Wstęp do informatyki

Wykład 7

Uniwersytet Wrocławski
Instytut Informatyki

Temat wykładu

Poprawność programów

- co oznacza?
- jak ją uzasadnić? sprawdzić?
- przykłady: potęgowanie, selekcja, bąbelki,…

Nowe narzędzia do sortowania i ich analiza:

- scalanie
- podział

Jak sprawdzić poprawność programu?

Co oznacza „poprawność”?

- zgodność ze specyfikacją… (?)

Sposoby weryfikacji:

- testowanie? zawsze czegoś nie da się przewidzieć…
- formalna analiza - dowód?

  żmudne, często trudniejsze od napisania programu
- automatyczne narzędzia dowodzenia?

  możliwe tylko w niektórych przypadkach (uniwersalne narzędzia nie istnieją i nie jest możliwe ich stworzenie!)

Zgodność ze specyfikacją…

Stan obliczeń = wartości zmiennych.

Specyfikacja formalna:

- warunek początkowy A
- program P
- warunek końcowy B

gdzie A, B to formuły logiczne, a zmienne programu to zmienne wolne w A, B

Zgodność ze specyfikacją…

UWAGA!

Dla uproszczenia:

- ignorujemy instrukcje wejścia/wyjścia
- analizujemy tylko

  stan obliczeń = wartości zmiennych
- ignorujemy kwestie typów zmiennych, zakresów, deklaracji, …

Zgodność ze specyfikacją…

Program P jest częściowo poprawny ze względu na specyfikację

{A} P {B}

gdy spełniona jest zależność:

jeśli przed wykonaniem P stan obliczeń spełnia A, to po wykonaniu P stan obliczeń spełnia B.

Mówimy wtedy skrótowo, że zachodzi:

{A} P {B}

({A} P {B} to trójka Hoare'a)

Częściowa poprawność

Zachodzi

$$\{x>0\}\ y \leftarrow x+1\ \{y>1\}$$

gdyż:

jeśli przed wykonaniem P= $y \leftarrow x+1$ stan obliczeń spełnia $\{x>0\}$, to po wykonaniu P stan obliczeń spełnia $\{y>1\}$.

UZASADNIENIE

Po wykonaniu obliczeń mamy: $y=x+1>0+1=1$

Częściowa poprawność

Zachodzi

$\{true\}$ if (a>b) $c \leftarrow a$; else $c \leftarrow b$ $\{c \geq a$ oraz $c \geq b\}$

gdyż po wykonaniu P= if (a>b) $c \leftarrow a$; else $c \leftarrow b$ stan obliczeń spełnia $\{c \geq a$ oraz $c \geq b\}$ niezależnie od stanu obliczeń przed P.

UZASADNIENIE

Jeśli $a>b$, to c jest równe a po wykonaniu P, więc $c=a>b$.

Analogicznie gdy $a \leq b$.

Częściowa poprawność

Zachodzi

$$\{x>0\}\ y \leftarrow x+1;\ z \leftarrow y+1\ \{z>2\}$$

gdyż:

jeśli przed P= $y \leftarrow x+1;\ z \leftarrow y+1$ stan obliczeń spełnia $\{x>0\}$, to po wykonaniu P stan obliczeń spełnia $\{z>2\}$.

UZASADNIENIE

- Po wykonaniu $y \leftarrow x+1$ zachodzi $y=x+1>0+1=1$
- Po wykonaniu $z \leftarrow y+1$ zachodzi $z=y+1>1+1=2$

Częściowa poprawność

Zachodzi

$$\{x \neq 0\} \text{ while } (x!=0)\ x \leftarrow x-1\ \{x=0\}$$

gdyż:

jeśli przed P= while (x!=0) $x \leftarrow x-1$ stan obliczeń spełnia $\{x \neq 0\}$, to po wykonaniu P stan obliczeń spełnia $\{x=0\}$.

UZASADNIENIE

- Po wyjściu z pętli while nie jest spełniony warunek $x \neq 0$, zatem jest spełniony warunek $x=0$

Częściowa poprawność

Zachodzi

$$\{x \neq 0\}\ \text{while } (x!=0)\ x \leftarrow x-1\ \{x=0\}$$

lecz powyższe oznacza jedynie częściową poprawność:

- jeśli $P = \text{while } (x!=0)\ x \leftarrow x-1$ zakończy działanie, to $\{x=0\}$
- **nie** pokazaliśmy: P **zawsze** kończy działanie!

