Linda Jurkasová

1. Dokážte, že x = n + 1 → kr = n + 1 (str. 240)

I1 $invariant\ y\ \leq kr\ \leq x\ \leq ks\ \leq y+1$

Def. 1 $p\ ensures\ q\ \equiv p\ unless\ q\ \wedge \langle \exists s : s\ in\ F\ :: \{p\ \wedge\ \neg q\}\ s\ \{q\}\rangle$

Def. 2 $p\ ensures\ q\ /\ p\ \rightarrow\ q$

x = n + 1 → kr = n + 1

x = n + 1 ensures kr = n + 1 — z def 2

Dokážeme pomocou def 1 pre ensures

$x = n+1\ \wedge\ kr\ \neq n+1\ \Rightarrow\ y = kr = n\ \wedge ks = x = y+1 = n+1$ — z I1

$\{x\ =\ n\ +\ 1\}\ kr \coloneqq x\ \{\ kr = n+1\ \}$ — Z programu

$\{x\ =\ n\ +\ 1\ \wedge\ kr\ \neq n+1\ \}\ kr \coloneqq x\ \{\ x = n+1\ \vee kr = n+1\ \}$ — def unless

2. Formálne odvoďte abc loss aaacc (str. 245)

(null loss null)	
(null loss null) ⇒ (null;a loss null)	Pravidlo straty
(null;a loss null) ⇒ (null;a loss null;a)	Pravidlo duplikacie
(null;a loss null;a) ⇒ (null;a loss null;a;a)	Duplikacia
(null;a loss null;a;a) ⇒ (null;a loss null;a;a;a)	Duplikacia
(null;a loss null;a;a;a) ⇒ (null;a;b loss null;a;a;a)	Strata
(null;a;b loss null;a;a;a) ⇒ (null;a;b;c loss null;a;a;a)	Strata
(null;a;b;c loss null;a;a;a) ⇒(null;a;b;c loss null;a;a;a;c)	Duplikacia
(null;a;b;c loss null;a;a;a;c) ⇒(null;a;b;c loss null;a;a;a;c;c)	Duplikacia
(abc loss aaacc)	odstranenie null

3. Program FC na str. 248. Bude program fungovať ak všade odstránime premennú b. Ak áno, prečo, ak nie, prečo nie?.

Odstránením premennej b, môže nastať situácia, že všetky správy sa budú strácať.

Ak sa totiž pozrieme na príkaz FC, ktorý stráca správy tak sa vykoná pod podmienkou, že cs nie je null. A teda ak máme v cs nekonečnú postupnosť, tak nikdy nenastane cs = null a teda program bude mať možnosť správy strácať nekonečne veľa krát.

A teda program nebude spĺnať podmienku, že strácať správy môže iba konečne krát. To isté platí pre duplikáciu.

Zavedením premennej b sme teda zabezpečili, že sa duplikovanie a strácanie vykoná iba konečne veľa krát.

Linda Jurkasová

4. Program P3 na str. 250 prepíšte do podoby Alternating Bit Protocolu.

Program P3

declare ks, kr, ix: integer

initially

ks, kr = 1, 0 ▯

cs, cr, acks, ackr = null, null, null, null ▯

mr = null ▯

$ix = 1$

assign

ackr := ackr;kr ▯

ks := (ks + 1) mod 2 if acks ≠ null ∧ ks = head(acks) || acks := tail(acks) if acks ≠ null || ix := ix + 1 ▯

cs := cs;(ks, ms[ix]) ▯

kr, mr := (kr + 1) mod 2 , mr;head(cr).val if cr ≠ null ∧ kr ≠ head(cr).dex || cr := tail(cr)

end