# 2. domáca úloha z predmetu 2-AIN-109 Programovanie paralelných a distribuovaných systémov LS 2023/24

Tomáš Bisták

5. marca 2024

## 1. úloha

Dokážte (ako viete ;-), že program `Binomial`:

Program Binomial
assign $\langle \Box n : 0 \leq n < N ::$
$\mathtt{c}[n,0] := 1 \Box \mathtt{c}[n,n] := 1$
$\Box \langle \Box k : 0 < k < n :: \mathtt{c}[n,k] := \mathtt{c}[n-1,k-1] + \mathtt{c}[n-1,k] \rangle \rangle$
end

funguje, t.j.:

a) Neurobí nič zlého (špecifikujte čo máte na mysli)

b) Urobí nakoniec niečo dobre (špecifikujte čo máte na mysli)

c) Dosiahne *FP* a vtedy je to "hotovo" (napíšte čo).

Predstavte si, že na začiatku inicializujeme všetky hodnoty na 0 v druhom prípade na 1. Dal by sa dôkaz, že program `Binomial` funguje, zjednodušiť?

### Riešenie

Úlohou programu `Binomial` je vyplniť tabuľku `c` tak, aby v nej napokon v pevnom bode na každej pozícii $(n,k)$, kde $n \in \{0,1,\ldots,N-1\}$ a $k \in \{0,1,\ldots,n\}$, bolo príslušné kombinačné číslo $C(n,k) = \binom{n}{k}$. Na úvod si

preto zadefinujeme metriku $\mu$, ktorou budeme merať, do akej miery je tabuľka `c` správne vyplnená. Táto metrika bude konkrétne každý stav vykonávania programu `Binomial` ohodnocovať usporiadanou dvojicou $(n^*, k^*)$ zodpovedajúcou prvej pozícii v `c`, na ktorej je nesprávna hodnota, t.j. iná ako $C(n, k)$, resp. prvej pozícii, ktorá sa v `c` už nenachádza, pri úplne správnej tabuľke `c`. Pozície budeme pritom vzostupne zoraďovať a porovnávať podľa štandardných relácií $<$ a $=$ nad usporiadanými dvojicami[1], ktoré sú pre všetky $(n_0, k_0), (n_1, k_1) \in \mathbb{N} \times \mathbb{N}$ určené vzťahmi:

$$(n_0, k_0) < (n_1, k_1) \equiv (n_0 < n_1 \vee (n_0 = n_1 \wedge k_0 < k_1)),$$
$$(n_0, k_0) = (n_1, k_1) \equiv (n_0 = n_1 \wedge k_0 = k_1).$$

Množinu všetkých pozícií (indexov) v `c` budeme tiež pre zjednodušenie zápisu označovať $I$.

Pre každé vykonávanie $R$ programu `Binomial` a každý stav $R_j$.state v ňom si teda formálne mieru $\mu(R_j.\text{state})$ definujeme ako:

$$\mu(R_j.\text{state}) = (n^*, k^*),$$

kde

$$n^* = \max\{n \in \{0, 1, \ldots, N\} \mid \forall (n', k') \in I : n' < n \Rightarrow \texttt{c}[n', k'] = C(n', k')\},$$
$$k^* = \text{ak } n^* = N, \text{ tak } 0, \text{ inak}$$
$$\max\{k \in \{0, 1, \ldots, n^*\} \mid \forall (n^*, k') \in I : k' < k \Rightarrow \texttt{c}[n^*, k'] = C(n^*, k')\}.$$

Výraz $\mu(R_j.\text{state})$ budeme nahrádzať iba $\mu$, ak je stav zrejmý z kontextu.

