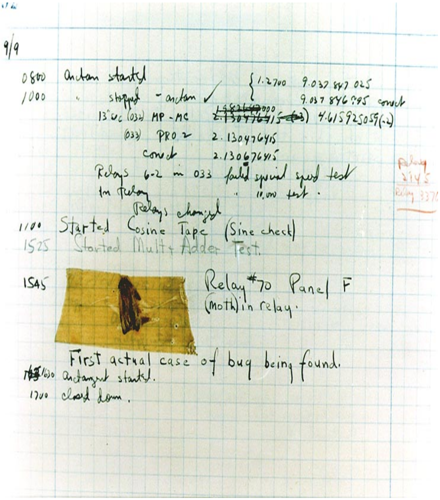
Korektnosť algoritmov

IB110

Prečo?



- 1962, Mariner1 - štart rakety
- 1981, Kanada - informácia o volebných preferenciách
- 1985-87, Therac-25 - nesprávne dávky röntgenového žiarenia
- problém Y2K
- http://www.devtopics.com/20-famous-software-disasters/

Typy chýb

- syntaktické chyby
  - *until* / *untli*
  - `for (k=0;k<101){ sum = sum + k }` *versus*
    `for (k=0;k<101;k=k+1){ sum = sum + k }`
- sémantické chyby
  - *výsledná hodnota premennej cyklu*
  - `for (k=0;k=k+1;k<101){ sum = sum + k }` *versus*
    `for (k=0;k<101;k=k+1){ sum = sum + k }`
- logické chyby
  pre daný text zisti, koľko viet obsahuje slovo kniha
  - *koniec vety indikuje výskyt symbolov ". " (bodka, medzera)*
  - *koniec vety indikuje výskyt symbolu "." (bodka)*

Počítače nerobia chyby

Testovanie a ladenie

- syntaktické chyby, run-time chyby
- testovanie, testovacie sady
- ladenie
- nezaručujú bezchybnosť algoritmu

Čiastočná a úplná korektnosť

Špecifikácia algoritmického problému:
1. určenie množiny vstupných inštancií
2. určenie vzťahu medzi vstupnými inštanciami a požadovaným výstupom.

- čiastočná korektnosť: pre každú vstupnú inštanciu X platí, že ak výpočet algoritmu na X skončí, tak výstup má požadovanú vlastnosť
- konečnosť: výpočet skončí pre každú vstupnú inštanciu
- úplná koreknosť: čiastočná korektnosť + konečnosť

Dôkaz korektnosti

invarianty

- kontrolné body programu
- invariant = tvrdenie, ktoré platí pri každom priechode kontrolným bodom
- čiastočná korektnosť

konvergencia

- s kontrolnými bodmi asociujeme kvantitatívnu vlastnosť
- pri každom priechode kontrolným bodom sa hodnota kvantitatívnej vlastnostni znižuje
- hodnota kvantitatívnej vlastnosi nesmie prekročiť dolnú hranicu
- konečnosť výpočtu

