ZÁKLADY MATEMATICKEJ TEÓRIE PROGRAMOV

Časť 1.3: Porovnávanie tried programových schém

Igor Prívara
Inštitút informatiky a štatistiky

FMFI UK Bratislava, Katedra informatiky

Základy teórie programovania

Programové schémy – abstrakcia programov

- základné pojmy, vlastnosti programových schém
- rozhodnuteľnosť vlastností programových schém
- **porovnávanie tried programových schém**

Správnosť programov – vzhľadom na špecifikácie

- Floydova metóda a Hoareova metóda
- indukčné techniky použité pri dokazovaní správnosti
- (systematická) konštrukcia správnych programov
- dokazovanie správnosti rekurzívnych programov

Sémantika programov – formálny význam programov

- princípy operačnej, denotačnej a axiomatickej sémantiky
- algebraická štruktúra sémantických domén
- formálna definícia denotačného a operačného významu imperatívnych a rekurzívnych programov
- porovnanie denotačného a operačného významu programov

Typy a sémantika – využitie typov pri definícii sémantiky

Príklad:

schémy s rôznymi riadiacimi konštrukciami

S: **begin** $[y_1, y_2] := [x, a]$

1: **if** $p(y_1)$ **then goto end**
2: $[y_1, y_2] := [f(y_1), g(y_1, y_2)]$
3: **goto** 1

end $[z] := [y_2]$

W: **begin** $[y_1, y_2] := [x, a]$

while $\tilde{p}(y_1)$
do $[y_1, y_2] := [f(y_1), g(y_1, y_2)]$ **od**

end $[z] := [y_2]$

R: **begin** $[y_1, y_2] := [x, a]$

$\phi(y_1, y_2) \Longleftarrow$ **if** $p(y_1)$ **then** y_2
else $\phi(f(y_1), g(y_1, y_2))$

end $[z] := [\phi(x, a)]$

Porovnávanie tried programových schém

$\mathcal{S}_1, \mathcal{S}_2$ – triedy programových schém

- Trieda $\mathcal{S}_1$ je **silnejšia** ako trieda $\mathcal{S}_2$ (označenie $\mathcal{S}_1 \sqsupseteq \mathcal{S}_2$), ak ku každej schéme $S_2 \in \mathcal{S}_2$ existuje $S_1 \in \mathcal{S}_1$ taká, že $S_1 \equiv S_2$ (trieda $\mathcal{S}_2$ je preložiteľná do $\mathcal{S}_1$).
- Trieda $\mathcal{S}_1$ je **ostro silnejšia** ako trieda $\mathcal{S}_2$ ($\mathcal{S}_1 \sqsupset \mathcal{S}_2$), ak $\mathcal{S}_1$ je silnejšia ako $\mathcal{S}_2$ a existuje $S_1 \in \mathcal{S}_1$, ku ktorej neexistuje ekvivalentná schéma $S_2 \in \mathcal{S}_2$.
- Triedy $\mathcal{S}_1, \mathcal{S}_2$ sú **rovnocenné**, ak $\mathcal{S}_1 \sqsupseteq \mathcal{S}_2$ a $\mathcal{S}_2 \sqsupseteq \mathcal{S}_1$.
- Schéma $\mathcal{S}_1$ je **efektívne preložiteľná** do $\mathcal{S}_2$ ak existuje algoritmus, ktorý transformuje ľubovolnú schému $S_1 \in \mathcal{S}_1$ do ekvivalentnej schémy $S_2 \in \mathcal{S}_2$.
- Triedy $\mathcal{S}_1, \mathcal{S}_2$ sú **efektívne rovnocenné**, ak $\mathcal{S}_1$ je efektívne preložiteľná do $\mathcal{S}_2$ a $\mathcal{S}_2$ je efektívne preložiteľná do $\mathcal{S}_1$.

Príklad:

$\mathcal{V}$ – trieda voľných schém, $\mathcal{J}$ – trieda Janovových scém.

- $\mathcal{V} \sqsubset \mathcal{S}$, $\mathcal{J} \sqsubset \mathcal{V}$

Štruktúrované schémy

Symboly – ako pri štandardných schémach

Termy a predikáty – ako pri štandardných schémach

Príkazy – $C = \{st, st_i, \cdots\}$

1. počiatočný – **begin** $[\overline{y}] := [t_1(\overline{x}), \cdots, t_n(\overline{x})]$
2. koncový – **end** $[\overline{z}] := [t_1(\overline{x}, \overline{y}), \cdots, t_n(\overline{x}, \overline{y})]$
3. priraďovací – $[\overline{y}] := [t_1(\overline{x}, \overline{y}), \cdots, t_n(\overline{x}, \overline{y})]$
4. zložený – st_1 ; st_2,
5. príkaz cyklu – **while** $p(\overline{x}, \overline{y})$ **do** st **od**
6. podmienkový – **if** $p(\overline{x}, \overline{y})$ **then** st_1 **else** st_2 **fi**

Štruktúrované schémy – $\mathcal{W} = \{W, W_i, \cdots\}$

begin $[\overline{y}] := [t_1(\overline{x}), \cdots, t_n(\overline{x})]$
st
end $[\overline{z}] := [t_1(\overline{x}, \overline{y}), \cdots, t_n(\overline{x}, \overline{y})]$

Interpretácia, program, stav a výsledok výpočtu – ako pri štandardných schémach

Výpočet – prirodzená operačná sémantika (simultánne priradenie)

Príklad:

W: **begin** $[y_1, y_2] := [x, a]$
while $p(y_1)$ **do** $[y_1, y_2] := [f(y_1), g(y_1, y_2)]$ **od**
end $[z]{:=}[y_2]$

$\mathcal{I}_1$	N
a	1
p	$y_1 \neq 0$
f	$y_1 - 1$
g	$y_1 * y_2$

P_1: **begin** $[y_1, y_2] := [x, 1]$
while $y_1 \neq 0$ **do** $[y_1, y_2] := [y_1 - 1, y_1 * y_2]$ **od**
end $[z]{:=}[y_2]$

Výpočet - simultánne priradenie - $[2, 1], [1, 2], [0, 2], [2]$

Výpočet - sekvenčné priradenie - $[2, 1], [1, 1], [0, 0], [0]$

Veta: $\mathcal{W} \sqsubseteq \mathcal{S}$

Ku každej štruktúrovanej schéme W existuje ekvivalentná štandardná schéma S_W.