Môžeme si všimnúť, že oborom hodnôt $H(\mu)$ metriky $\mu$ je množina $I \cup \{(N, 0)\}$ usporiadaná v $<$ od $(0, 0)$ po $(N, 0)$. Správnosť programu `Binomial` tak ukážeme overením, že (a) $\mu$ v priebehu jeho vykonávania neklesá, (b) ak v nejakom stave je hodnota $\mu$ rôzna od $(N, 0)$, tak sa raz určite zvýši a (c) každé vykonávanie `Binomial` dosiahne stav s $\mu$ rovnou $(N, 0)$, čo je hľadaným pevným bodom, v ktorom je `c` správne vyplnená.

a) *Dôkaz.* Dokazujeme tvrdenie, že pre všetky $(n^*, k^*) \in H(\mu)$ a všetky kroky $R_j$ ľubovoľného vykonávania $R$ programu `Binomial` platí, že ak $\mu(R_j.\text{state}) = (n^*, k^*)$, tak po vykonaní príkazu $R_j$.label bude v nasledujúcom stave $\mu(R_{j+1}.\text{state}) \geq (n^*, k^*)$.

   Vezmime si ľubovoľnú dvojicu $(n^*, k^*) \in H(\mu)$, nejaké vykonávanie $R$ programu `Binomial` a krok $R_j$ z $R$ taký, že $\mu(R_j.\text{state}) = (n^*, k^*)$. Potom v závislosti od príkazu $R_j$.label môžu nastať dovedna tri prípady:

[1] A takisto pomocou $>$, čo ale budeme chápať jednoducho ako inverznú reláciu k $<$.

1. Príkaz $R_j$.label priraďuje hodnotu do $\mathtt{c}[n,k]$ pre nejaké $(n,k) < (n^*,k^*)$. Keďže $(n^*,k^*)$ je prvou zle vyplnenou pozíciou v $\mathtt{c}$, vieme, že $\mathtt{c}[n,k]$ v $R_j$.state už musí obsahovať správne číslo, čiže $C(n,k)$.[2] Ak je teda $k = 0$ alebo $k = n$, t.j. $R_j$.label má tvar $\mathtt{c}[n,0]$ `:=` `1` alebo $\mathtt{c}[n,n]$ `:=` `1`, tak po vykonaní $R_j$.label bude platiť, že $\mathtt{c}[n,k]^{(R_{j+1}.\mathrm{state})} = 1 = C(n,k) = \mathtt{c}[n,k]^{(R_j.\mathrm{state})}$, a ostatné políčka v $\mathtt{c}$ ostanú (tiež) nezmenené. Zo skutočnosti, že $(n,k) < (n^*,k^*)$, však zároveň plynie, že aj pre všetky $(n',k') < (n,k)$ platí: $\mathtt{c}[n',k'] = C(n',k')$. Odtiaľ dostávame, že pokiaľ $R_j$.label je $\mathtt{c}[n,k]$ `:=` $\mathtt{c}[n-1,k-1]$ `+` $\mathtt{c}[n-1,k]$, jediným zásahom do $\mathtt{c}$ bude, že do $\mathtt{c}[n,k]$ sa zapíše $\mathtt{c}[n-1,k-1] + \mathtt{c}[n-1,k] = C(n-1,k-1) + C(n-1,k) = C(n,k) = \mathtt{c}[n,k]^{(R_j.\mathrm{state})}$. Bez ohľadu na tvar $R_j$.label (iné ako skúmané tvary príkazov v `Binomial` nemáme), sa tým pádom $\mathtt{c}$ nezmení, čo znamená, že $\mu(R_{j+1}.\mathrm{state}) = (n^*,k^*)$.
2. Príkaz $R_j$.label je priradením, v ktorom na ľavej strane je $\mathtt{c}[n^*,k^*]$. Ak $k^* = 0$, resp. $k^* = n^*$, tak $R_j$.label je $\mathtt{c}[n^*,0]$ `:=` `1` alebo $\mathtt{c}[n^*,n^*]$ `:=` `1` a v ďalšom stave máme $\mathtt{c}[n^*,k^*]^{(R_{j+1}.\mathrm{state})} = 1$, čiže na indexe $(n^*,k^*)$ bude správna hodnota. Z definície $\mu$ pritom vieme, že $(n^*,k^*)$ bola v $R_j$.state prvou pozíciou v $\mathtt{c}$ so zlou hodnotou, a preto pri $k^* = 0$ pre $\mu(R_{j+1}.\mathrm{state})$ získame, že $\mu(R_{j+1}.\mathrm{state}) \geq (n^*,1) > (n^*,0) = (n^*,k^*)$, a pri $k^* = n^*$ zase, že $\mu(R_{j+1}.\mathrm{state}) \geq (n^*+1,0) > (n^*,n^*) = (n^*,k^*)$. Teraz uvažujme, že $0 < k^* < n^*$, čo značí, že $R_j$.label má tvar $\mathtt{c}[n^*,k^*]$ `:=` $\mathtt{c}[n^*-1,k^*-1]$ `+` $\mathtt{c}[n^*-1,k^*]$. Z poznatku, že $(n^*,k^*) = \mu(R_j.\mathrm{state})$, t.j. že pre všetky $(n,k) < (n^*,k^*)$ je $\mathtt{c}[n,k] = C(n,k)$, môžeme aj tentokrát vyvodiť, že na $\mathtt{c}[n^*,k^*]$ sa dostane požadovaná hodnota: $\mathtt{c}[n^*,k^*]^{(R_{j+1}.\mathrm{state})} = \mathtt{c}[n^*-1,k^*-1] + \mathtt{c}[n^*-1,k^*] = C(n^*-1,k^*-1) + C(n^*-1,k^*) = C(n^*,k^*) \neq \mathtt{c}[n^*,k^*]^{(R_j.\mathrm{state})}$. Miera $\mu$ sa preto znova určite zvýši: $\mu(R_{j+1}.\mathrm{state}) \geq (n^*,k^*+1) > (n^*,k^*)$. V každom prípade tak platí, že $\mu(R_{j+1}.\mathrm{state}) > (n^*,k^*)$.
3. Príkaz $R_j$.label zapisuje hodnotu do $\mathtt{c}[n,k]$ pre nejaké $(n,k) > (n^*,k^*)$. Odhliadnuc od presného tvaru, $R_j$.label môže modifikovať tabuľku $\mathtt{c}$ len za $(n^*,k^*)$, čo značí, že prvý nesprávny prvok v $\mathtt{c}[n^*,k^*]$ a ani žiadne predošlé výsledok tejto operácie neovplyvní. Miera $\mu$ preto ostane rovnaká: $\mu(R_{j+1}.\mathrm{state}) = (n^*,k^*)$.

