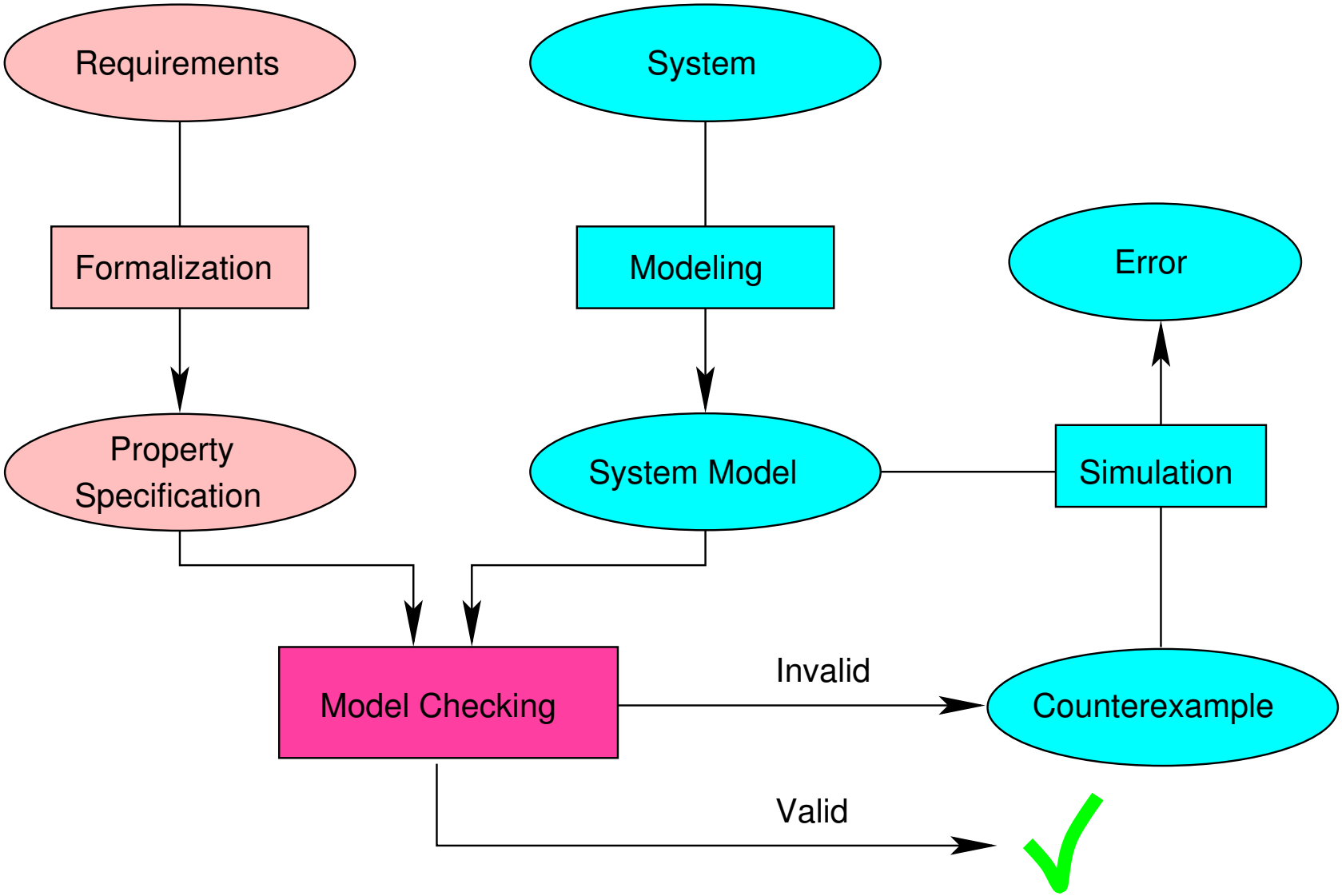
IV113 Validace a verifikace

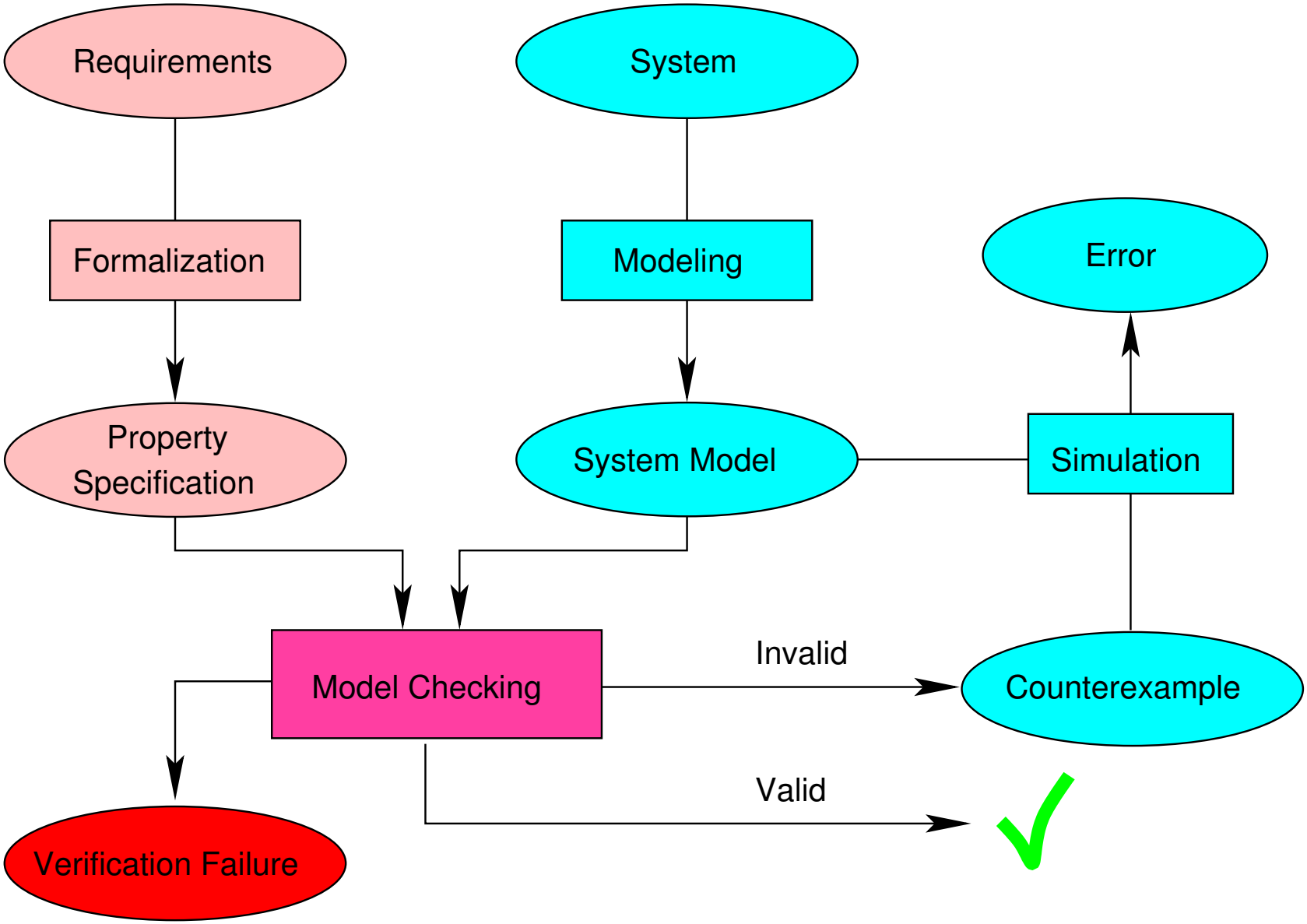
Symbolický přístup k metodě ověřování modelu

Jiří Barnat

Problém stavové exploze při ověřování modelu



Problém stavové exploze při ověřování modelu



Motivace

Pozorování

- Stav modelovaného systému je určen valuací proměnných.
- Každá proměnná má konečnou doménu, její hodnotu lze uložit s využitím pevně stanoveného počtu bitů.
- Stav je v paměti počítače reprezentován jako bitový vektor $(a_1, \ldots, a_n)$ fixní délky n.

Množiny stavů

- Algoritmy pro verifikaci uchovávají množiny stavů.
- Množinu stavů lze chápat jako množinu binárních vektorů.
- Množinu binárních vektorů lze popsat **boolovskou funkcí**.

Boolovské funkce

- Jsou formule výrokové logiky, nad konečnou množinou proměnných typu Bool.

Příklad

- Stav systému je dán valuací 4 proměnných (a1,b1,a2,b2) každé do 2 hodnotové domény $\{0,1\}$.
- Stav systému je chybový, pokud se shodují proměnné a1 a b1 a proměnné a2 a b2.
- Popište množinu chybových stavů boolovskou funkcí.

