Programovanie paralélnych a distribuovaných systémov
Domáca úloha 7

Autor: Marián Kravec

Úloha 1

Upravený program bude vyzerať následovne:

```
Program
declare sum, v : integer, seq : sequence of integer
always sum = ⟨ + i : i ∈ seq :: A [i] ⟩ + A [v]
initially seq, v = null, 0
assign
seq, v := seq;v, v+1 if sum ≤ M1
        ~ pop(seq), top(seq) +1 if sum ≥ M2
end
```

Uvažujme, že M1 a M2 sú tak=, že riešenie môže existovať, čiže existuje M0 pre ktoré platí M1<M0<M2. Ak toto platí a my pracujeme s celými číslami tak M2-1 je najvyššia správna hodnota, preto túto pridáme na koniec zoznamu aby sme mali istotu, že algoritmus dá riešenie.

Safety -> ak tam existuje riešenie (neobsahujúce posledný prvok) tak ho nájde -> algoritmus prechádza podmnožiny v lexikografickom poradí, vďaka tomu množina obsahujúca iba posledný prvok bude skontrolovaná až ako posledná, čiže kým ju dosiahne prejde všetky ostatné podmnožiny, ak by medzi nimi bola podmnožina spĺňajúcu podmienku tak ani jeden príkaz by nebol aplikovateľný a tým pádom by dosiahol program pevný bod

Progress -> algoritmus vždy dosiahne pevný bod -> toto zabezpečujeme pridaním M2-1 na koniec zoznamu, keďže tento algoritmus vyskúša všetky podmnožiny určite raz vyskúša aj iba tento prvok

Úloha 2

Program upravíme takto:

```
Program
assign
⟨|| i : 1 ≤ i ≤ N :: g[i,i] = 0 ⟩
||
⟨|| i,j : 1 ≤ i < j ≤ N :: g[i,j] =
        ⟨min k : i ≤ k < j :: g[i,k] + g[k+1,j] + r[i-1] x r[k] x r[j] ⟩
⟩
end
```

Keďže chceme aby vykonal všetky úpravy naraz, máme istotu, že ak pôvodný program dával správny výsledok tak ho dá aj tento.

Úloha 3

Safety -> žiadnu cestu nepreskočí -> algoritmus postupne prehľadávaním do hĺbky prechádza všetky cesty, keďže kontroluje aby či už vrchol videl nedostane sa do cyklov, vďaka tomu vieme, že všetky cesty budú konečné ale zároveň ich určite skontroluje
Progress -> ak cesta existuje tak ju nájde a bude to pevný bod -> keďže prejde všetky cesty tak ak existuje cesta do destinácie tak v nejakom bode programu bude platiť $u = dest$ v tejto chvíli algoritmus nevie aplikovať žiaden príkaz vďaka čomu ide o pevný bod

Úloha 4

Ak platí $x = n + 1$, tak sa raz vykoná príkaz $kr := x$ z čoho vyplýva, že bude platiť $kr = n + 1$

Úloha 6

(na prednáške sme podľa mňa na tabuli použili príkaz duplicita aj v prípadoch kde sa úplne nemal použiť keďže on by nemal vedieť pridávať na ľavú stranu nový prvok iba posledný prvok pridať na koniec pravej strany)
null loss null
a loss null (strata)
a loss a (duplicita)
a loss aa (duplicita)
a loss aaa (duplicita)
ab loss aaa (strata)
abc loss aaa (strata)
abc loss aaac (duplicita)
abc loss aaacc (duplicita)

Úloha 5

Po prepísaní vyzerá takto, proste pridáme modulo 2:

```
Program P3
declare ks, kr: integer
initially
ks, kr = 1, 0
□ cs, cr, acks, ackr = null, null, null, null
□ mr = null
assign
        ackr := ackr;kr
□ ks := (ks + 1) mod 2 if acks ≠ null ∧ ks = head(acks)
        || acks := tail(acks) if acks ≠ null
□ cs := cs;(ks, ms[ks])
□ kr, mr := (kr + 1) mod 2, mr;head(cr).val if cr ≠ null ∧ kr ≠ head(cr).dex
        || cr := tail(cr)
end
```