Vo všetkých prípadoch sa nám podarilo ukázať, že $\mu(R_{j+1}.\mathrm{state}) \geq$

[2]Pre jednoznačnosť budeme ďalej v hornom indexe pri názvoch premenných, resp. prvkov poľa uvádzať označenie stavu, v ktorom hodnoty daných premenných uvažujeme. Napr. $\mathtt{c}[n,k]$ v $R_j$.state ako $\mathtt{c}[n,k]^{(R_j.\mathrm{state})}$.

$(n^*, k^*)$, na základe čoho môžeme tvrdiť, že $\mu$ nikdy pri vykonávaní programu `Binomial` neklesne. □

Z formálneho hľadiska sme v skutočnosti dokázali, že program `Binomial` má vlastnosť: $\mu = (n^*, k^*)$ unless $\mu > (n^*, k^*)$, pre všetky $(n^*, k^*) \in H(\mu)$.

b) *Dôkaz.* V tejto časti je naším cieľom ukázať, že nech je $\mu$ v danom stave akákoľvek, raz bude určite vyššia alebo bude rovná maximu $(N, 0)$. Inými slovami, musíme overiť, že $\mu$ nikdy neklesne a že pre každé $\mu$ okrem $(N, 0)$ existuje jednoznačný príkaz, ktorý $\mu$ zvýši, čo nám spolu zabezpečí, že k tomuto zvýšeniu naozaj dôjde. V každom vykonávaní sa totiž každý príkaz vyberie nekonečne veľa ráz, čiže pri vykonávaní programu `Binomial` tak bude hodnota $\mu$, pokiaľ nie je rovná $(N, 0)$, vždy s odstupom konečného počtu krokov narastať.