Některá možná řešení

Boolovské funkce

- Jsou formule výrokové logiky, nad konečnou množinou proměnných typu Bool.

Příklad

- Stav systému je dán valuací 4 proměnných (a1,b1,a2,b2) každé do 2 hodnotové domény $\{0,1\}$.
- Stav systému je chybový, pokud se shodují proměnné a1 a b1 a proměnné a2 a b2.
- Popište množinu chybových stavů boolovskou funkcí.

Některá možná řešení

- $(a1 \wedge b1 \wedge a2 \wedge b2) \vee (a1 \wedge b1 \wedge \neg a2 \wedge \neg b2) \vee$ $(\neg a1 \wedge \neg b1 \wedge \neg a2 \wedge \neg b2) \vee (\neg a1 \wedge \neg b1 \wedge a2 \wedge b2)$
- $a1 \Leftrightarrow b1 \wedge a2 \Leftrightarrow b2$

Reprezentace boolovských funkcí

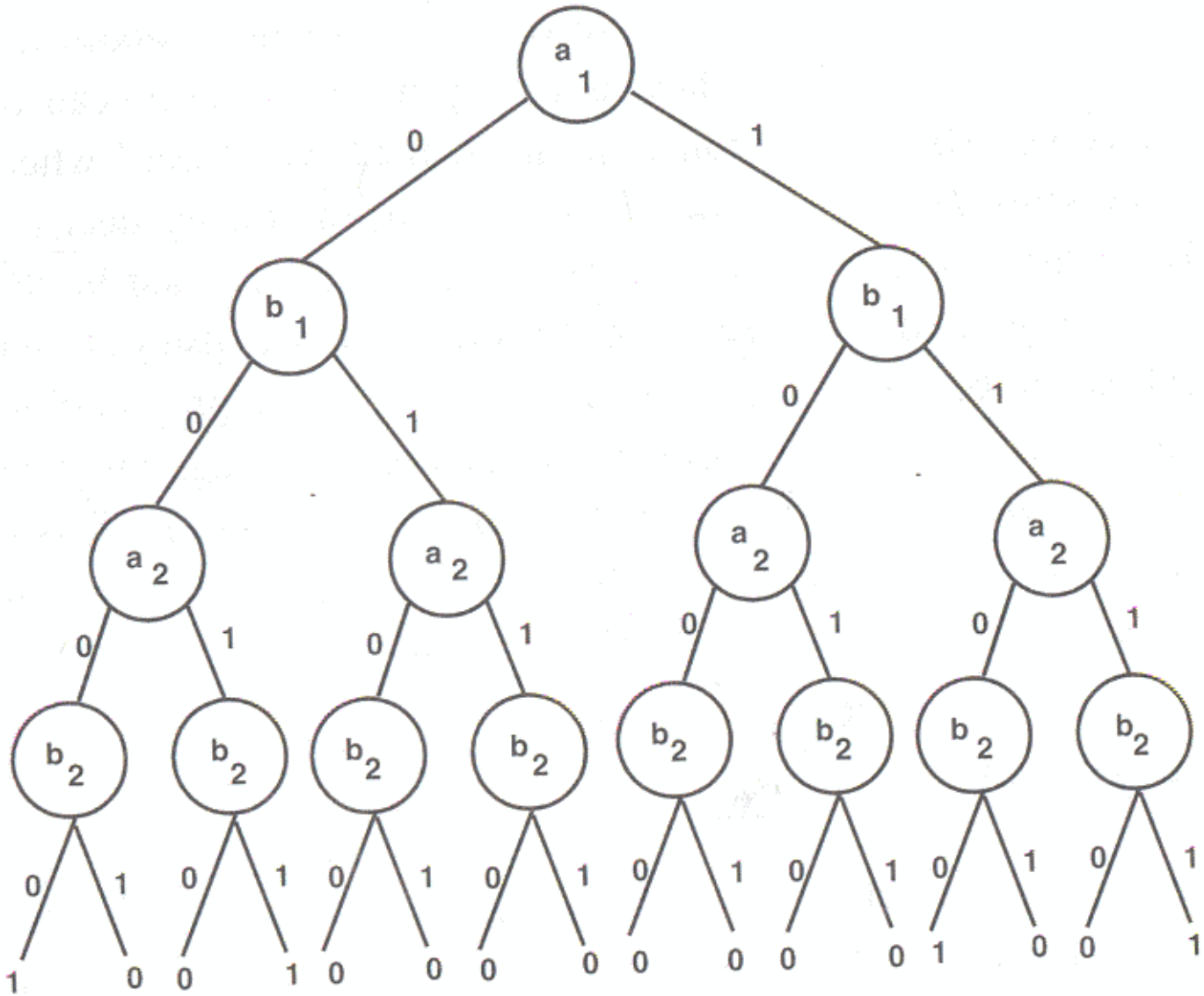
Binární rozhodovací stromy (BDTs)