Príklad - zrkadlový obraz

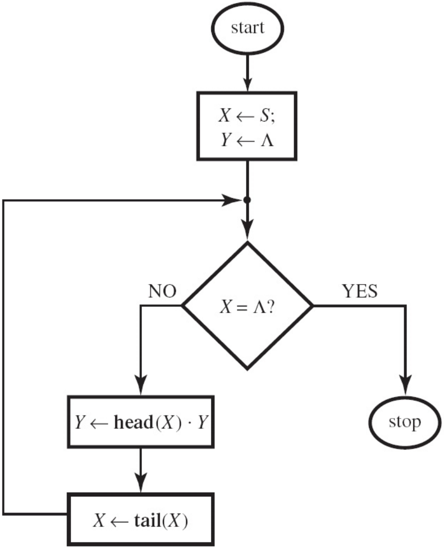
Vstup: reťazec S

Výstup: symboly reťazca S v obrátenom poradí

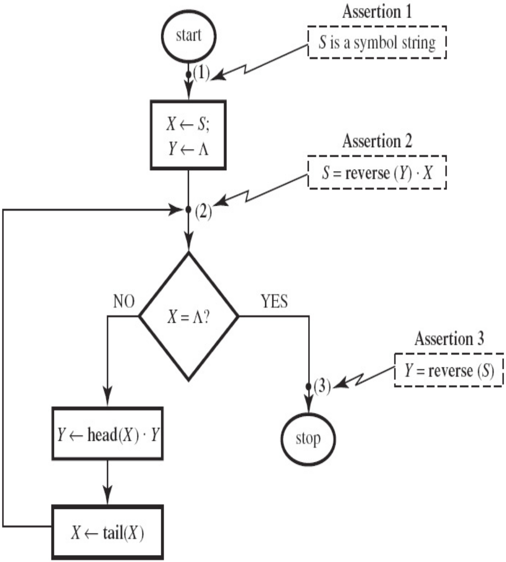
Notácia

- **reverse**("fakulta") = "'atlukaf"
- **head**("fakulta") = "f"
- **tail**("fakulta") = "akulta"
- symbol Λ označuje prázdny reťazec (reťazec neobsahuje žiaden symbol)
- symbol $\cdot$ označuje zreťazenie (spojenie) dvoch reťazcov

Zrkadlový obraz — algoritmus



Zrkadlový obraz — kontrolné body a invarianty

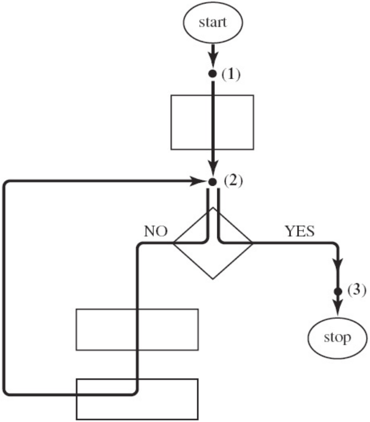


- Invariant 1
  vstupná podmienka
- Invariant 2
  $S = \textbf{reverse}(Y) \cdot X$
  charakterizuje výpočet
- Invariant 3
  $Y = \textbf{reverse}(S)$
  požadovaný vzťah medzi vstupom S a výstupom Y

Zrkadlový obraz — platnosť invariantov

dokazujeme, že pre každý platný vstup: ak výpočet dosiahne kontrolný bod, tak tvrdenie je pravdivé

v akom poradí sa prechádzajú kontrolné body?



$$1 \longrightarrow 2 \longrightarrow 2 \longrightarrow \cdots 2 \longrightarrow 3$$

Zrkadlový obraz — platnosť invariantov

$1 \longrightarrow 2$ pre každý reťazec S po vykonaní príkazov $X \leftarrow S, Y \leftarrow \Lambda$ platí rovnosť $S = \textbf{reverse}(Y) \cdot X$

$2 \longrightarrow 3$ ak $S = \textbf{reverse}(Y) \cdot X$ a $X = \Lambda$, tak $Y = \textbf{reverse}(S)$

$2 \longrightarrow 2$ ak $S = \textbf{reverse}(Y) \cdot X$ a $X \neq \Lambda$, tak po vykonaní príkazov $Y \leftarrow \textbf{head}(X) \cdot Y;\ X \leftarrow \textbf{tail}(X)$ platí znovu tá istá rovnosť pre nové hodnoty premenných X a Y

dokázali sme čiastočnú korektnosť

Zrkadlový obraz — konečnosť

- výpočet algoritmu je nekonečný práve ak prechádza kontrolným bodom 2 nekonečne veľa krát
- s kontrolným bodom 2 asociujeme kvantitatívnu vlastnosť (tzv. *konvergent*) a ukážeme, že jej hodnota klesá a pritom je zdola ohraničená
- konvergentom pre kontrolný bod 2 je dĺžka reťazca X
- pri každom priechode kontrolným bodom 2 dĺžka reťazca X klesne o 1
- ak dĺžka X klesne na 0 (X je prázdny reťazec), tak výpočet neprechádza cyklom a nenavštívi kontrolný bod 2

dokázali sme konečnosť

Zrkadlový obraz — korektnosť

korektnosť = čiastočná korektnosť + konečnosť

Príklad - Euklidov algoritmus

Vstup dve kladné celé čísla X a Y

Výstup najväčší spoločný deliteľ Z čísel X a Y

spoločný deliteľ Z Z delí X a Z delí Y (celočíselne)

najväčší deliteľ pre každé číslo $U > Z$, buď U nedelí X alebo U nedelí Y

Euklidov algoritmus — implementácia

function *Euclid*(X, Y)
$V \leftarrow X$
$W \leftarrow Y$
while $V \neq W$ **do**
  if $V > W$ **then** $V \leftarrow V - W$ **fi**
  if $V < W$ **then** $V \leftarrow W - V$ **fi**
od
return (V)