Neklesajúcosť $\mu$ sme si dokázali v časti a), preto nám zostáva už len nájsť príslušné príkazy zvyšujúce $\mu$. Lenže aj tie vieme poľahky nájsť za pomoci úvah v časti a).

Nech $(n^*, k^*)$ je ľubovoľná dvojica z $I\,(= H(\mu) \setminus \{(N, 0)\})$ a $s$ nech je ľubovoľný stav v akomkoľvek vykonávaní programu `Binomial` taký, že $\mu(s) = (n^*, k^*)$. Z bodu 2. v časti a) potom dostávame, že hľadaným príkazom je príkaz priraďujúci hodnotu do $\mathtt{c}[n^*, k^*]$, lebo po jeho vykonaní bude pre všetky $(n, k) \leq (n^*, k^*)$ platiť, že $\mathtt{c}[n, k] = C(n, k)$, čiže $\mu > (n^*, k^*)$. □

Vlastnosť programu `Binomial`, ktorú sme v tejto časti ukázali, môžeme s prihliadnutím na isté technické detaily[3] sformulovať ako: $\mu = (n^*, k^*)$ ensures $(\mu > (n^*, k^*) \vee \mu = (N, 0))$, pre všetky $(n^*, k^*) \in H(\mu)$.

c) Najprv ukážeme, že každé vykonávanie `Binomial` vedie do stavu, v ktorom je $\mu$ rovná $(N, 0)$.

*Dôkaz.* Z častí a) a b) riešenia tohto príkladu vieme, že bez ohľadu na to, v ktorom stave vykonávanie začne, hodnota $\mu$ bude postupne po konečných počtoch krokov narastať, no len pokým nie je rovná $(N, 0)$. Množina možných hodnôt metriky $\mu$ je však konečná a jej najväčším prvkom je práve $(N, 0)$. Po konečnom počte krokov od začiatku vykonávania programu preto za každým určite dospejeme do stavu, kedy $\mu$ je rovná $(N, 0)$, čo sme chceli dokázať. □

[3] Ak aplikujeme triviálne pravdivú implikáciu: $\mu > (n^*, k^*) \Rightarrow (\mu > (n^*, k^*) \vee \mu = (N, 0))$.

Vo vyššie uvedených úvahách sme formálne využili indukčný princíp pre vlastnosť leads-to, ktorým sme za pomoci $\mu = (n^*, k^*) \rightarrow (\mu > (n^*, k^*) \vee \mu = (N, 0))$, čo plynie z tvrdenia $\mu = (n^*, k^*)$ ensures $(\mu > (n^*, k^*) \vee \mu = (N, 0))$ dokázaného v časti b), odvodili, že pre `Binomial` platí: `true` $\rightarrow \mu = (N, 0)$. Z konečnosti množiny $H(\mu)$ a lineárnosti usporiadania $<$ sme pritom implicitne usúdili, že $H(\mu)$ je dobre založená množina.

Po ukázaní, že každé vykonávanie `Binomial` vedie do stavu s $\mu$ rovnou $(N, 0)$ nám tak zostáva už iba overiť, že (1) tabuľka `c` v ňom je správne vyplnená a že (2) takýto stav je pevným bodom.

*Dôkaz.* Zoberme si ľubovoľný stav $R_j$.state akéhokoľvek vykonávania $R$ programu `Binomial` taký, že $\mu(R_j.\text{state}) = (N, 0)$.