- Orientovaný strom s právě jedním kořenem.
- Každý vnitřní vrchol je označen binární proměnnou (v) a má právě 2 následníky ($low(v)$, $high(v)$).
- Každý list je označen binární hodnotou (0,1).

Kódování boolovské funkce pomocí BDT

- Každá kombinace hodnot vstupních proměnných odpovídá právě jedné cestě z kořene stromu do listu.
- Hodnoty uložené v jednotlivých listech udávají hodnoty funkce pro jednotlivé vstupy.

Binární rozhodovací strom $\psi = (a_1 \Leftrightarrow b_1) \wedge (a_2 \Leftrightarrow b_2)$



Reprezentace boolovských funkcí

Nevýhoda BDTs

- BDTs jsou zbytečně prostorově náročné (obsahují redundantní informace).

Příklad

- Identifikujte isomorfní podstromy BDT z předchozího příkladu.

Binární rozhodovací diagram (BDD)

- Acyklický graf, jehož vrcholy mají výstupní stupeň vždy buď 0 (listy) nebo 2 (vnitřní vrcholy).
- Vrcholy BDD mají stejné atributy jako vrcholy BDT.

Vytvoření (minimálního) BDD

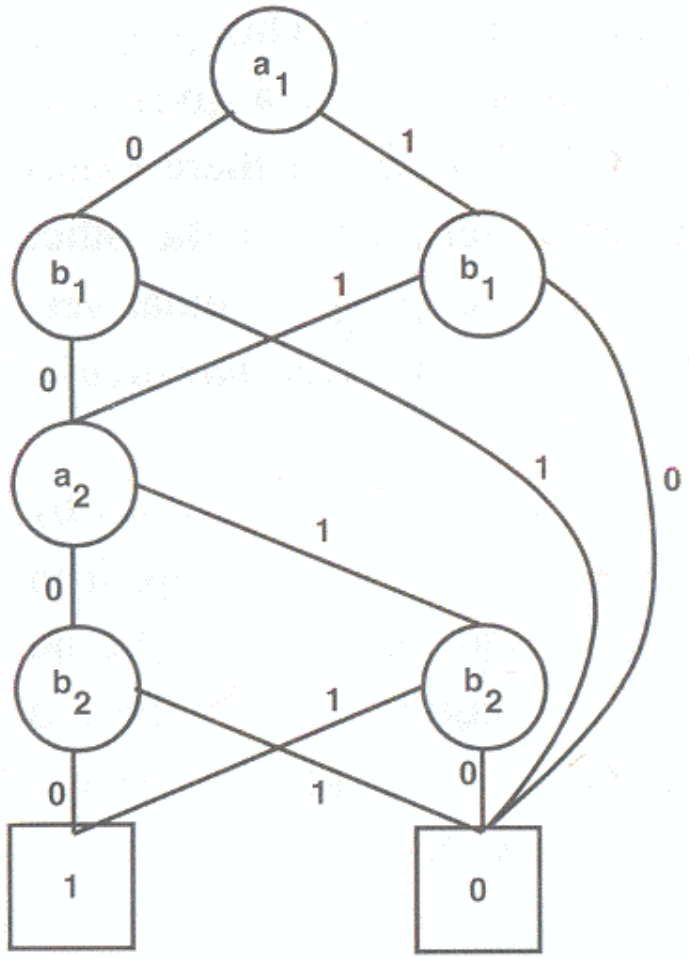
Inicializace

- Pro danou boolovskou funkci uvaž nějaké BDD (BDT).
- Eliminace nedosažitelných vrcholů
  - Odstraň vrcholy nedosažitelné z kořene BDD.
- Eliminace duplikátních listů
  1) Odstraň až na jeden stejně označené listy BDT.
  2) Všechny hrany eliminovaných stejně označených listů přepoj do zbývajícího stejně označeného listu.

