Programovanie paralélnych a distribuovaných systémov
Domáca úloha 1

Autor: Marián Kravec

Časť 1

Chceme dokázať, že program P3

```
Program P3
        initially r = 0
        assign r := max{ f(r), g(r), h(r) }
end{P3}
```

funguje, čiže spĺňa dve podmienky a to:

1. Neurobí nič zlého – nepreskočí nejaký voľný spoločný termín“
2. Urobí nakoniec niečo dobre – nájde spoločný termín

Poďme to riešiť postupne:

1. Chceme ukázať, že žiaden zo spoločných termínov (čiže ani najmenší) nebude preskočený. Dokážeme to sporom. Vznikajú dva možné prípady.
   - r túto hodnotu nadobudne ale neostane na nej, v takomto prípade vo chvíli keď túto hodnotu nadobudne bude podľa počiatočných podmienok platiť $f(r) = r$, $g(r) = r$, $h(r) = r$ a z toho vyplýva, že $max(f(r), g(r), h(r)) = r$ čo v SPORE s predpokladom, že na tejto hodnote neostane
   - r túto hodnotu preskočí, keďže r nikdy neklesá ak by táto situácia nastala muselo by existovať r_1(krok pred preskočením) a r_2(krok po preskočení) pričom pre preskočenú hodnotu spoločného termínu $r*$ platí $r_1 < r* < r_2$ a zároveň
     $max(f(r_1), g(r_1), h(r_1)) = r_2$ avšak keďže predpokladáme, že naše funkcie vždy vrátia najbližší voľný termín (a $r*$ je voľný pre všetkých) takže $f(r_1) \leq r*$, $g(r_1) \leq r*$, $h(r_1) \leq r*$ z čoho vyplýva $max(f(r_1), g(r_1), h(r_1)) \leq r*$ z čoho vyplýva SPOR (keďže r_2 nemôže byť väčšie $r*$)
2. Potrebujeme ukázať, že tento kód nájde spoločný termín, keďže všetky funkcie kontrolujúce či je termín voľný vždy vráti hodnotu väčšiu nanajvýš rovnú vstupnej hodnote tak aj ich maximum nikdy neklesne, zároveň platí, že ak aspoň jednému termín nevyhovuje tak vráti väčšiu hodnotu z čoho vyplynie, že hodnota r bude rásť kým nenastane hodnota vyhovujúca všetkým a keďže pracujeme s prirodzenými číslami a predpokladáme, že takáto hodnota existuje raz ju tento program dosiahne

Časť 2

Chceme dokázať, že program sort1

```
Program sort1
        assign
        ⟨□j: 0 ≤ j < N ::
                A[j], A[j + 1] := A[j + 1], A[j] if A[j] > A[j + 1]⟩
end{sort1}
```

naozaj zoradí zoznam A.
Znovu chceme ukázať splnenie dvoch podmienok:

1. V žiadnom kroku nebude zoznam menej utriedený ako v predchádzajúcom
2. Od určitého bodu bude zoznam utriedený (a už sa nebude meniť)

Poďme tieto podmienky dokázať:

1. Uvažujme, že utriedenosť zoznamu budeme počítať ako počet dvojíc ktoré sú v nesprávnom poradí (väčšia hodnota má menší index) pričom platí, že zoradený zoznam má hodnotu 0. Tento program vymení hodnoty v prípade, že väčšia hodnota je pred menšou, čím odstráni jednu zle usporiadanú dvojicu čiže zníži počet zle usporiadaných dvojíc, z čoho vyplýva, že každým krokom je zoznam viac a viac utriedený
2. Kým je zoznam neutriedený existuje aspoň jedna pozícia kde väčšie číslo je pred menším, keďže prehodenie každej dvojice je kontrolované nekonečne veľa krát, táto dvojica bude určite niekedy vymenená. Keďže takýmto prehodením sa počet možných takýchto dvojíc zníži (je ich nanajvýš toľko koľko je celkový počet neutriedených dvojíc o ktorých sme ukázali, že vymenením sa zníži) ale kým je zoznam neutriedený neexistuje, tieto výmeny sa budú diať kým už nebude 0 dvojíc v zlom poradí čiže je zoznam utriedený