1. Podľa definície $\mu$ situácia, že $\mu = (N, 0)$, nastáva vtedy a len vtedy, keď pre všetky $(n, k) \in I$ platí: $\texttt{c}[n, k] = C(n, k)$. V $R_j$.state je preto `c` správne vyplnená.
2. Pri pohľade na program `Binomial` vidíme, že žiadny príkaz za týchto podmienok tabuľku `c` (jedinú premennú) zmeniť nemôže, lebo buď by priradil do nejakého $\texttt{c}[n, 0]$ či $\texttt{c}[n, n]$ hodnotu $1 = C(n, 0) = C(n, n) = \texttt{c}[n, 0]^{(R_j.\text{state})} = \texttt{c}[n, n]^{(R_j.\text{state})}$, alebo do $\texttt{c}[n, k]$, kde $0 < k < n$, hodnotu $\texttt{c}[n-1, k-1] + \texttt{c}[n-1, k] = C(n-1, k-1) + C(n-1, k) = C(n, k) = \texttt{c}[n, k]^{(R_j.\text{state})}$. Naopak, ak neexistuje príkaz, ktorý by `c` modifikoval, tak `c` musí byť na základe práve uvedeného správne vyplnená. □

Každé vykonávanie programu `Binomial` teda určite dosiahne pevný bod, v ktorom je na každej pozícii $(n, k)$ v `c` hodnota $C(n, k)$, z čoho plynie požadovaná správnosť programu `Binomial`.

Na záver sa zamyslíme, či by inicializácia prvkov tabuľky `c` na 0, resp. 1 mohla zjednodušiť dôkaz správnosti `Binomial`. V prvom prípade by nám nanajvýš pribudol predpoklad, že na začiatku máme všade zlé hodnoty. V druhom by sme zase mohli uvažovať, že krajné hodnoty `c` sú od začiatku správne. Keďže ale priradenia nevykonávame v nijakom konkrétnom poradí, situáciu pri dokazovaní správnosti `Binomial` by nám žiadna z týchto inicializácií neuľahčila. Stále by sme totiž museli sledovať, aká súvislá časť tabuľky `c` od hora po riadkoch (alebo z ľavého kraja po stĺpcoch) je už správne vyplnená.[4]

[4]Predpokladáme vizualizáciu tabuľky `c` s prvkami zoradenými do riadkov zhora nadol podľa rastúcej prvej zložky ich pozície a usporiadanými v jednotlivých riadkoch do stĺpcov zľava doprava podľa rastúcej druhej zložky ich pozície.

# 2. úloha

Dokážte (ako viete;-), že nasledovné tvrdenia sú pravdivé/nepravdivé

a) $\{\texttt{x} > 3\}$ `x := x + 5` $\{\texttt{x} > 4\}$

b) $\{\texttt{x} > 3\}$ `x := x + 5` $\{\texttt{x} > 2\}$

c) $\{\texttt{x} > 3\}$ `x := x + 5` $\{\texttt{y} > 2\}$

d) $\{\texttt{x} > 3\}$ `x := x + 5` $\{\texttt{x} > 6\}$

e) $\{$`true`$\}$ `x := 2` $\cdot$ `x` $\{$`x` je párne$\}$

f) $\{$`false`$\}$ `x := 2` $\cdot$ `x` $\{$`x` je párne$\}$

g) $\{$`k` je celé kladné číslo$\}$ `x := k` $\cdot$ `x` $\{$`x` je zložené číslo$\}$

## Riešenie

V prípade, že je tvrdenie pravdivé, uvedieme dôkaz. Pri nepravdivých tvrdeniach zase poskytneme kontrapríklad. Pri posudzovaní pravdivosti budeme vychádzať zo všeobecnej interpretácie tvrdenia tvaru $\{p\}\ s\ \{q\}$, kde $p, q$ sú predikáty a $s$ je príkaz, ktoré hovorí, že pre všetky kroky $R_j$ v ľubovoľnom vykonávaní $R$ programu obsahujúceho príkaz $s$ také, že $R_j.\text{label} = s$, platí: $p[R_j.\text{state}] \Rightarrow q[R_{j+1}.\text{state}]$. Túto implikáciu budeme ďalej zjednodušovať na $p \Rightarrow q'$, kde $q'$ je obmena $q$, v ktorej je každý výskyt každej premennej z ľavej strany v $s$ nahradený príslušným priradením v $s$.

a) Tvrdenie platí.

