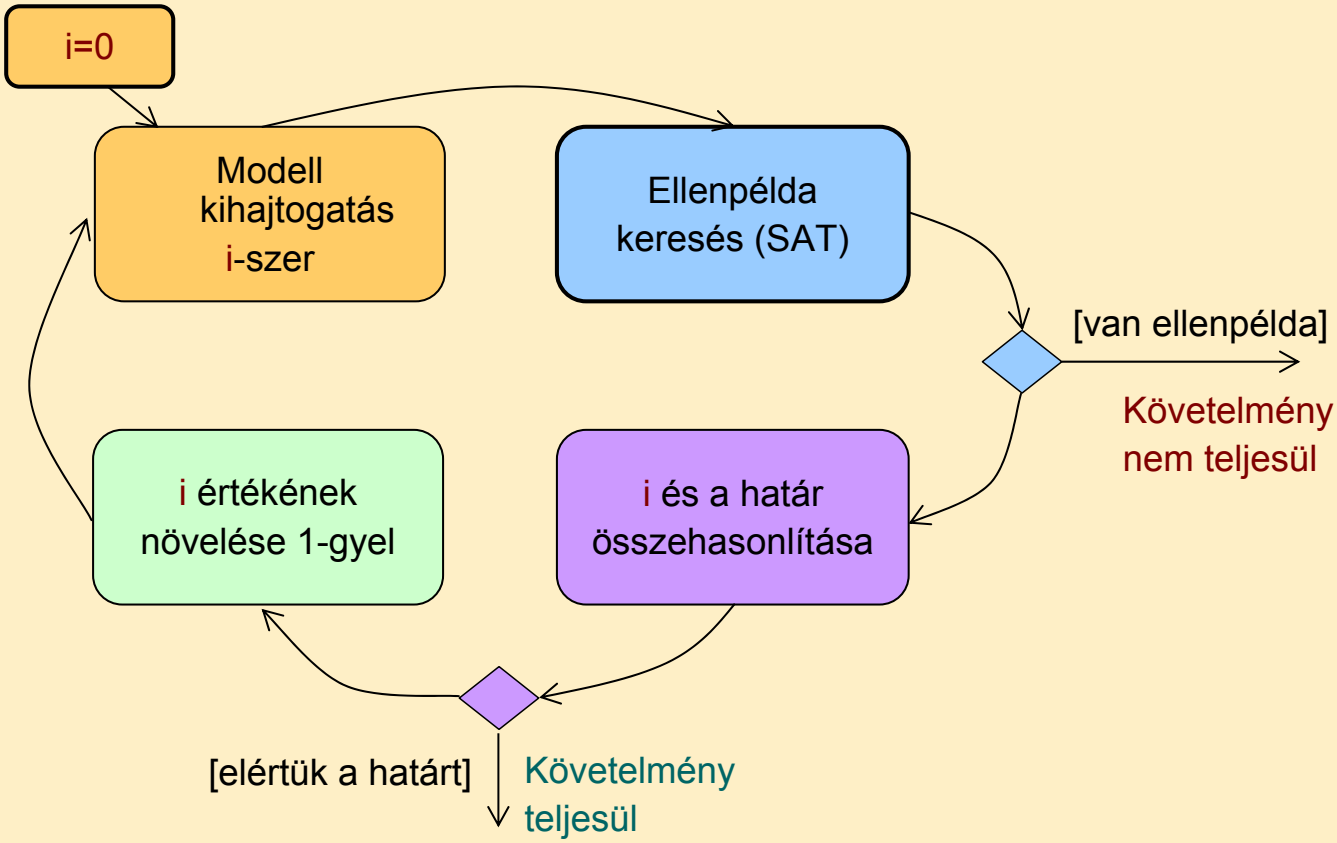
Nincs már i hosszú ciklusmentes út kezdőállapotból

Nincs i hosszú ciklusmentes út „rossz” állapotra

Van i hosszú út kezdőállapotból „rossz” állapotba

- Ha az eredmény True: Az invariáns igaz.
- Ha az eredmény egy $(s^0,s^1,...,s^i)$ útvonalat meghatározó (i+1)n bitérték: ez lesz az ellenpélda olyan állapot eléréséhez, ahol P(s) nem igaz