Opakovaně aplikuj následující transformace

- Eliminace duplikátních (vnitřních) vrcholů
  - Pokud v BDD existují stejně označené vnitřní vrcholy u, v takové, že $low(v) = low(u)$ a $high(v) = high(u)$, tak odstraň vrchol u a hrany původně vedoucí do vrcholu u přepoj do vrcholu v.
- Eliminace zbytečných testů
  - Eliminuj vnitřní vrchol v, pokud $low(v) = high(v)$ a hrany původně vedoucí do v přepoj do vrcholu $low(v)$.

BDD pro $\psi = (a_1 \Leftrightarrow b_1) \wedge (a_2 \Leftrightarrow b_2)$



Kódování boolovské funkce pomocí BDD

Tvrzení

- Každý vrchol v binárního rozhodovacího diagramu kóduje nějakou boolovskou funkci $F_v(x_1, \ldots, x_n)$.

Výpočet $F_v(x_1, \ldots, x_n)$ pro konkrétní hodnoty $h_1, \ldots, h_n$.

- Je-li vrchol v list, pak
  - $F_v(h_1, \ldots, h_n) = 1$, pokud je list v označen hodnotou 1.
  - $F_v(h_1, \ldots, h_n) = 0$, pokud je list v označen hodnotou 0.
- Je-li vrchol v vnitřní vrchol označený proměnou x_i, pak
  - $F_v(h_1, \ldots, h_n) = F_{low(v)}(h_1, \ldots, h_n)$, pokud $h_i == 0$.
  - $F_v(h_1, \ldots, h_n) = F_{high(v)}(h_1, \ldots, h_n)$, pokud $h_i == 1$.

Uspořádání proměnných v BDD — OBDD

Pozorování

- Každý mezivýsledek výpočtu minimálního BDD je BDD.
- Výpočet minimálního BDD lze začít v libovolném BDD.
- Pro danou boolovskou funkci existují různá BDD.