*Dôkaz.* Nech `F` je ľubovoľný program s príkazom `x := x + 5`, nech $R$ je ľubovoľné vykonávanie `F` a $R_j$ nech je ľubovoľný krok v $R$ taký, že $R_j.\text{label} =$ `x := x + 5`. Naším cieľom je ukázať, že $\texttt{x} > 3 \Rightarrow \texttt{x} + 5 > 4$.

Platnosť tejto implikácie vyplýva z nasledovnej postupnosti úprav:

$$\texttt{x} > 3 \Rightarrow \texttt{x} + 5 > 8 \overset{8>4}{\Rightarrow} \texttt{x} + 5 > 4. \qquad \square$$

b) Tvrdenie platí.

*Dôkaz.* Nech `F` je ľubovoľný program s príkazom `x := x + 5`, nech $R$ je ľubovoľné vykonávanie `F` a $R_j$ nech je ľubovoľný krok v $R$ taký, že $R_j.\text{label} =$ `x := x + 5`. Naším cieľom je ukázať, že $\texttt{x} > 3 \Rightarrow \texttt{x} + 5 > 2$.

Platnosť tejto implikácie vyplýva z nasledovnej postupnosti úprav:

$$\texttt{x} > 3 \Rightarrow \texttt{x} + 5 > 8 \overset{8>2}{\Rightarrow} \texttt{x} + 5 > 2. \qquad \square$$

c) Tvrdenie neplatí.

*Kontrapríklad.* Nech `F` je program:

```
Program F
    initially y = 0
    assign x := x + 5
end
```

Nech ďalej $R$ je ľubovoľné vykonávanie `F` a $R_j$ nech je ľubovoľný krok v $R$ taký, že v $R_j$.state platí, že $\mathtt{x} > 3$, a $R_j$.label = `x := x + 5` (čo ale platí vždy, lebo v `assign` je len tento príkaz).

Pre `y` zjavne platí: $\mathtt{y} = 0$ (je totiž konštantou). Máme teda situáciu, že platí, že $\mathtt{x} > 3$, a súčasne je pravdou, že $\neg(\mathtt{y} > 2)$. Požadovaná implikácia, $\mathtt{x} > 3 \Rightarrow \mathtt{y} > 2$, tým pádom nemôže byť platná.

d) Tvrdenie platí.

*Dôkaz.* Nech `F` je ľubovoľný program s príkazom `x := x + 5`, nech $R$ je ľubovoľné vykonávanie `F` a $R_j$ nech je ľubovoľný krok v $R$ taký, že $R_j$.label = `x := x + 5`. Naším cieľom je ukázať, že $\mathtt{x} > 3 \Rightarrow \mathtt{x} + 5 > 6$.

Platnosť tejto implikácie vyplýva z nasledovnej postupnosti úprav:

$$\mathtt{x} > 3 \Rightarrow \mathtt{x} + 5 > 8 \overset{8 \geq 6}{\Rightarrow} \mathtt{x} + 5 > 6. \qquad \square$$

e) Ak `x` je typu `int`, tak tvrdenie platí.

*Dôkaz.* Nech `F` je ľubovoľný program s príkazom `x := 2 · x`, nech $R$ je ľubovoľné vykonávanie `F` a $R_j$ nech je ľubovoľný krok v $R$ taký, že $R_j$.label = `x := x + 5`. Naším cieľom je ukázať, že `true` $\Rightarrow$ $2 \cdot$ `x` je párne, teda len že $2 \cdot \mathtt{x}$ je párne (`true` totiž platí vždy).

Párnosť $2 \cdot \mathtt{x}$ môžeme na základe definície tohto pojmu dokázať overením, že $2 \cdot \mathtt{x}$ je deliteľné dvomi. Na to nám podľa definície deliteľnosti stačí nájsť také číslo $z \in \mathbb{Z}$, že $2 \cdot \mathtt{x} = 2 \cdot z$. V našom prípade platí, že `x` je celé číslo, hľadaným $z$ je preto rovno `x`. $\square$

Ak `x` nie je typu `int`, tak tvrdenie neplatí.