W naszym przykładzie:

dla $x<0$ program P nie kończy działania.

Poprawność

Poprawność względem

{A} P {B}

wymaga spełnienia dwóch warunków:

- częściowa poprawność {A} P {B}
  („wynik poprawny”, gdy P zakończy działanie)
- własność stopu:

  jeśli przed wykonaniem P zachodzi A, to P zawsze kończy działanie

Własność stopu – przykład

Własność stopu dla {A} P {B}:

jeśli przed wykonaniem P zachodzi A, to P zawsze kończy działanie

zachodzi własność stopu

P= while (x!=0) x $\leftarrow$ x-1

A= x – liczba **naturalna**

nie zachodzi własność stopu

P= while (x!=0) x $\leftarrow$ x-1

A= x – liczba **całkowita**

Częściowa poprawność… cd

Warunek N jest niezmiennikiem programu P względem warunku początkowego A, gdy:

$$\{N \wedge A\}\ P\ \{ N\}$$

czyli:

jeśli przed rozpoczęciem P jest spełniony warunek początkowy A i niezmiennik N, to po wykonaniu P nadal spełniony jest niezmiennik N.

Niezmiennik - przykład

Warunek $a+b=10$ jest niezmiennikiem programu

$a{+}{+};\ b{-}{-}$

względem warunku początkowego

$b>0$

gdyż:

$$\{a+b=10 \wedge b>0\}\ a{+}{+};\ b{-}{-}\ \{a+b=10\}$$

Niezmiennik pętli

Def. Warunek N jest niezmiennikiem pętli

while (A) P;

gdy N to niezmiennik P względem warunku początkowego A, czyli:

$$\{ N \wedge A \} \; P \; \{ N \}$$

INTUICJA

niezmiennik to warunek, którego prawdziwość nie zmienia się wskutek wykonania pętli.

Niezmiennik programu/pętli

Warunek $a+|b|=10$ jest niezmiennikiem programu a++; b – – względem warunku początkowego $b>0$ gdyż:

$$\{a+|b|=10 \wedge b>0\}$$

a++; b – –

$$\{a+|b|=10\}$$

Ale: $a+|b|=10$ nie jest niezmiennikiem programu a++; b – – **bez** warunku początkowego. Dlaczego?

Niezmiennik progr./pętli - przykład

Warunek $a+|b|=10$ jest niezmiennikiem pętli `while (b>0) {a++; b – –}` gdyż:

$$\{a+|b|=10 \wedge b>0\}\ a{+}{+};\ b{-}{-}\ \{a+|b|=10\}$$

Niezmiennik pętli - zastosowanie

Obserwacja.

Jeśli N jest niezmiennikiem pętli while (A) P to

$$\{ N \}\ \text{while (A) P}\ \{ N \wedge \neg A \}$$

INTUICJA (do zapamiętania)

Jeśli przed rozpoczęciem pętli spełniony jest niezmiennik N to po zakończeniu pętli (nadal) spełniony jest N oraz zaprzeczenie warunku kontynuacji pętli, czyli $\neg A$.

Niezmiennik pętli - przykład

Warunek $a+|b|=10$ jest niezmiennikiem pętli `while (b>0) {a++; b – –;}`, czyli :

$$\{a+|b|=10 \land b>0\}\ a{+}{+};\ b{-}{-}\ \{a+|b|=10\}$$

Zatem zachodzi:

$$\{a+|b|=10\}$$

`while (b>0) {a++; b – –; }`

$$\{a+|b|=10 \land \neg b>0\}$$

Niezmienniki a częściowa poprawność

Po co taka sformalizowana notacja:

- automatyczne dowodzenie poprawności (przedmiot: metody programowania i in.)

Praktyka dowodzenia (prostych) programów:

- „wymyśl” niezmienniki kluczowych pętli
- uzasadnij (mniej lub bardziej formalnie) ich poprawność
- uzasadnij, że niezmiennik jest spełniony przy wejściu do pętli
- skorzystaj z tego, że niezmiennik jest spełniony po wyjściu z pętli

Praktyka dowodzenia poprawności

Bardziej formalnie:

- podaj „asercje” (warunki określające stan zmiennych) zachodzące w kluczowych punktach programu;
- wykaż, że podane asercje zachodzą, wykorzystując metodę niezmienników.

Przykład

Program P:

```
x=a; y=b; rez=1;
while (y!=0) { rez = rez*x; y=y – 1; }
```

Jaki efekt działania programu?

Gdzie program umieszcza „wynik” obliczeń?