Eredmény: Modellellenőrzés iterációval



Az algoritmus első finomítása

- Az iterálást nem 0-ról kezdjük
  - Adott k hosszú útvonallal kezdjük, és erre először az ellenpéldát próbáljuk meg generálni:
    - Ha van ilyen ellenpélda, akkor azt gyorsan megtaláljuk (iteráció nélkül)!
  - Ezután vizsgáljuk, hogy k+1-re terminál-e az iteráció, majd növeljük az útvonal hosszát (k-indukció)
- Nem garantált, hogy az eredményül kapott ellenpélda (útvonal) minimális hosszúságú
  - Nem 0-ról kezdtük az iterációt; ha k nagy, akkor „túllövünk a célon”
  - Itt k értékére heurisztika kell, ha rövid útvonalra törekszünk
- További megkötések a SAT megoldó bemenetére:
  - Az előre haladó útvonalakon ne legyenek kezdőállapotok (ez nem ciklust jelent, hiszen akár több kezdőállapot is lehet!)
  - A visszafelé haladó útvonalakon ne legyenek olyan köztes állapotok, ahol P(s) nem igaz (ezt a korábbi iteráció ellenpéldaként adná)

Az első finomított algoritmus

$i = k$

Kezdő érték

while True do

if $\mathrm{SAT}(I(s^0) \wedge path(s^0, s^1, ..., s^i) \wedge \neg \bigwedge_{j=0}^{i} (P(s^j))$

Van i hosszú útvonal kezdőállapotból „rossz” állapoton át

then return $(s^0, s^1, ..., s^i)$

if $\mathrm{not\ SAT}(I(s^0) \wedge \bigwedge_{j=1}^{i+1} (\neg I(s^j)) \wedge lfpath(s^0, s^1, ..., s^{i+1}))$

Nincs i+1 hosszú ciklusmentes út (amiben nincs másik kezdőállapot)

or $\mathrm{not\ SAT}(lfpath(s^0, s^1, ..., s^{i+1}) \wedge \bigwedge_{j=0}^{i} P(s^j) \wedge \neg P(s^{i+1}))$

Nincs i+1 hosszú ciklusmentes út „rossz” állapotra (közben „jó” állapotokat érintve)

then return True

$i = i + 1$

end

Az algoritmus második finomítása

- Eddig az iteráció hosszát a leghosszabb ciklusmentes útvonal határozta meg az állapottérben
  - (Az állapottér hamarabb bejárható, ha) az állapotpárok között a legrövidebb útvonalat keressük: *shpath*

$$shpath(s^0, s^1, ..., s^k) = path(s^0, s^1, ..., s^k) \wedge \neg(\bigvee_{0 \le i < k} path_i(s^0, s^k))$$

  - Az algoritmus átírása: *lfpath* helyett *shpath*
- Hátrány: Sok egzisztenciális absztrakció szükséges ($path_i$)
  - Speciális algoritmusok lesznek szükségesek (kvantor eliminálás)
- Iterációk száma: A minimális érték a következők közül:
  - Előre haladó átmérő: Leghosszabb az állapotpárokat összekötő legrövidebb útvonalak közül, a kezdőállapotból indulva, nem kezdőállapotokon át
  - Visszafelé haladó átmérő: Leghosszabb az állapotpárokat összekötő legrövidebb útvonalak közül, olyan állapotból, ahol P nem igaz, olyan állapotokon át, ahol P igaz

A második finomított algoritmus

$i = k$

while True do

if $\mathrm{SAT}(I(s^0) \wedge path(s^0, s^1, ..., s^i) \wedge \neg \bigwedge_{j=0}^{i} (P(s^j))$

then return $(s^0, s^1, ..., s^i)$

if $\mathrm{not\ SAT}(I(s^0) \wedge \bigwedge_{j=1}^{i+1} (\neg I(s^j)) \wedge shpath(s^0, s^1, ..., s^{i+1}))$

or $\mathrm{not\ SAT}((shpath(s^0, s^1, ..., s^{i+1}) \wedge \bigwedge_{j=0}^{i} P(s^j) \wedge \neg P(s^{i+1}))$

then return True

$i = i + 1$

end

Van i hosszú útvonal kezdőállapotból „rossz” állapoton át

i+1-re vizsgált terminálás, shpath szerepel

- Az eredményül kapott útvonal minimális hosszúságú
- Speciális megoldó kell

Az algoritmus harmadik finomítása

- A legrövidebb utak keresése során az összes kezdőállapotot figyelembe vesszük
  - Olyan utak elkerülhetők, amelyek esetén a végállapot egy másik kezdőállapotból rövidebben elérhető
  - A kezdőállapotokat tehát *együttesen* vesszük figyelembe

$$shpath'_j(I, s^k) = \exists s^0 : (I(s^0) \wedge path_j(s^0, s^k)) \wedge$$
$$\neg \exists s^0 : (I(s^0) \wedge \bigvee_{0 \le i < j} path_i(s^0, s^k))$$