*Kontrapríklad.* Nech `F` je program:

```
Program F
    initially x = 1.2
    assign x := 2 · x
end
```

Nech ďalej $R$ je ľubovoľné vykonávanie `F` a $R_0$ nech je prvý krok v $R$. Keďže v `assign` sekcii `F` je jediný príkaz, $R_0$.label = `x := 2 · x`.

Pre $2 \cdot \texttt{x}$ zjavne platí: $2 \cdot \texttt{x} = 2 \cdot 1.2 = 2.4$ (lebo $R_0$ je prvý krok a `x` je inicializované na 1.2). Máme teda situáciu, že platí `true` (čo platí vždy), a súčasne je pravdou, že $2 \nmid 2 \cdot \texttt{x}$. Požadovaná implikácia, `true` $\Rightarrow 2 \cdot$ `x` je párne, tým pádom nemôže byť platná.

f) Tvrdenie platí.

*Dôkaz.* Nech `F` je ľubovoľný program s príkazom `x := 2 · x`, nech $R$ je ľubovoľné vykonávanie `F` a $R_j$ nech je ľubovoľný krok v $R$ taký, že $R_j$.label = `x := x + 5`. Naším cieľom je ukázať, že `false` $\Rightarrow 2 \cdot$ `x` je párne.

Platnosť tejto implikácie vyplýva z nasledovnej postupnosti úprav:

$$(\texttt{false} \Rightarrow 2 \cdot \texttt{x} \text{ je párne }) \equiv (\texttt{true} \vee 2 \cdot \texttt{x} \text{ je párne }) \equiv \texttt{true}. \qquad \square$$

g) Tvrdenie neplatí.

*Kontrapríklad.* Nech `F` je program:

```
Program F
    initially x = 0 || k = 3
    assign x := k · x
end
```

Nech ďalej $R$ je ľubovoľné vykonávanie `F` a $R_0$ nech je prvý krok v $R$. Keďže v `assign` sekcii `F` je jediný príkaz, $R_0$.label = `x := k · x`. Zároveň vieme, že v $R_0$.state je `k` celé kladné číslo (presnejšie 3).

Pre $\texttt{k} \cdot \texttt{x}$ zjavne platí: $\texttt{k} \cdot \texttt{x} = 0$ (lebo $R_0$ je prvý krok a `x` je inicializované na 0). Máme teda situáciu, že platí, že `k` je celé kladné číslo, a súčasne je pravdou, že $\texttt{k} \cdot \texttt{x} = 0$, čo podľa definície zloženého čísla nie je zloženým číslom. Požadovaná implikácia, `k` je celé kladné číslo $\Rightarrow$ $\texttt{k} \cdot \texttt{x}$ je zložené číslo, tým pádom nemôže byť platná.

# 3. úloha

Sformulujte nejaké vlastnosti unless, stable, ensures, leads-to pre program `Binomial`.

## Riešenie

Okrem vlastností spomenutých v riešení prvého príkladu môžeme pre program `Binomial` sformulovať aj nasledovné vlastnosti.

**unless.** Priamym prepísaním všetkých rôznych príkazov v `Binomial` dostávame:

$$\begin{gathered}\texttt{true}\ \text{unless}\ \texttt{c}[n,0]=1,\\ \texttt{true}\ \text{unless}\ \texttt{c}[n,n]=1,\\ \texttt{true}\ \text{unless}\ \texttt{c}[n,k]=\texttt{c}[n-1,k-1]+\texttt{c}[n-1,k],\end{gathered}$$

samozrejme, implicitne kvantifikované cez všetky vhodné $n$ a $k$.[5]