Jak to pokazać?

Przykład – częściowa poprawność

Program P:

$x \leftarrow a$; $y \leftarrow b$; $rez \leftarrow 1$;

while (y!=0) { $rez \leftarrow rez*x$; $y \leftarrow y - 1$; }

Warunek wstępny A:

b – liczba naturalna

Warunek końcowy B:

$rez = a^b$

czyli program wyznacza a^b

PYTANIE: czy {A} P {B}

Przykład – częściowa poprawność

Program P:

~~$x \leftarrow a$; $y \leftarrow b$; $rez \leftarrow 1$;~~

while (y!=0) { $rez \leftarrow rez*x$; $y \leftarrow y - 1$; }

Niezmiennik pętli (???):

$$rez * x^y = a^b$$

1. Czy to rzeczywiście niezmiennik?
2. Czy pomoże pokazać, że $rez = a^b$ po zakończeniu programu?

Przykład – ad. 2 (czy pomoże?)

Pętla L:

while (y!=0) { rez $\leftarrow$ rez*x; y $\leftarrow$ y – 1; }

Jeśli $N \equiv rez * x^y = a^b$ to niezmiennik L, który jest spełniony przed wejściem do L, wówczas po wyjściu z pętli L mamy:

$$rez * x^y = a^b \text{ ORAZ } \neg y!=0$$

czyli $rez * x^y = a^b$ ORAZ $y=0$

czyli $rez * x^0 = a^b$

czyli $rez = a^b$ …… !

Przykład – ad. 2

Pętla L:

while (y!=0) { rez $\leftarrow$ rez*x; y $\leftarrow$ y – 1; }

- $N \equiv rez * x^y = a^b$ to niezmiennik L (założenie)
- instrukcje przed pętlą: $x \leftarrow a$; $y \leftarrow b$; $rez \leftarrow 1$ powodują, że

$$rez * x^y = 1 * a^b = a^b$$

przed wejściem do pętli, czyli niezmiennik

$$N \equiv rez * x^y = a^b$$

jest spełniony przed wejściem do pętli.

Przykład – asercje

Program P:

$\{ b$ – liczba naturalna $\}$

$x \leftarrow a;\ y \leftarrow b;\ rez \leftarrow 1;$

$\{ rez * x^y = a^b \}$

while (y!=0) $\{ rez \leftarrow rez*x;\ y \leftarrow y - 1;\ \}$

$\{ rez * x^y = a^b$ oraz $y=0 \}$

Notacja – formuła z parametrami

Oznaczenie. Niech $F(x_1, \ldots, x_p)$ to formuła logiczna, w której występują (między innymi) zmienne wolne $x_1, \ldots, x_p$.

Wówczas $F(a_1, \ldots, a_p)$ to formuła F, w której wystąpienia zmiennej x_i zastępujemy przez a_i dla $i=1, \ldots, p$.

Przykład:

$N(rez, y) \equiv (y \geq 0) \wedge rez * x^y = a^b$

Wówczas:

$N(z, y+1) \equiv (y+1 \geq 0) \wedge z * x^{y+1} = a^b$

Przykład – ad. 1 (czy niezmiennik?)

Pętla L:

while (y!=0) { rez $\leftarrow$ rez*x; y $\leftarrow$ y – 1; }

TW. Warunek:

$$N(rez,y) \quad \equiv \quad rez * x^y = a^b$$

to niezmiennik pętli L.

DOWÓD.

Musimy pokazać:

{ N(rez,y) $\wedge$ (y≠0) } rez $\leftarrow$ rez*x; y $\leftarrow$ y – 1; {N(rez,y)}

Inaczej:

$\{(rez * x^y = a^b) \wedge (y \neq 0)\}$ rez $\leftarrow$ rez*x; y $\leftarrow$ y – 1; $\{rez * x^y = a^b\}$

Przykład – częściowa poprawność

Mamy pokazać:

$\{(y \neq 0) \wedge rez * x^y = a^b\}$ rez←rez*x; y←y – 1; $\{rez * x^y = a^b\}$

Niech:

rez’, y’ – wartości rez i y po wykonaniu

rez ← rez*x; y=y – 1;

Wówczas zakładając, że $\{rez * x^y = a^b\}$ musimy pokazać, że: $\{rez' * x^{y'} = a^b\}$

MAMY:

- rez’ = rez*x, y’=y – 1
- $rez' * x^{y'} = rez * x * x^{y'} = rez * x * x^{y-1} = rez * x^y = a^b$