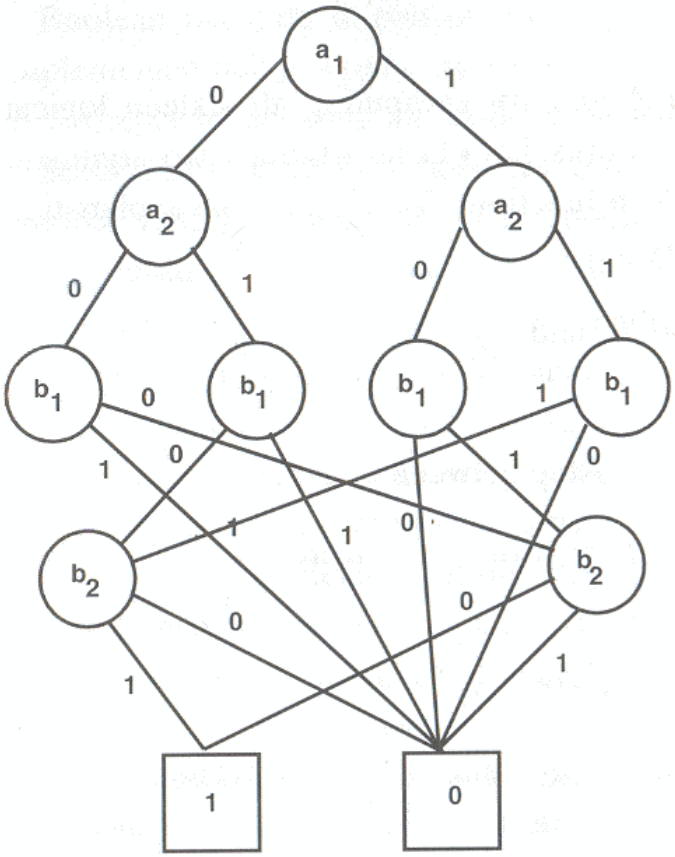
Kanonická forma BDD

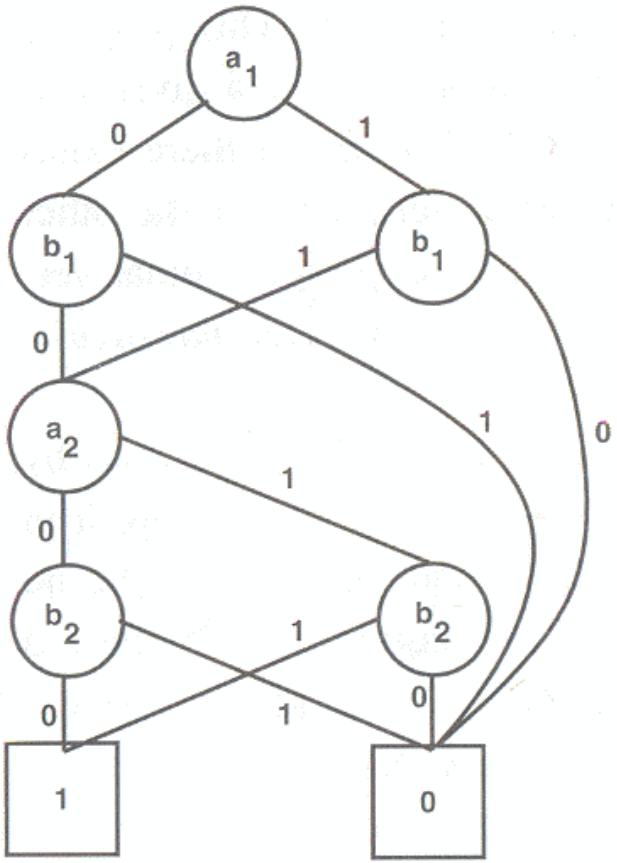
- Minimální BDD vzniklé z BDT s předem daným pořadím proměnných je určeno jednoznačně.
- BDD s fixovaným pořadím proměnných se označují jako Ordered BDD (OBDD).

Kanonizace

- Kanonizaci BDD, které respektuje pořadí proměnných lze provést v čase $\mathcal{O}(n)$, kde n je velikost původního BDD.
- [Transformace se aplikují nejprve na vrcholy, které jsou označeny větší proměnnou.]

Různá OBDD dle různého uspořádání proměnných





Operace restrikce na OBDD

Pozorování

- Každé OBDD reprezentuje nějakou boolovskou funkci.
- Boolovské funkce lze skládat pomocí unárních a binárních logických operátorů ($\neg, \wedge, \vee, \implies, XOR, \ldots$).
- Skládání lze přenést na OBDD.

Aplikace operací na OBDD – funkce *Apply*

- Máme OBDD O a O', která odpovídají funkcím f a f'.
- Chceme proceduru $Apply(O, O', \star)$, která vypočítá OBDD odpovídající složení funkcí f a f' logickým operátorem $\star$.

Operace restrikce

- $F_{x_i \leftarrow b}(x_1, \ldots, x_n) = F(x_1, \ldots, x_{i-1}, b, x_{i+1}, \ldots, x_n)$
- Vytvoří boolovskou funkci s menším počtem proměnných.

Realizace na OBDD

- Pokud je kořen r označen proměnnou x_i, novým kořenem se stává vrchol
  - $low(r)$ pokud $b = 0$
  - $high(r)$ pokud $b = 1$
- Pokud vrchol v má hranu vedoucí do vrcholu t označeným proměnou x_i, tak se tato hrana přepojí do:
  - $low(t)$ pokud $b = 0$
  - $high(t)$ pokud $b = 1$
- OBDD se minimalizuje (vrcholy x_i jsou nedosažitelné).

Shannonova expanze

- Pro aplikaci libovolného binární logického operátoru lze využít tzv. Shannonovu expanzi:

$$F = (\neg x \wedge F_{x \leftarrow 0}) \vee (x \wedge F_{x \leftarrow 1})$$

- Pokud $F = f \star f'$, kde $\star$ je binární logická operace, pak

$$f \star f' = (\neg x \wedge (f_{x \leftarrow 0} \star f'_{x \leftarrow 0})) \vee (x \wedge (f_{x \leftarrow 1} \star f'_{x \leftarrow 1}))$$

Ostatní operace na OBDD

$Apply(O, O', \star)$

- Nechť v, v' jsou kořenové uzly O, O', označené proměnnými x, x'.
- Pokud v a v' jsou listy označené hodnotami h a h', pak vrať list označený hodnotou $h \star h'$.
- Jinak, pokud

$x = x'$ pak vrať nový vrchol w označený proměnnou x, kde

- $low(w) = Apply(low(v), low(v'), \star)$
- $high(w) = Apply(high(v), high(v'), \star)$

$x < x'$ pak vrať nový vrchol w označený proměnnou x, kde

- $low(w) = Apply(low(v), O', \star)$
- $high(w) = Apply(high(v), O', \star)$

$x' < x$ pak vrať nový vrchol w označený proměnnou x', kde

- $low(w) = Apply(O, low(v), \star)$
- $high(w) = Apply(O, high(v), \star)$

Operace negace a test na prázdnost

Pozorování

- Pokud *OBDD X* realizuje funkci F_X, tak *OBDD Y* realizující funkci $\neg F_X$ se vytvoří kopií *OBDD X* a přeznačením listů kopie duální hodnotou.

