1.úloha

$x = n + 1 \rightarrow kr = n + 1$
Toto zadanie si prepíšeme na:
$x.dex = n + 1 \rightarrow kr = n + 1$

Je to z toho dôvodu, že do x v programe na strane 240 už priradzujeme dvojicu, pričom poradie správy, teda nejaké n, je vyjadrené v x.dex a v druhej časti x, teda v x.val je iba obsah správy. Túto správu ale získame z ms práve na základe x.dex.

I1 z prednášky si pre program P2 zmeníme nasledovne:
$y \leq kr \leq x.dex \leq ks \leq y + 1$ (I1)

V programe máme príkaz (posledný v poradí), ktorý nám zaručí, že kr sa zmení na hodnotu x.dex, ale musíme to dokázať:

x.dex = n + 1 ensures kr = n + 1 - je špeciálny prípad leads to a vyplýva z nasledovného:

$x.dex = n + 1 \wedge kr \neq n + 1 \Rightarrow y = kr \wedge x.dex = ks$ {z (I1)}

{x.dex = n + 1} po vykonaní príkazu kr, mr := x.dex, mr;x.val if kr ≠ x.dex {kr = n + 1}
(Vidíme, že sa nám do kr zapíše hodnota x.dex, pretože podmienka kr ≠ x.dex určite platí, keďže sme v predchádzajúcom riadku tvrdili, že x = n + 1 a kr ≠ n + 1, a teda určite ide o rôzne hodnoty. Premenná mr sa rozšíri o správu x.val.)

musíme ešte ukázať unless podmienku:
$\{x.dex = n + 1 \wedge kr \neq n+1\}$ s $\{x.dex = n + 1 \vee kr = n+1\}$
Platí to, pretože nech sa príkazy vykonajú v hocijakom poradí, tak na to, aby neplatila táto unless podmienka by musel nastať stav $\{x.dex \neq n + 1 \wedge kr \neq n+1\}$.
Skúsime teda:
zmeniť x.dex tak, aby platilo $x.dex \neq n + 1$, ale zároveň nezmeniť platnosť $kr \neq n+1$, pretože to chceme zachovať.
Hodnota x.dex sa môže zmeniť iba príkazom x := (ks, ms[ks]). Zároveň vieme, že ks = y + 1 a y = kr. Z I1 vieme, že ak x.dex = n + 1, potom jediné prípustné je, aby kr = n. No ale v tom prípade $y = kr = n \Rightarrow ks = y + 1 = n + 1 \Rightarrow x.dex = n + 1$. Hodnotu x.dex teda nevieme zmeniť bez toho, aby sme zmenili kr, čo ale zmeniť nechceme. Ak by sme zmenili kr, vieme už, že kr musí byť n, tak potom by sa určite kr zmenilo na kr = n + 1, pretože posledný príkaz priradzuje do kr := x.dex, no a v x.dex je hodnota n + 1. V tom prípade by sa teda kr = n + 1 a opäť by sme „pokazený“ stav nedosiahli. Platí teda unless podmienka.

2.úloha

abc loss aaacc

Ukážeme, že ak začneme s prázdnou postupnosťou u = null, v = null, tak sa vieme iba použitím (null loss null) a pravidiel pre stratu a duplikáciu zo str. 245 dopracovať k (abc loss aaacc). Samozrejme musíme dodržať pravidlo, že sa správy prenášajú v nezmenenom poradí (čiže najprv a, potom b a nakoniec c).

null označíme ako _ (kvôli lepšej čitateľnosti, ak postupnosť pred bodkočiarkou obsahuje aspoň 1 znak, tak vynecháme už null)

strata – S, duplikácia – D
prechod cez prázdnu šípku popisuje rovnaké postupnosti, iba bez bodkočiarky (pre u alebo v)

(_ loss _) $\overset{S}{\Rightarrow}$ (_;a loss _) $\overset{D}{\Rightarrow}$

(_;a loss _;a) => (_;a loss a) $\overset{D}{\Rightarrow}$

(_;a loss a;a) => (_;a loss aa) $\overset{D}{\Rightarrow}$

(_;a loss aa;a) => (a loss aaa) $\overset{S}{\Rightarrow}$

(a;b loss aaa) => (ab loss aaa) $\overset{S}{\Rightarrow}$

(ab;c loss aaa) $\overset{D}{\Rightarrow}$ (ab;c loss aaa;c) =>

(ab;c loss aaac) $\overset{D}{\Rightarrow}$ (ab;c loss aaac;c) =>

(abc loss aaacc)

Ukázali sme, že je možné skonštruovať pomocou pravidiel pre stratu a duplikáciu (abc loss aaacc) a teda je možné, aby pri vstupe abc sme ako výstup chybného kanála dostali aaacc.

3.úloha

Myslím si, že jediný problém, respektíve zmena oproti tomu pôvodnému programu, ktorá by teoreticky mohla nastať je, že by sa tretí príkaz vždy vykonal v takom stave, kedy by v cs nebolo nič, pretože by sa správy postrácali pomocou prvého príkazu a zároveň by sa ešte nestihlo do cs nič poslať, teda by cs = null. Predtým to bolo vyriešené tak, že keď sa od istého momentu „stoplo“ strácanie aj duplikovanie správ, tak aj keby v tom momente nič nebolo v cs, tak určite raz by sa tam poslalo a zároveň ak by bolo b true, tak by sa to už stratiť nemohlo. Neviem, do akej miery je pravdepodobné , že sa také niečo môže stať, keďže neviem, či môže nastať situácia, že by sa do cs správy posielali pomalšie, ale ak nemáme zaručené poradie príkazov, tak sa asi takáto situácia nevylučuje.
Síce v tomto programe môže dochádzať k nekonečnému strácaniu alebo duplikovaniu správ, tak v podstate v prednáške na str. 247 bolo takisto povedané, že pôvodný program simulujúci FC neobsahuje predpoklad, že len konečne veľa chýb možno vykonať za sebou, takže neviem, či je toto brané za chybu alebo nie.
Pôvodne sme pomocou premennej b iba od istého momentu „zastavili“ možnosť strácať alebo duplikovať správy, pričom sa muselo počkať na správny transfer, teda zároveň zmenu aj cs aj cr, pričom sa vtedy nastavila premenná b späť na false. Potom mohol faulty channel opäť ľubovoľne strácať, duplikovať správy, až kým sa nenastavila premenná b na true. Mohla však nastať situácia, že by sa správa ľubovoľne veľakrát stratila a duplikovala v rôznom poradí, a následne sa vykonal správny transfer správy. Nikde nie je zaručené, že by sa b muselo nastaviť na true ešte predtým, než by sa vykonal tretí príkaz v poradí v programe FC. Táto situácia zodpovedá tomu, do akej podoby by sme program dostali, ak by sme b odstránili.

4.úloha

Program P3
declare ks, kr, msgNum : integer

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

V programe by ostalo všetko rovnaké, s tým rozdielom, že v kr aj ks by bol iba bit 0 alebo 1. Podľa toho by sme vedeli, či už môže nastať posielanie ďalšej, ak by sa tento bit zmenil. Do cs sa však posielala dvojica (ks, ms[ks]), čo predtým sedelo, lebo ks vyjadrovalo poradie správy. Teraz by to však vždy bola iba 0 a 1 a teda už by sme z ms[ks] nedostali pravdepodobne to, čo chceme. Preto som zaviedla ešte premennú msgNum, ktorá je „počítadlom“ a určuje poradie správy, ktorá sa posiela. Hodnota msgNum sa zvýši zároveň s príkazom, ktorý mení ks.