$\square$

Przykład – przypomnienie

Program P:

$x \leftarrow a$; $y \leftarrow b$; $rez \leftarrow 1$;

while $(y>0)$ { $rez \leftarrow rez*x$; $y \leftarrow y - 1$; }

Warunek wstępny A:

b – liczba naturalna

Warunek końcowy B:

$rez = a^b$

Przykład – asercje

Program P:

{ b – liczba naturalna }

x=a; y=b; rez=1;

{ $rez * x^y = a^b$ }

while (y!=0) { rez = rez*x; y=y – 1; }

{ $rez * x^y = a^b$ oraz $y=0$ }

czyli { $rez = a^b$ }

Przykład – podsumowanie

Kroki dowodu częściowej poprawności programu

{A} P {B}:

- „wymyśliliśmy” niezmiennik N pętli L
- udowodniliśmy poprawność niezmiennika N
- pokazaliśmy, że niezmiennik N zachodzi przed wejściem do pętli L (o ile spełniony jest warunek początkowy programu A)
- sprawdziliśmy, że spełnienie N i zaprzeczenia warunku kontynuacji pętli L (czyli warunku y!=0) pociąga za sobą warunek końcowy B

Przykład – refleksja

Podany dowód częściowej poprawności:

- żmudny, sformalizowany ☹;
- najciekawsze – jaki wybrać (wymyślić?) niezmiennik pętli… tak, żeby było możliwe jego uzasadnienie i żeby pomógł w analizie programu

Co dalej (z częściową poprawnością):

- będziemy wybierać niezmienniki…
- … takie, które przekonają **nas**, że nasze programy są poprawne
- … a dowody będą mniej formalne.

Przykład – własność stopu

Pętla:

while (y!=0) { rez ← rez*x; y ← y – 1; }

Pyt.: Czy zawsze kończy działanie dla naturalnego y?

Odp.: TAK, ponieważ:

- jeśli y=0: zakończy natychmiast
- wpp: y będzie zmniejszane w każdym obrocie pętli, aż do osiągnięcia wartości zero – pętla również się zakończy (formalnie – np. indukcja).

Przykład – bubble

Program P:

```
i ← 0;
while (i<j–1)     {
   if (a[i]>a[i+1]) zamień(a[i],a[i+1]);
   i ← i+1;
}
```

Jaki cel tego programu?

Jaki efekt?

Przykład – bubble

Program P:

```
i ← 0;
while (i<j–1)    {
   if (a[i]>a[i+1]) zamień(a[i],a[i+1]);
   i ← i+1;
}
```

Warunek wstępny A:

$j>0$

Warunek końcowy B:

$a[j-1]=\max\{a[0],\ldots,a[j-1]\}$

Przykład – bubble

Pętla L:

~~$i \leftarrow 0$;~~

```
while (i<j–1)     {
   if (a[i]>a[i+1]) zamień(a[i],a[i+1]);
   i ← i+1;
}
```

Niezmiennik:

$a[i]=\max\{a[0],\ldots,a[i]\} \wedge 0 \le i \le j-1$

Przykład – bubble

Pętla L:

while (i<j–1) {

if (a[i]>a[i+1]) zamień(a[i],a[i+1]); i ← i+1;}

Niezmien.: $N(a,i) \equiv a[i]=\max\{a[0],\ldots,a[i]\} \wedge 0 \leq i \leq j-1$

Dowód poprawności niezmiennika (szkic):

Założenie: (N(a,i) oraz i<j – 1) przed wykonaniem:

if (a[i]>a[i+1]) zamień(a[i],a[i+1]); i ← i+1;.

Cel: pokazać, że N(a',i') po wykonaniu

if (a[i]>a[i+1]) zamień(a[i],a[i+1]); i ← i+1.

gdzie a', i' to wartości a oraz i po wykonaniu powyższej instrukcji

Przykład – bubble

Pętla L:

while $(i<j-1)$ {

if $(a[i]>a[i+1])$ zamień$(a[i],a[i+1])$; $i \leftarrow i+1$;}

Niezmiennik: $N(a,i) \equiv a[i]=\max\{a[0],\ldots,a[i]\} \wedge 0 \leq i \leq j-1$

Dowód poprawności niezmiennika:

Przypadek 1: $a[i]>a[i+1]$

- $a[i]$ był równy $\max\{a[0],\ldots,a[i]\}$, został zamieniony z elementem na pozycji $i+1$ i jest większy od elementu, z którym został zamieniony, więc mamy:

$$a'[i']=\max\{a'[0],\ldots,a'[i']\}$$

gdzie a' to tablica a po zamienie $a[i]$ z $a[i+1]$, $i'=i+1$.