Invariant 1 V a W sú násobkom Z
Invariant 2 $V \geq Z$ a $W \geq Z$
Invariant 3 neexistuje väčší spoločný deliteľ čísel V a W než číslo Z
všetky invariatny platia v každom bode výpočtu

Euklidov algoritmus — čiastočná korektnosť

Invariant 1 V a W sú násobkom Z
Invariant 2 $V \geq Z$ a $W \geq Z$
Invariant 3 neexistuje väčší spoločný deliteľ čísel V a W než číslo Z

Inicializácia $V \leftarrow X, W \leftarrow Y$

- invarianty 1, 2, 3 sa priradením neporušia

IF príkaz **if** $V > W$ **then** $V \leftarrow V - W$ **fi**

- Fakt Ak $V > W$, tak dvojice čísel V, W a $V - W, W$ majú rovnakých spoločných deliteľov
- ak Z delí V, W a $V > W$, tak $V - W > 0$ a $V - W \geq Z$
- invarianty 1, 2, 3 zostávajú zachované

IF príkaz **if** $W > V$ **then** $W \leftarrow W - V$ **fi**

- symetricky

Euklidov algoritmus — čiastočná korektnosť

Invariant 1 V a W sú násobkom Z
Invariant 2 $V \geq Z$ a $W \geq Z$
Invariant 3 neexistuje väčší spoločný deliteľ čísel V a W než číslo Z

---

while príkaz

- všetky invarianty zostávajú v platnosti po prevedení jednotlivých príkazov cyklu
- cyklus končí keď $V = W$
- V je najväčším spoločným deliteľom V, W
- $V = Z$

čiastočná korektnosť

Euklidov algoritmus — konečnosť

- výpočet je nekonečný práve ak **while** príkaz sa vykoná nekonečne veľa krát
- konvergentom **while** cyklu je súčet $V + W$
- pri každom vstupe do tela cyklu je $V \geq Z > 0, \;\; W \geq Z > 0$ a $V \neq W$
- pri vykonaní tela cyklu sa odčíta celé kladné číslo buď od V alebo od W
- suma $V + W$ sa pri každom priechode cyklom zníži aspoň o 1
- na začiatku je $V + W = X + Y$ a preto sa cyklus vykoná nanajvýš $X + Y$ krát

konečnosť

Príklad - triedenie vkladaním

$Insertion - Sort(A)$
for $j \leftarrow 2$ **to** $length[A]$ **do**
  $key \leftarrow A[j]$
  $i \leftarrow j - 1$
  while $i > 0 \wedge A[i] > key$ **do** $A[i+1] \leftarrow A[i]$
    $i \leftarrow i - 1$ **od**
  $A[i+1] \leftarrow key$
od

Invariant na začiatku iterácie **for** cyklu obsahuje $A[1 \ldots j-1]$ tie isté prvky, ako obsahovalo na týchto pozíciách pole A na začiatku výpočtu, ale utriedené od najmenšieho po najväčší

Triedenie vkladaním - čiastočná korektnosť a konečnosť

Invariant na začiatku iterácie **for** cyklu obsahuje $A[1 \ldots j-1]$ tie isté prvky, ako obsahovalo na týchto pozíciách pole A na začiatku výpočtu, ale utriedené od najmenšieho po najväčší

Inicializácia tvrdenie platí na začiatku výpočtu ($j = 2$, postupnosť $A[1]$ obsahuje jediný prvok a je utriedená)

FOR cyklus v tele cyklu sa hodnoty $A[j-1], A[j-2], A[j-3], \ldots$ posúvajú o jednu pozíciu doprava až kým sa nenájde vhodná pozícia pre $A[j]$

Ukončenie **for** cyklus sa ukončí keď $j = n+1$. Substitúciou $n+1$ za j dostávame, že pole $A[+ \ldots n$ obsahuje tie isté prvky, ako na začiatku výpočtu, ale utriedené.

Konečnosť **for** cyklus nemení hodnotu riadiacej premennej cyklu

Formálna verifikácia

- interaktívne dokazovanie
- dokazovanie formálnym odovedním (*theorem proving*)
- overovanie modelu (*model checking*)