Legrövidebb út s^k-ba bármelyik kezdőállapotból, j hosszúságú

Nem igaz, hogy van olyan kezdőállapot, amiből j-nél rövidebb út vezet s^k-ba

- Hasonlóan, a kezdőállapotból a „rossz” állapotokba vezető legrövidebb utak is definiálhatók:

$$shpath'_j(s^0, \neg P)$$

A harmadik finomított algoritmus

$i = k$

while True do

if $\mathrm{SAT}(I(s^0) \wedge path(s^0, s^1, ..., s^i) \wedge \neg \bigwedge_{j=0}^{i} (P(s^j))$

then return $(s^0, s^1, ..., s^i)$

if not $\mathrm{SAT}(shpath'_{i+1}(I, s^{i+1}))$

or not $\mathrm{SAT}(shpath'_{i+1}(s^0, \neg P))$

then return True

$i = i + 1$

end

- A fixpont iterációs algoritmushoz közelít...

Összefoglalás

- Első algoritmus: Egyre hosszabb utak vizsgálata
  - SAT megoldóval ellenpélda keresés:
    - Előrefelé haladó ciklusmentes utak keresése (kezdőállapotból)
    - 0,1,2,... hosszú utak
- Nem 0-ról kezdett iteráció
  - Legrövidebb ellenpélda megtalálása nem garantált
- Legrövidebb utak figyelembe vétele
  - Nagyobb kihívás a SAT megoldónak (kvantor elimináció)
- Állapothalmazok figyelembe vétele
  - Kezdőállapotok, „rossz” állapotok együttesen kezelve

Összefoglalás: A BMC használata

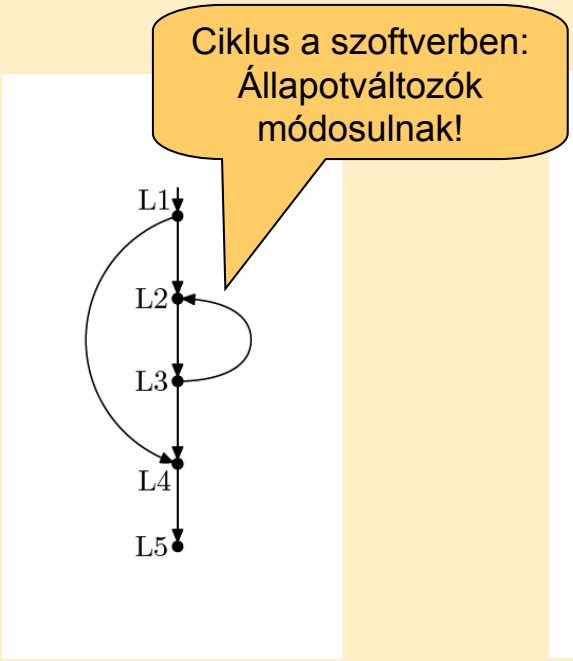
- Invariánsok vizsgálatára hatékony
  - Nem az általános modell ellenőrzési feladat megoldása!
- Helyes és teljes módszer az adott korlát mellett
  - Ha van ellenpélda, azt megtalálja (egyébként az invariáns igaz)
  - Ha ellenpéldát talál, akkor az valódi ellenpélda
  - A ciklusmentes utak használata bonyolultabbá tesz!
- Állapottér robbanásának „elkerülése”
  - Adott számú iteráció vizsgálatával részleges eredmény kapható
- Legrövidebb ellenpélda keresése
  - Tesztgeneráláshoz használható
- Automatikus módszer
  - A korlát kijelölése lehet heurisztikus (az állapottér „átmérője”)
- Eszközök:
  - SAL: sal-smc, sal-bmc, sal-atg

Az Intel eredményei (hardver verifikáció)