Dôkaz: indukciou vzhľadom na štruktúru tela st schémy W.

- úvodný krok indukcie: st preložíme na $1 : st$,
- $i : [\overline{y}] := [\overline{t}(\overline{x}, \overline{y})]$ – pri preklade opíšeme,
- $i :$ **if** q **then** st_1 **else** st_2 – preložíme na fragment schémy v stĺpci 1,
- $i : st_1; st_2$ – preložíme do fragmentu zo stĺpca 2,
- $i :$ **while** q **do** st **od** – preložíme do fragmentu v stĺpci 3.

$i :$ **if** q **then goto** $j+1$	$i :$	$i :$ **if** q **then goto** $i+2$
$i+1 :$	$\cdots \quad st_1$	$i+1 :$ **goto** $j+1$
$\cdots \quad st_2$	$j-1 :$	$i+2 :$
$j-1 :$	$j :$	$\cdots \quad st$
$j :$ **goto** k	$\cdots \quad st_2$	$j-1 :$
$j+1 :$	$k-1 :$	$j :$ **goto** i
$\cdots \quad st_1$	$k : \cdots$	$j+1 : \cdots$
$k-1 :$		
$k : \cdots$		

- dôkaz ekvivalencie W a S_W je priamočiary.

Veta: $\mathcal{W} \sqsubset \mathcal{S}$

Neexistuje štruktúrovaná schéma $W \in \mathcal{W}$, ekvivalentná s voľnou štandardnou schémou

S: **begin** $[y] := [x]$
1: **if** $p_1(y)$ **then goto** 3
2: **goto end**
3: **if** $p_2(y)$ **then goto** 5
4: **goto end**
5: $[y] := [f(y)]$
6: **goto** 1
end $[z] := [y]$

Dôkaz: predpokladajme, že existuje $W \in \mathcal{W}$ taká, že $W \equiv S$.

- W musí obsahovať aspoň jeden cyklus s podmienkou p_1 (alebo p_2).
- Existujú voľné interpretácie $\mathcal{I}_H$, pri ktorých sa cyklus **while** p_1 **do** $\cdots$ **od** vykoná.
- Uvažujme $\mathcal{I}_H$:
  - $i(p_1)(t) = 1$ pre všetky $t \in \mathcal{H}$,
  - $i(p_2)(t) = 1$ len pre tie $t \in \mathcal{H}$, ktoré sa vyskytujú pri výpočte $(W, \mathcal{I}_H, \text{“}x\text{”})$, kým sa nezačne výpočet cyklu s predikátom p_1 ($i(p_2)(t) = 1$ len pre konečný počet termov z $\mathcal{H}$).
- $(S, \mathcal{I}_H, \text{“}x\text{”})$ sa zastaví, zatiaľ čo $(W, \mathcal{I}_H, \text{“}x\text{”})$ cyklí – spor s predpokladom $S \equiv W$.

Rekurzívne schémy

Symboly

- funkčné a predikátové symboly – ako pri štandardných schémach,
- vstupné a pracovné premenné – ako pri štandardných schémach,
- výstupná premenná – z,
- funkčné premenné – $\Phi = \cup_{i=1}^{\infty} \Phi^i$,
  - $\Phi^n : \{\phi, \phi_i, \cdots\}$ – n–árne funkčné premenné.

Termy – $T = \{t[\overline{\phi}](\overline{x}, \overline{y}), \cdots\}$

- základ ako pri štandardných schémach,
- ak $\phi \in \Phi^n, t_1, \cdots, t_n \in T$, potom $\phi(t_1, \cdots, t_n) \in T$,
- ak $q \in P, t_1, t_2 \in T$, potom **if** q **then** t_1 **else** $t_2 \in T$.

Rekurzívne schémy – $\mathcal{R} = \{R, R_i, \cdots\}$

$$
\begin{array}{rl}
R\text{:} & \textbf{begin } [\overline{y}] := [\overline{t}(\overline{x})] \\
& \quad \phi_1(\overline{y}) \Longleftarrow t_1[\overline{\phi}](\overline{x}, \overline{y}) \\
& \qquad \vdots \\
& \quad \phi_n(\overline{y}) \Longleftarrow t_n[\overline{\phi}](\overline{x}, \overline{y}) \\
& \textbf{end } [z] := [t[\overline{\phi}](\overline{x})]
\end{array}
$$

Vstupný term - výpočet začína termom $t[\overline{\phi}](\overline{x})$

Systémy na prepisovanie termov

Prepisovacie pravidlo – orientovaná rovnosť

$$l \mapsto r$$

- $l, r \in T(F, X)$ – termy nad symbolmi F a premennými X
- $Var(r) \subseteq Var(l)$

Systém na prepisovanie termov – trs (term rewriting system)

$$R = \{l_1 \mapsto r_1, \cdots, l_n \mapsto r_n\}$$

- konečná množina prepisovacích pravidiel

Prepisovací krok – krok odvodenia systémom R

$$s \rightarrow_R t$$

- $s/\alpha = u = \sigma l$
  podterm $u = s/\alpha$ termu s na pozícii α je R–redexom, ak existujú pravidlo $l \mapsto r \in R$ a substitúcia σ taká, že $u = \sigma l$ (term s môže mať viac redexov, na rôznych pozíciách)
- $t = s[\alpha \leftarrow \sigma r]$
  redex $u = s/\alpha$ sa nahradí inštanciou pravej strany pravidla σr

R-odvodenie – výpočtová postupnosť $s_1 \rightarrow_R s_2 \rightarrow_R \cdots \rightarrow_R s_n$

Vlastnosti prepisovacích systémov

Normálny tvar – term u je R–normálnym tvarom termu t, ak existuje R–odvodenie $t \to_R^* u$ a u je R–ireducibilný term (nemá redex)