Test na prázdnost

- OBDD mají kanonický tvar.
- Kanonické OBDD reprezentující prázdnou množinu je vždy tvořeno pouze listem označeným hodnotou 0.

Test na přítomnost stavu v množině

- Vytvořím OBDD realizující jednoprvkovou množinu s dotazovaným stavem.
- Aplikuji operaci $\wedge$ na dané OBDD.
- Provedu test na prázdnost.

Symbolická reprezentace Kripkeho struktury

Kódování přechodové funkce KS pomocí OBDD

Pozorování

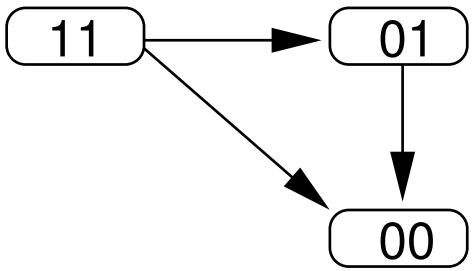
- Stav Kripkeho struktury $M = (S, T, I)$ je popsán n binárními proměnnými $a_1, \ldots, a_n$.
- Každou podmnožinu stavů Kripkeho struktury je možné zachytit pomocí OBDD s n proměnnými.
- Podobně přechodovou relaci $T \subseteq S \times S$ je možné kódovat boolovskou funkcí s $2n$ proměnnými.

Zjednodušení reprezentace OBDD

- Hrany vedoucí do listu 0 je možné vynechat.
- Neexistence hrany indikuje přechod do listu 0.

Příklad

- $M = (\{00, 01, 11\}\}, \{(11, 00), (11, 01), (01, 00)\}, I)$



- T lze popsat pomocí $F(a, b, a', b')$
- $F(a, b, a', b') =$
  $(a \wedge b \wedge \neg a' \wedge b') \vee (a \wedge b \wedge \neg a' \wedge \neg b') \vee (\neg a \wedge b \wedge \neg a' \wedge \neg b')$
- Zakreslete OBDD pro F je-li dáno $a < b < a' < b'$.

Následníci množiny stavů

Pozorování

- Mějme $M = (S, T, I)$ a $OBDD_T(a, b, a', b')$.
- Mějme množinu stavů X zadanou pomocí $OBDD_X(a, b)$.
- Z $OBDD_T$ a $OBDD_X$ lze vytvořit $OBDD_{X'}(a', b')$, které reprezentuje množinu následníků množiny X, tj.

$$X' = \{v \in S \mid u \in X \wedge (u, v) \in T\}.$$

Následníci množiny stavů

Výpočet $OBDD_{X'}$ (intuitivně)

- $OBDD_{X'} = Apply(OBDD_T, OBDD_X, \wedge)$
- Modifikuj $OBDD'_X$ tak, aby na každé cestě byl vrchol a'.
- V $OBDD_{X'}$ smaž všechny vrcholy a a b.
- Postupně vol všechny vrcholy označené a' jako kořeny a vypočti k nim minimální OBDD.
- Množinu vypočtených OBDD spoj pomocí operace $\vee$.
- Výsledné OBDD minimalizuj.
- Přejmenuj čárkované proměnné na nečárkované.

Příklad

- Spočítejte reprezentaci následníků množiny $\{00, 11\}$.

Předchůdci množiny stavů

Podobně lze počítat předchůdce dané množiny (intuitivně).

- Přeznač proměnné v $OBDD_X$ na čárkované.
- $OBDD_{X'} = Apply(OBDD_T, OBDD_X, \wedge)$
- Modifikuj $OBDD'_X$ tak, aby na každé cestě byl vrchol a'.
- Vrcholy a', ze kterých neexistuje cesta do listu ohodnoceným 1 nahraď novým listem ohodnoceným 0.
- Ostatní vrcholy a' nahraď listem ohodnoceným 1.
- Smaž staré listy a ostatní vrcholy označené čárkovanou proměnnou a minimalizuj OBDD.