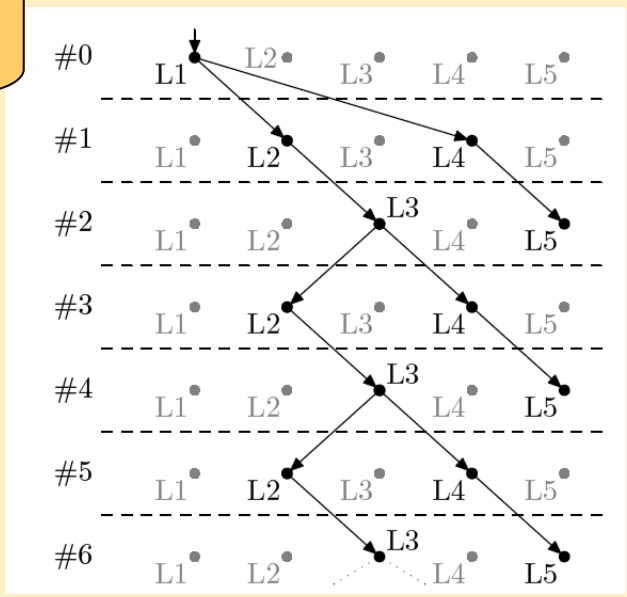
Model	k	Forecast (BDD)	Thunder (SAT)
Circuit 1	5	114	2.4
Circuit 2	7	2	0.8
Circuit 3	7	106	2
Circuit 4	11	6189	1.9
Circuit 5	11	4196	10
Circuit 6	10	2354	5.5
Circuit 7	20	2795	236
Circuit 8	28	—	45.6
Circuit 9	28	—	39.9
Circuit 10	8	2487	5
Circuit 11	8	2940	5
Circuit 12	10	5524	378
Circuit 13	37	—	195.1
Circuit 14	41	—	—
Circuit 15	12	—	1070

Alkalmazás szoftverek esetén: A ciklusok problémája

A ciklusok bejárása új állapotokat eredményez!



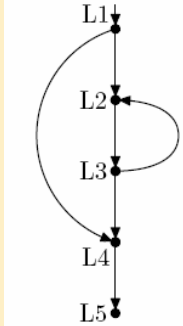
Vezérlési gráf (CFG) példa

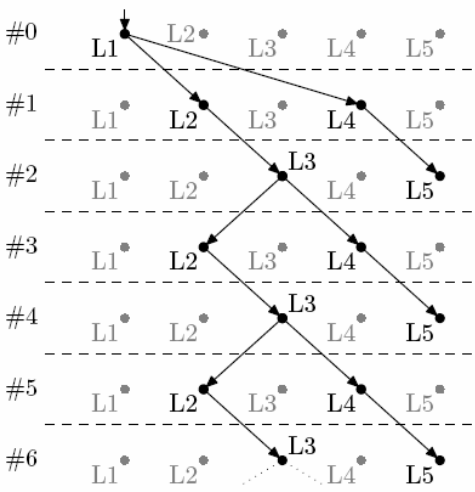


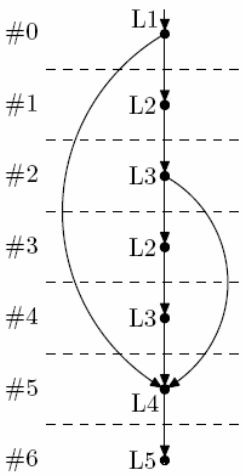
Széthajtogatás

Korlátozott ciklusbejárás

- Modell széthajtogatás optimalizálása:
  - Alapeset: Teljes széthajtogatás (path enumeration)
    - Mindig szisztematikusan előre
  - Korlátozott ciklusbejárás (loop unrolling)
    - Ciklusokra egyenként lefutási korlátot adni és úgy kibontani







Eszközök

- F-SOFT (NEC):
  - Hagyományos teljes széthajtogatás
  - Unix rendszerprogramokra alkalmazták (pl. pppd)
- CBMC (CMU, Oxford):
  - C, SystemC támogatása
  - Korlátozott ciklusbejárás (loop unrolling)
  - Egyes Linux, Windows, MacOS rendszerkönyvtárak támogatása
  - Integer aritmetikai műveletek leképzése
    - Bit-flattening (bit-blasting): „áramköri megfelelő” bitvektorokra
  - CBMC SMT megoldóval
    - Satisfiability Modulo Theories: A SAT megoldó kiterjesztése különböző domének kezelésére (pl. integer aritmetika)
- SATURN:
  - Korlátozott ciklusbejárás: max. 2 lefutás
  - Teljes Linux kernel ellenőrizhető: Null pointer hivatkozásokra