**stable.** Keďže v pevnom bode platí, že $\mu=(N,0)$, určite platí aj vlastnosť:

$$\mu=(N,0)\ \text{is stable}.$$

Takisto si môžeme všimnúť, že hodnoty na krajoch tabuľky ($\texttt{c}[n,0]$ a $\texttt{c}[n,n]$) sa po dosiahnutí správnej (čiže 1) už nemenia, z čoho môžeme vyvodiť:

$$\begin{gathered}\texttt{c}[n,0]=1\ \text{is stable},\\ \texttt{c}[n,n]=1\ \text{is stable},\\ \texttt{c}[n,0]\ \text{je nepárne}\ \text{is stable},\\ \texttt{c}[n,n]\ \text{je nepárne}\ \text{is stable}.\end{gathered}$$

V časti a) riešenia prvého príkladu sme si tiež v podstate dokázali, že ak je správne vyplnená súvislá oblasť tabuľky `c` od pozície $(0,0)$ po nejaké $(n^*,k^*)$, tak tieto hodnoty v nej zostanú v danom vykonávaní už stále, čiže

$$\langle\forall n\forall k:(n,k)\in I\wedge(n,k)<(n^*,k^*)::\texttt{c}[n,k]=C(n,k)\rangle\ \text{is stable}.$$

Rovnaké by sme ale mohli tvrdiť aj pri prechádzaní `c` od $(0,0)$ po nejaké $(n^*,k^*)$ po stĺpcoch zľava doprava a zhora nadol:

$$\langle\forall n\forall k:(n,k)\in I\wedge(k,n)<(k^*,n^*)::\texttt{c}[n,k]=C(n,k)\rangle\ \text{is stable}.$$

[5] Vlastnosti typu $\texttt{true}\ \text{unless}\ q$ nám síce vo všeobecnosti žiadnu ďalšiu informáciu o danom programe nedávajú, lebo `true` platí stále, čiže $q$ môže byť aj ľubovoľná nepravda, no tieto konkrétne vlastnosti uvádzame pre účely ďalších podsekcií (najmä o ensures).

**ensures.** Opäť by sme mohli o niečo zosilniť priamo príkazy z programu, lebo každý z nich sa určite raz vykoná:

$$\begin{gathered}\texttt{true}\ \text{ensures}\ \texttt{c}[n,0]=1,\\ \texttt{true}\ \text{ensures}\ \texttt{c}[n,n]=1,\\ \texttt{true}\ \text{ensures}\ \texttt{c}[n,k]=\texttt{c}[n-1,k-1]+\texttt{c}[n-1,k].\end{gathered}$$

Špeciálne by sme sa mohli zamerať aj na prvok $\texttt{c}[2,1]$, ktorý ak existuje a $\texttt{c}[1,0]$ a $\texttt{c}[1,1]$ sú už správne, sa raz určite po aplikovaní príkazu $\texttt{c}[2,1]$ `:=` $\texttt{c}[1,0]$ `+` $\texttt{c}[1,1]$ správnym stane:

$$(\texttt{c}[1,0]=1\wedge\texttt{c}[1,1]=1)\ \text{ensures}\ \texttt{c}[2,1]=2.$$

**leads-to.** Po uplatnení pravidla, že ensures implikuje leads-to, z vlastností ensures z predošlej sekcie získame:

$$\begin{gathered}\texttt{true}\to\texttt{c}[n,0]=1,\\ \texttt{true}\to\texttt{c}[n,n]=1,\\ \texttt{true}\to\texttt{c}[n,k]=\texttt{c}[n-1,k-1]+\texttt{c}[n-1,k],\\ (\texttt{c}[1,0]=1\wedge\texttt{c}[1,1]=1)\to\texttt{c}[2,1]=2.\end{gathered}$$

Keďže vieme, že každé vykonávanie dosiahne pevný bod, v ktorom je `c` Pascalovým trojuholníkom, môžeme si zobrať aj ľubovoľnú vlastnosť tohto trojuholníka. Napríklad, že $\texttt{c}[n,k]=C(n,k)$:

$$\texttt{true}\to\texttt{c}[n,k]=C(n,k),$$

alebo že súčet prvkov v každom riadku je $2^n$, kde $n$ je index daného riadka:

$$\texttt{true}\to\langle+k:0\le k\le n::\texttt{c}[n,k]\rangle=2^n.$$