Příklad

- Spočítejte OBDD reprezentující množinu stavů $\{00\}$.
- Spočítejte OBDD reprezentující předchůdce dané množiny.

Symbolický přístup k verifikaci CTL

Připomenutí

Pozorování

- Znám-li pro každý stav platnost formulí φ a ψ, snadno odvodím platnost formulí $\neg\varphi$, $\varphi \vee \psi$, $EX\,\varphi$, ...

Idea algoritmu pro CTL Model Checking

- Je dána Kripkeho struktura $M = (S, T, I)$ a formule φ.
- Spočítám značkovací funkci $label : S \to 2^{\varphi}$, která o každém stavu $s \in S$ Kripkeho struktury M řekne jaké podformule formule φ platí v daném stavu.
- Platí, že $s_0 \models \varphi \iff \varphi \in label(s_0)$.
- Funkci *label* budu počítat postupně pro jednotlivé podformule formule φ, a to od nejjednodušších podformulí (atomické propozice) ke složitějším (až po podformuli φ).

Symbolický přístup

Myšlenka

- Budu si kompaktně pamatovat/budovat množiny stavů, ve kterých platí jednotlivé podformule verifikované CTL formule.
- Explicitní počítání funkce *label* nahradím manipulací s těmito kompaktními reprezentacemi.

Realizace

- Množiny stavů udržovány pomocí OBDD struktur.
- Výchozím bodem jsou OBDD pro jednotlivé AP.
- Podle struktury formule počítám OBDD pro jednotlivé podformule.
- Otestuji přítomnost iniciálního stavu v množině stavů splňující verifikovanou formuli.

Atomické propozice a logické operátory

Připomenutí syntax CTL

- $\varphi ::= p \mid \neg\varphi \mid \varphi \vee \varphi \mid EX\,\varphi \mid E[\varphi\, U\, \varphi] \mid EG\,\varphi$

Výpočet množiny stavů splňující CTL formuli

- Značení
  - $F(\psi)$ označme (funkci popisující) množinu stavů splňující ψ.
  - $Succ(X)$ označme všechny následníky množiny stavů X.
  - $Pred(X)$ označme všechny předchůdce množiny stavů X.
- Boolovské funkce pro atomické propozice
  - Atomická propozice se vyjadřuje o platnosti proměnných.
  - Atomická propozice je boolovská funkce.
- Výpočet logických operátorů ($\neg$, $\vee$)
  - $F(\neg\psi_1) = \neg(F\psi_1)$
  - $F(\varphi \vee \psi) = F(\varphi) \vee F(\psi)$

Temporální operátory $EX(\varphi)$, $E[\varphi\, U\, \psi]$ a $EG(\varphi)$

- Množina splňující $EX(\varphi)$
  - $F(EX(\varphi)) = Pred(F(\varphi))$
- Množina splňující $E(\varphi\, U\, \psi)$
  - $F(E(\varphi\, U\, \psi)) = X$,
    kde X je nejmenší pevný bod rekurzivního předpisu

$$X = F(\psi) \cup (F(\varphi) \cap EX(X))$$

- Množina splňující $EG\,(\varphi)$
  - $F(EG\,\varphi) = X$,
    kde X je největší pevný bod rekurzivního předpisu

$$X = F(\varphi) \cap EX(X)$$

Výpočet pevných bodů

Nejmenší pevný bod $f(x)$

proc $LFP(f)$
  $X = \emptyset$
  $Xold = \emptyset$
  do
    $Xold = X$
    $X := f(X)$
  while $(X \neq Xold)$
end

Největší pevný bod $f(x)$

proc $GFP(f)$
  $X = S$
  $Xold = S$
  do
    $Xold = X$
    $X := f(X)$
  while $(X \neq Xold)$
end

Ověřování modelu – shrnutí

Ověřování modelu – shrnutí

Enumerativní × symbolický přístup

- Enumerativní – orientován na "control-flow"
- Symbolický – orientován na "data-flow"

Výhody oproti testování

- Není třeba zdrojový kód (aplikovatelné ve fázi návrhu).
- Aplikovatelné na paralelní programy.

Výhody oproti statickým metodám

- Metoda je úplná, tj. vždy dává přesné výsledky.
- Možno verifikovat temporální vlastnosti.

Nevýhoda

- Problém stavové exploze.