Normálne odvodenie – R–odvodenie $s \to_R^* t$ je R–normálne, ak t je R–normálnym tvarom termu s

Nedeterministický vypočtový model – pripúšťa

- R–odvodenia z termu t s rôznymi R–normálnymi tvarmi
- nekonečné R–odvodenia

Normalizácia – prepisovací systém R je

- (slabo) normalizujúci, ak pre každý uzavretý term (term bez premenných) $t \in T(F)$ existuje R–normálne odvodenie
- **jednoznačne normalizujúci**, ak pre ľubovoľný uzavretý term $s \in T(F)$ a R–normálne odvodenia $s \to_R^* t, s \to_R^* u$ platí $t = u$

Zastavenie – prepisovací systém R sa zastaví – **neotherovský systém**

- neexistuje nekonečné odvodenie – $s_1 \to_R s_2 \to_R \cdots s_n \to_R \cdots$

Konvergencia – konvergentný prepisovací systém R

- jednoznačne normalizujúci + neotherovský systém.

Rekurzívne programy

Rekurzívny program – $P = (R, \mathcal{I})$ s interpretáciou $\mathcal{I} = (D, i)$

Ohodnotenie vstupných premenných – $v : X_x^k \mapsto D^k$

Model pre výpočet programu – systém na prepisovanie termov

- rekurzívna definícia – prepisovacie pravidlo $\phi(\overline{y}) \rightarrow t_1[\overline{\phi}](\overline{x}, \overline{y})$
- systém rekurzívnych definícií - systém na prepisovanie termov

Výpočet $(R, \mathcal{I}, v)$ – výpočtová postupnosť – $\alpha_0, \alpha_1, \cdots, \alpha_i, \cdots$

- stav výpočtu – term α_i,
- počiatočný stav $\alpha_0 = i(t[\overline{\phi}](\overline{v(x)}))$ – interpretovaný vstupný term
- α_{i+1} vytvoríme z α_i nahradením najľavejšieho a najvnútornejšieho výskytu symbola ϕ_j zodpovedajúcim výrazom $i(t_j[\overline{\phi}](\cdots))$
- uskutočníme všetky možné zjednodušenia, definované interpretáciou

Výsledok výpočtu –

- $val(R, \mathcal{I}, v) = d$ – ak sa d nedá ani prepísať rekurzívnymi definíciami ani zjednodušiť
- $val(R, \mathcal{I}, v)$ nie je definovaná – v prípade nekonečného výpočtu

Formálna operačná sémantika rekurzívnych programov

Prepisovací systém – $P = R + Z$

- R – množina interpretovaných rekurzívnych definícií daného rekurzívneho programu
- Z – kánonický prepisovací systém zodpovedajúci pravidlám, ktoré charakterizujú vlastnosti preddefinovaných funkcií (zjednodušujúce pravidlá)

Výpočtový krok – $s \Rightarrow_P t \quad \equiv \quad \rightarrow_R^{val} \rightarrow_Z^*$

- krok výpočtu je definovaný "metapravidlom", riadiacim poradie aplikácií prepisovacích pravidiel P
- pri kroku $\rightarrow_R^{val}$ sa redukuje najľavejší z najvnútornejších R–redexov termu s (t.j. najľavejší podterm termu s, ktorého vlastné podtermy neobsahujú symboly funkčných premenných – volanie hodnotou)
- krokom $\rightarrow_Z^*$ sa (príslušný) term prepíše do Z–normálneho tvaru

Výsledok výpočtu – P–normálny term t

- prepisovací systém P je jednoznačne normalizujúci
- t neobsahuje symbol funkčnej premennej
- t je Z–ireducibilný term

Príklad:

R_1: **begin** $[y_1, y_2] := [x, x]$

$\phi(y_1, y_2) \Longleftarrow$ **if** $p(y_1)$ **then** $f_1(y_2)$
else if $p(y_2)$ **then** $\phi(f_2(y_1), a)$
else $\phi(f_2(y_1), \phi(y_1, f_2(y_2)))$

end $[z] := [\phi(x, x)]$

$\mathcal{I}$	N
a	1
p	$y = 0$
f_1	$y + 1$
f_2	$y - 1$

Program $P = (R_1, \mathcal{I})$ – Ackermannova funkcia:

P: **begin** $[y_1, y_2] := [x, x]$

$\phi(y_1, y_2) \Longleftarrow$ **if** $y_1 = 0$ **then** $y_2 + 1$
else if $y_2 = 0$
then $\phi(y_1 - 1, 1)$
else $\phi(y_1 - 1, \phi(y_1, y_2 - 1))$

end $[z] := [\phi(x, x)]$

Výpočet programu $(R_1, \mathcal{I}, \{x \mapsto 1\})$

$$\phi(1,1) \to \phi(0, \phi(1,0)) \to \phi(0, \phi(0,1)) \to \phi(0,2) \to 3$$

Veta: $\mathcal{S} \sqsubseteq \mathcal{R}$

Každá štandardná schéma S sa dá preložiť do ekvivalentnej rekurzívnej schémy R_S.

Dôkaz:

- Pred každý príkaz st_i predradíme funkčnú premennú $\phi_i(\overline{y})$, resp. $\phi_b(\overline{y})$, $\phi_e(\overline{y})$ – deliace body.
- Spätné substitúcie – vzťahy medzi ϕ_i, ϕ_j:
  - $\underline{\phi_b}$ **begin** $[\overline{y}] := [\overline{t}(\overline{x})]$ $\underline{\phi_1}$
    $$\phi_b(\overline{y}) \;=\; z \;\Leftarrow\; \phi_1(t(\overline{x}))$$
  - $\underline{\phi_e}$ **end** $[z] := [t(\overline{x}, \overline{y})]$
    $$\phi_e(\overline{y}) \;\Leftarrow\; t(\overline{x}, \overline{y})$$
  - $\underline{\phi_i}$ $i:$ $[\overline{y}] := [\overline{t}(\overline{x}, \overline{y})]$ $\underline{\phi_{i+1}}$
    $$\phi_i(\overline{y}) \;\Leftarrow\; \phi_{i+1}(\overline{t}(\overline{x}, \overline{y}))$$
  - $\underline{\phi_i}$ $i:$ **goto** j $\underline{\phi_j}$
    $$\phi_i(\overline{y}) \;\Leftarrow\; \phi_j(\overline{y})$$
  - $\underline{\phi_i}$ $i:$ **if** $p(\overline{x}, \overline{y})$ **then** $[\overline{y}] := [\overline{t}(\overline{x}, \overline{y})]$ $\underline{\phi_{i+1}}$
    $$\phi_i(\overline{y}) \;\Leftarrow \textbf{if } p(\overline{x}, \overline{y}) \textbf{ then } \phi_{i+1}(\overline{t}(\overline{x}, \overline{y})) \textbf{ else } \phi_{i+1}(\overline{y})$$
  - $\underline{\phi_i}$ $i:$ **if** $p(\overline{x}, \overline{y})$ **then goto** j $\underline{\phi_j}$
    $$\phi_i(\overline{y}) \;\Leftarrow \textbf{if } p(\overline{x}, \overline{y}) \textbf{ then } \phi_j(\overline{y}) \textbf{ else } \phi_{i+1}(\overline{y})$$
- Zjednodušenie systému rekurzívnych definícií.
- S a R_S sú ekvivalentné – indukcia vzhľadom na výpočty (vektor argumentov ϕ_i zodpovedá vektoru pracovných premenných).

Príklad:

S: **begin** $[y_1, y_2] := [x, a]$

1: **if** $p(y_1)$ **then goto end**
2: $[y_1, y_2] := [f(y_1), g(y_1, y_2)]$
3: **goto** 1

end $[z] := [y_2]$

$\phi_b(y_1, y_2) = z = \phi_1(x, a)$
$\phi_1(y_1, y_2) =$ **if** $p(y_1)$ **then** $\phi_e(y_1, y_2)$ **else** $\phi_2(y_1, y_2)$
$\phi_2(y_1, y_2) = \phi_3(f(y_1), g(y_1, y_2))$
$\phi_3(y_1, y_2) = \phi_1(y_1, y_2)$
$\phi_e(y_1, y_2) = y_2$

$z := \phi_1(x, a)$

$\phi_1(y_1, y_2) \Leftarrow$ **if** $p(y_1)$ **then** y_2 **else** $\phi_1(f(y_1), g(y_1, y_2))$

Veta: $\mathcal{S} \sqsubset \mathcal{R}$

K rekurzívnej schéme

R_A: **begin** $[y] := [a]$

$\phi(y) \Longleftarrow$ **if** $p(y)$ **then** $f(y)$ **else** $h(\phi(g_1(y)), \phi(g_2(y)))$

end $[z] := [\phi(a)]$

neexistuje žiadna ekvivalentná štandardná schéma.

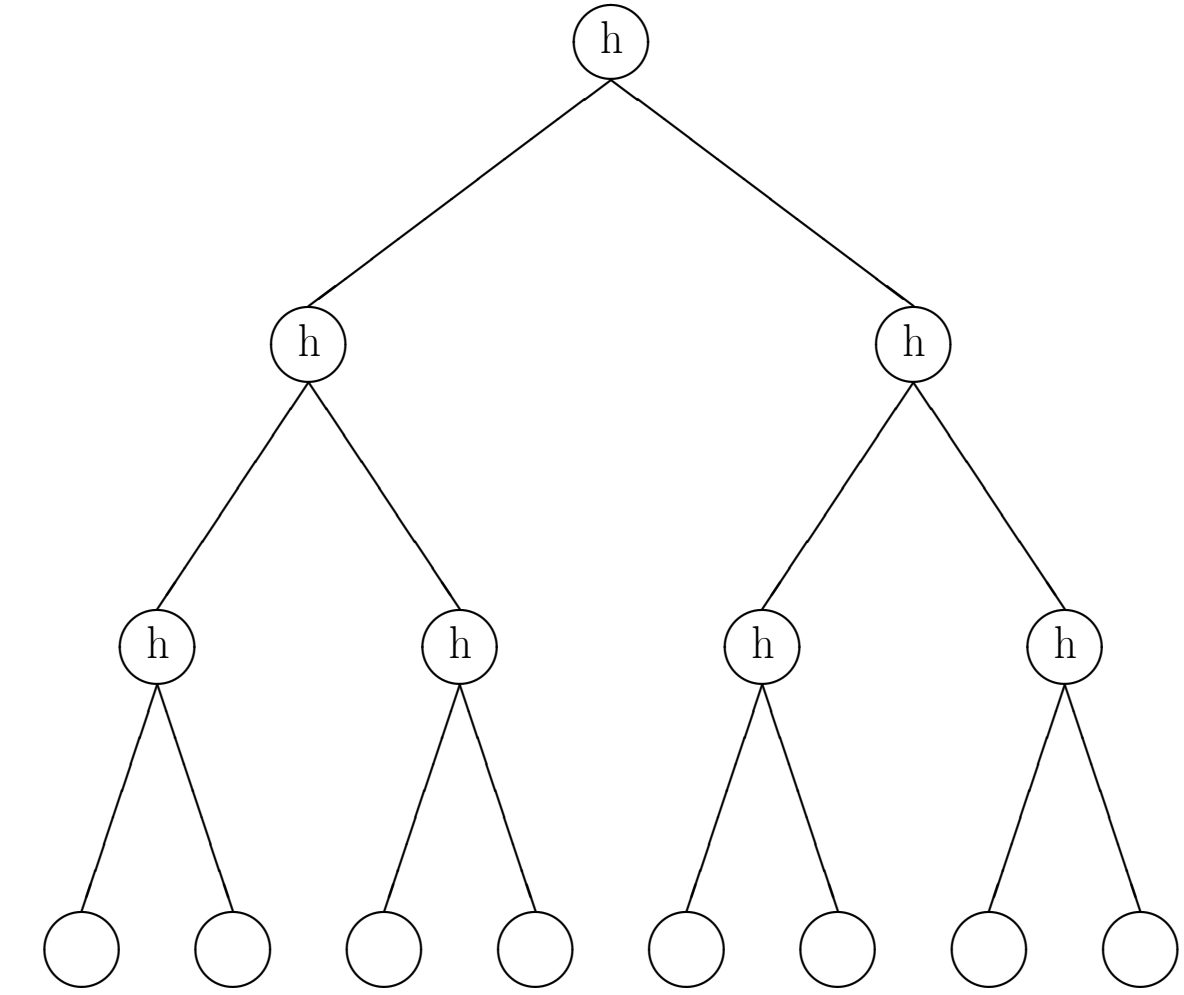
Dôkaz:

- uvažujme podtriedu Herbrandových interpretácií $\{\mathcal{I}_n^*\}_{n\geq 0}$:
  - Herbrandovo univerzum zo symbolmi a, f, g_1, g_2, h
  - $p(t) = 1$, keď t obsahuje n výskytov symbolov g_i
- výstupné hodnoty $val(R_A, \mathcal{I}_n^*)$ pre $n = 0, 1, 2, \cdots$:

$$\begin{array}{ll} n = 0 & z = f(a) \\ n = 1 & z = h(fg_1a, fg_2a) \\ n = 2 & z = h(h(fg_1g_1a, fg_1, g_2a), h(fg_2, g_1a, fg_2g_2a)) \end{array}$$

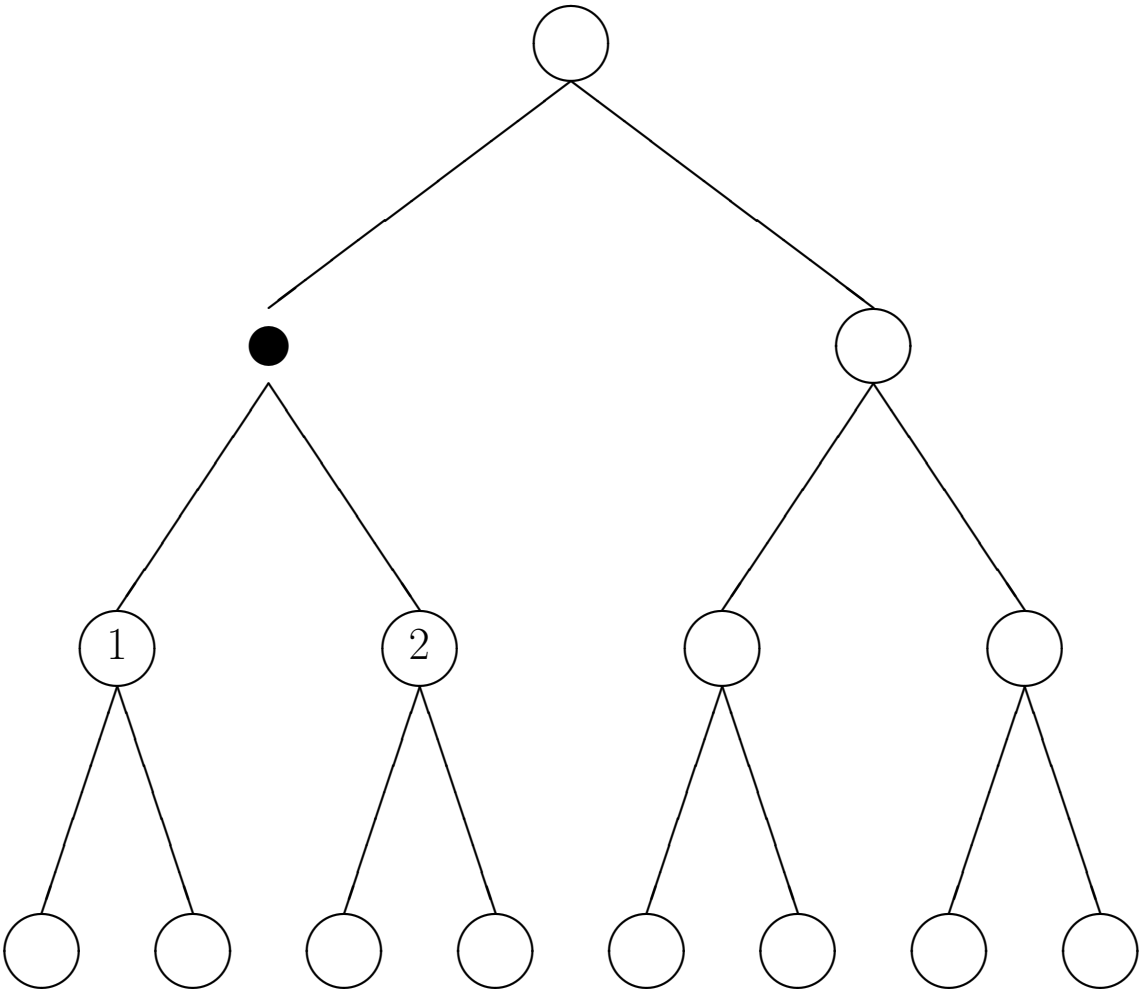
- reprezentácia termu pre $\mathcal{I}_n^*$ binárnym stromom T_n
- hra s kameňmi na binárnom strome (peblovanie) – (pravidlá umiestnenia a odobratia kameňa)
- súvislosť medzi hrou a nutnými priradeniami – (všetky podstromy sú navzájom rôzne)
- minimálny počet kameňov pre pokrytie koreňa stromu
- na pokrytie stromu T_n treba práve $n + 1$ kameňov

Reprezentácia termu T_n binárnym stromom



- strom T_3 reprezentuje term definovaný interpretáciou $\mathcal{I}_3^*$ (term obsahuje na každej ceste práve 3 symboly g_i) – zovšeobecnenie T_n
- žiadne dva podstromy nie sú identické – reprezentujú rôzne podtermy