Przykład – bubble

Pętla L:

while $(i<j-1)$ {

if $(a[i]>a[i+1])$ zamień$(a[i],a[i+1])$; $i \leftarrow i+1$;}

Niezmiennik: $N(a,i) \equiv a[i]=\max\{a[0],\ldots,a[i]\} \wedge 0 \leq i \leq j-1$

Dowód poprawności niezmiennika:

Przypadek 2: $\neg\, (a[i]>a[i+1])$

$a[i]$ był równy $\max\{a[0],\ldots,a[i]\}$, $a[i+1]$ jest od niego większy (lub równy), tablica a się nie zmienia więc:

$$a'[i']=\max\{a'[0],\ldots,a'[i']\}$$

gdzie a' jest taka sama jak a, $i'=i+1$.

Przykład – bubble

Pętla L:

```
while (i<j–1)     {
   if (a[i]>a[i+1]) zamień(a[i],a[i+1]); i=i+1;}
```

Pokazaliśmy:

$$N(a,i) \equiv a[i]=\max\{a[0],\ldots,a[i]\} \wedge 0 \leq i \leq j-1$$

to niezmiennik pętli L.

Przykład – bubble

Pętla L:

```
while (i<j–1)        {
    if (a[i]>a[i+1])
        zamień(a[i],a[i+1]);
    i ← i+1;}
```

Niezmiennik:

$N(a,i) \equiv$

$a[i] = \max\{a[0],\ldots,a[i]\} \wedge$

$0 \leq i \leq j-1.$

Sortowanie bąbelkowe:

```
j=n;
while (j>0) {
    i ← 0;
    while (i<j–1)        {
        if (a[i]>a[i+1])
            zamień(a[i],a[i+1]);
        i ← i+1;} j--;
}
```

CZĘŚCIOWA POPRAWNOŚĆ:

po każdym obrocie głównej pętli max{a[0],…,[j-1]} trafia na pozycję j-1;

Przykłady – dlaczego tylko while?

Inne pętle można „zamienić” na while
(ćw.)

Niezmienniki - podsumowanie

- Niezmienniki pętli możemy określać bez formułowania formalnej specyfikacji całych programów.
- Będziemy używać niezmienników w mniej formalnych „dowodach” poprawności programów.
- Samo sformułowanie i nieformalne uzasadnienie właściwego niezmiennika jest często ważnym krokiem do dowodu poprawności.

Jeszcze jeden przykład – selection

Program P:

```
k ← 0; i ← 1;
while (i<n)        {
   if (a[i]<a[k]) k ← i;
   i++;
}
zamień(a[k],a[0])
```

Jaki cel tego programu?

Jaki efekt?

Przykład – selection

Program P:

$k \leftarrow 0$; $i \leftarrow 1$;

while (i<n) {

   if (a[i]<a[k]) $k \leftarrow i$;

   i++;

}

zamień(a[k],a[0])

Można pokazać:

- niezmiennik pętli:

$N(k,i) \equiv$

$a[k]=\min\{a[0],\ldots,a[i-1]\}$

$\wedge\ 0 \le i \le n$

A dla całego programu P:

- $\{n>0,\ n \text{ naturalne}\}$

  P

  $\{\ a[0]=\min\{a[0],\ldots,a[n-1]\}\ \}$

Przykład – selection - pętla

Pętla L:

```
while (i<n)        {
   if (a[i]<a[k]) k ← i;
   i++;
}
```

Niezmiennik:

$N(k,i) \equiv a[k]=\min\{a[0],\ldots,a[i-1]\} \wedge 0 \le i \le n$

Przykład – selection - pętla

TW. $N(k,i) \equiv a[k]=\min\{a[0],\ldots,a[i-1]\} \wedge 0 \leq i \leq n$ to niezmiennik pętli L.

Dowód:

Zakł. że $N(k,i) \wedge (i<n)$ przed wykonaniem

if (a[i]<a[k]) k ← i; i++;

Chcemy pokazać, że N(k',i'), gdzie

k', i' to wartości k, i po if (a[i]<a[k]) k ← i; i++;

Przypadek 1: a[i]<a[k]

k' ustawiamy na „nowe minimum"

Przypadek 2: $\neg$a[i]<a[k]

k' równe k, minimum się nie zmienia