Zložitosť výpočtu termu vs. peblovanie



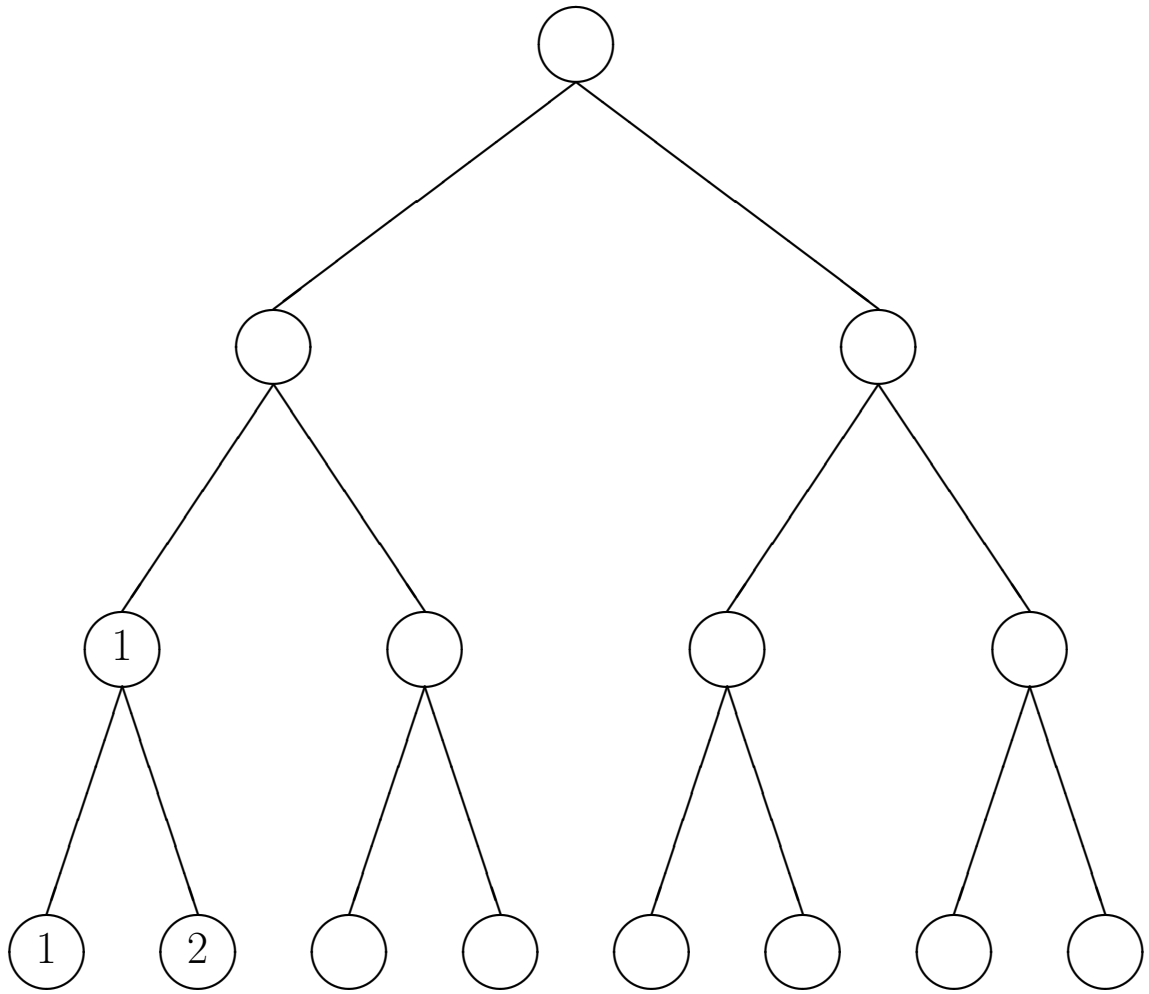
Peblovanie – hra s kameňmi na strome

- pravidlo – kameň môžme položiť na vrchol, ak sú kamene na oboch potomkoch
- cieľ – pokryť koreň stromu

Zložitosť výpočtu – počet použitých premenných

- priradenie vs. peblovací krok – $x_1 := h(x_1, x_2)$

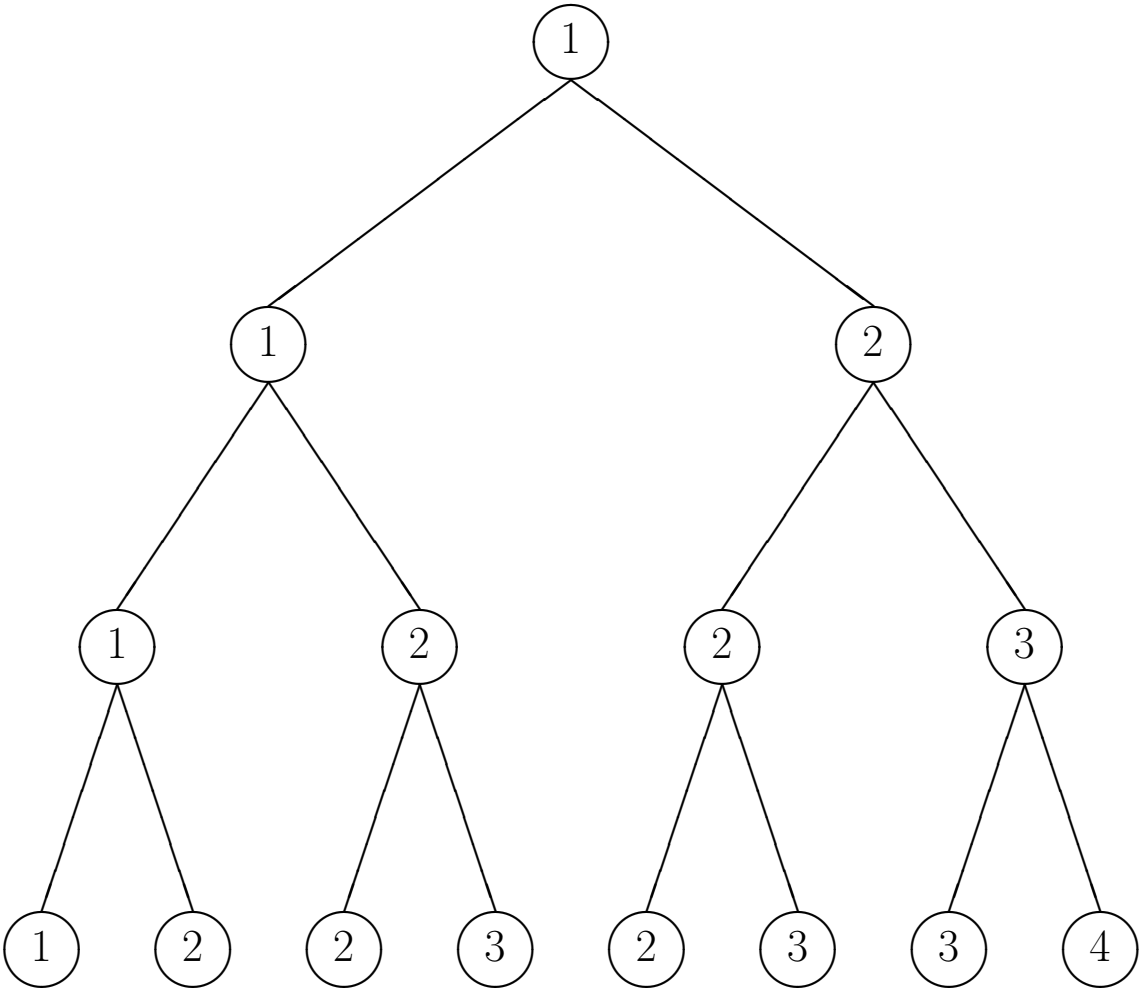
Peblovanie – horný odhad – algoritmus



Algoritmus – od listov zľava–doprava, akonáhle sa dá zdola–nahor

- ak sú pokrytý obaja potomkovia vrcholu, presunieme jeden z kameňov zo syna na otca a druhý odstránime
- ak nie sú, položíme kameň na nasledujúci list
- postup opakujeme, pokým nie je pokrytý koreň

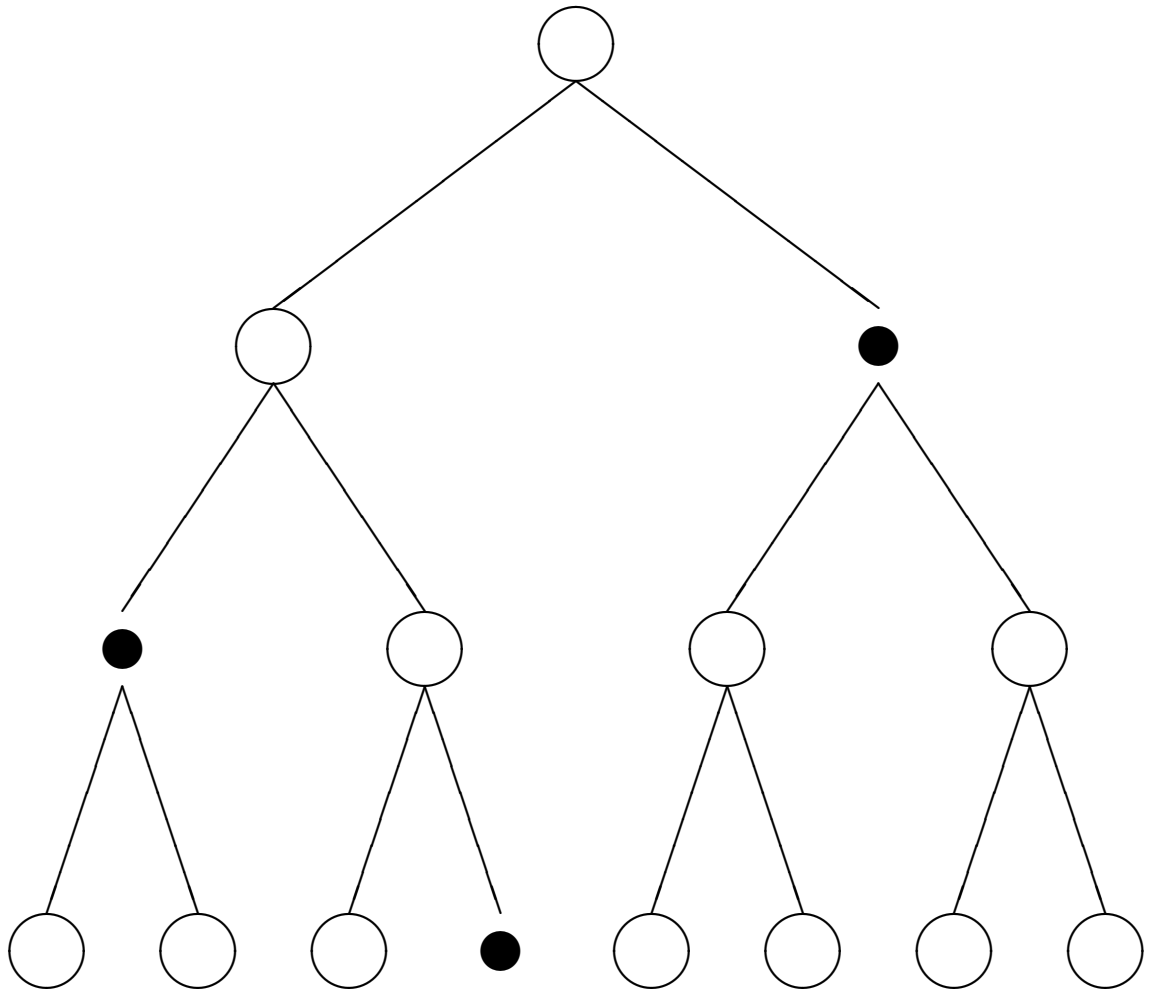
Peblovanie – horný odhad – tvrdenie



Tvrdenie – na pokrytie ľubovoľného binárneho stromu T_n stačí $n+1$ kameňov

Dôsledok – na výpočet termu T_n stačí $n+1$ premenných

Peblovanie – dolný odhad – predpoklad

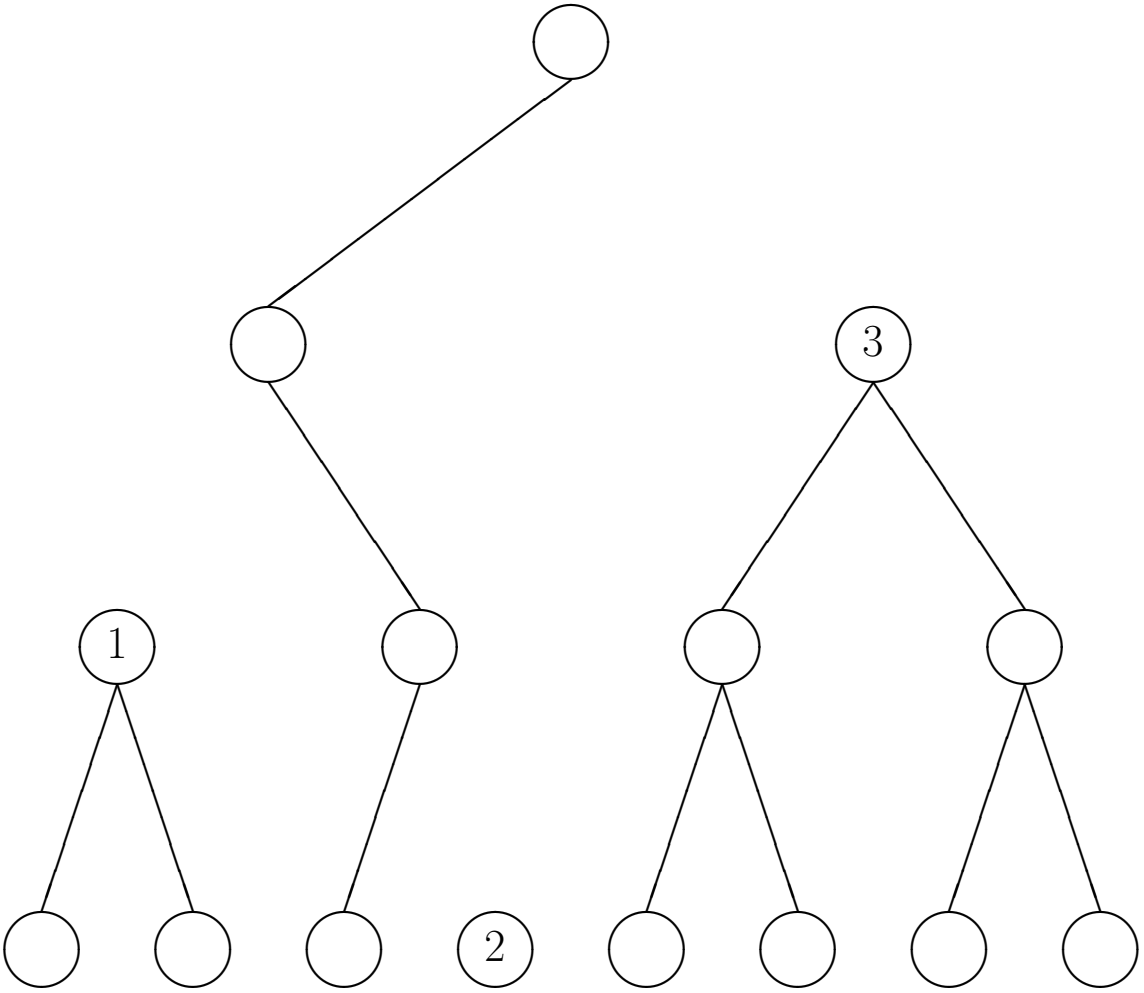


otvorený strom – existuje cesta z koreňa stromu do listu, na ktorej nie je kameň (otvorená cesta)

uzavretý strom – na každej ceste z koreňa do listu je aspoň jeden kameň

zmena otvoreného na uzavretý strom – jedine položením kameňa na list otvoreného stromu

Peblovanie – dolný odhad – tvrdenie



rozklad stromu – odstránením otvorenej cesty

Tvrdenie (dolný odhad) – na výpočet termu T_n potrebujeme práve $n + 1$ premenných

Čiastočne interpretované schémy

obohatenie triedy štandardných schém o interpretované objekty (premenné, predikátové a funkčné symboly).

Schémy s počítadlami – $\mathcal{S}^c$

- umožňujú synchronizáciu a kontrolu výpočtu (cyklické fragmenty)
- konečný počet počítadiel – premenných y_c (inicializácia 0),
- operácie: $y_c := y_c + 1, y_c := y_c - 1$; predikát: $y_c = 0$.

Schémy so zásobníkmi – $\mathcal{S}^s$

- simulujú nekonečnú dynamickú pamäť (potenciálne nekonečnú)
- konečný počet zásobníkov – premenných y_s (inicializácia $empty$),
- operácie: $y_s := push(\overline{y}), \overline{y} := toppop(y_s)$; predikát: $y_s = empty$.

Porovnanie tried schém – $\mathcal{S}, \mathcal{R}, \mathcal{S}^c, \mathcal{S}^s$

- $\mathcal{S}^c \sqsupset \mathcal{S}$, $\mathcal{S}^{1c} = \mathcal{S}$, $\mathcal{S}^{2c} = \mathcal{S}^c$;
- $\mathcal{S}^s \sqsupset \mathcal{S}$;
- $\mathcal{S}^c \not\sqsupseteq \mathcal{R}$, $\mathcal{S}^c \not\sqsubseteq \mathcal{R}$;
- $\mathcal{S}^s \not\sqsubseteq \mathcal{R}$, $\mathcal{S}^s \sqsupset \mathcal{R}$, $\mathcal{S}^{1s} = \mathcal{R}$, $\mathcal{S}^{1s} \sqsubset \mathcal{S}^{2s}$;
- $\mathcal{S}^c \sqsubset \mathcal{S}^s$.