<u>Semantyczna poprawność algorytmów</u>

Pojęcia:

<u>Dane wejściowe</u> – wartość dostarczona do algorytmu z zewnątrz
<u>Wynik</u> – wartość przekazywana na zewnątrz
<u>Warunek początkowy</u> – ograniczenia na dane wejściowe
<u>Warunek końcowy</u> – własności wyników algorytmu i jego związek z danymi wejściowymi.

<u>Całkowita poprawność algorytmu</u>

Algorytm S jest całkowicie poprawny względem warunków początkowego P i końcowego Q jeśli dla każdych danych wejściowych spełniających warunek P obliczenie algorytmu S dochodzi do punktu końcowego i końcowe wartościowanie zmiennych spełnia warunek Q.

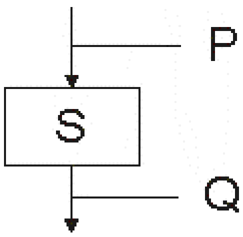
<u>Częściowa poprawność algorytmu</u>

Algorytm S jest częściowo poprawny względem warunków początkowego P i końcowego Q o ile dla każdych danych wejściowych spełniających warunek P jeżeli obliczenie algorytmu S dochodzi do punktu końcowego, to otrzymane wyniki spełniają warunek końcowy Q.

<u>Dowodzenie częściowej poprawności</u>

Metoda niezmienników:

1) Wyróżnienie pewnych miejsc w algorytmie i przypisanie do nich warunków na wartościowanie zmiennych,
2) udowodnienie, że jeśli spełniony jest warunek początkowy, to zawsze gdy obliczenie algorytmu dochodzi do wyróżnionego miejsca, to warunek przyporządkowany temu miejscu jest spełniony przez aktualne wartościowanie zmiennych.



schemat w postaci specyfikacji {P}S{Q}

Specyfikacja programu S:

{P}S{Q}

„ Jeśli relacja P jest prawdziwa przed wykonaniem S, to po wykonaniu S będzie zachodzić Q”

Specyfikacja instrukcji I:

{P}I{Q}

„ Jeśli relacja P jest prawdziwa przed wykonaniem I, to po wykonaniu I będzie zachodzić Q”

Reguły dowodzenia

Asercja to formuła rachunku zdań lub specyfikacja. Ogólna postać reguły dowodzenia $\frac{H1,\dots,Hn}{H}$

„ Jeśli H_1, …, H_n są prawdziwymi asercjami, to H jest prawdziwą asercją”

Reguły wynikowe

$$\frac{\{P\}S\{R\}, R => Q}{\{P\}S\{Q\}} \qquad \frac{P => R, \{R\}S\{Q\}}{\{P\}S\{Q\}}$$

Przykład:

Jeśli:

$$\{(x > 0) \wedge (y > 0)\}S\{(z + u * y = x * y) \wedge (u = 0)\}$$

$\{(z + u * y = x * y) \wedge (u = 0)\} => (z = x * y)$

$to\ z\ reguły\ dowodzenia:$

$\{(x > 0) \wedge (y > 0)\}S\{z = x * y\}$

Reguły dowodzenia dla instrukcji prostych:

Niech P-formuła logiczna
Instrukcja przypisania x:=e przypisuje zmiennej x wartość wyrażenia e

Ozn. P_e^x - formuła otrzymana z P w wyniku zastąpienia wyrażeniem e wszystkich wystąpień zmiennej x.

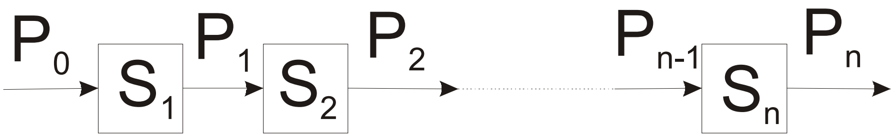
Reguła dla instrukcji przypisania

$\{P_e^x\}x := e\{P\}$

$Ex: \{x + y = 10\}\ z := x + y\{z = 10\}$

Instrukcja złożona

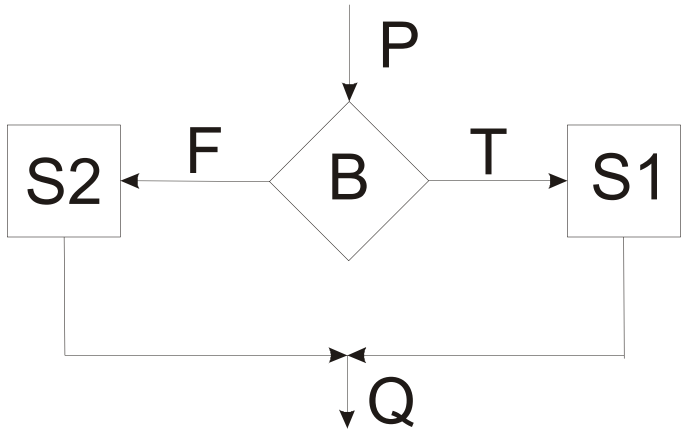
Sekwencyjne wykonanie instrukcji wewnętrznych:
BEGIN S1, S2, …, Sn END



Reguła dowodzenia dla instrukcji złożonej

$$\frac{\{P_{i-1}\}S_i\{P_i\}\ \ dla\ i = 1, \dots, n}{\{P_0\}\ begin\ S_1, S_2, \dots, S_n\ end\ \{P_n\}}$$

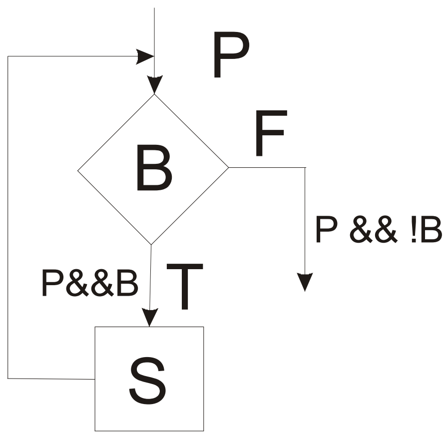
Instrukcja „jeśli”
if B then S1 else S2



Reguła dowodzenia dla instrukcji warunkowej

$$\frac{\{P \wedge B\}S_1\{Q\},\ \ \{P \wedge \sim B\}S_2\{Q\}}{\{P\}\ if\ B\ then\ S1\ else\ S2\ \{Q\}}$$

Instrukcja „dopóki”



Reguła dowodzenia dla instrukcji „dopóki”

$$\frac{\{P \wedge B\}S\{P\}}{\{P\}while\ \{B\}do\ S\{P \wedge \sim B\}}$$

P – niezmiennik pętli

Własność stopu

Algorytm S ma własność stopu pod względem warunku początkowego P, o ile dla każdych danych wejściowych spełniających warunek P obliczenie algorytmu S nie jest nieskończone.

Dowodzenie własności stopu

Pytanie: Kiedy może wystąpić obliczenie nieskończone algorytmu?

Odp: Przy wykonywaniu instrukcji iteracyjnej.

Dwie metody dowodzenia własności stopu:

1) metoda malejących wielkości
2) metoda liczników iteracji

Kryterium malejących wielkości

P – wyrażenie logiczne

e – wyrażenie całkowite nie występujące w „while B do S"

Tw. (kryterium malejących wielkości)

Zał.

(1) prawdziwa jest specyfikacja: $\{P \wedge B\}S\{P\}$

(2) prawdziwa jest implikacja $(P \wedge B) => e > 0$

(3) dla dowolnej zmiennej e_0 typu integer prawdziwa jest specyfikacja

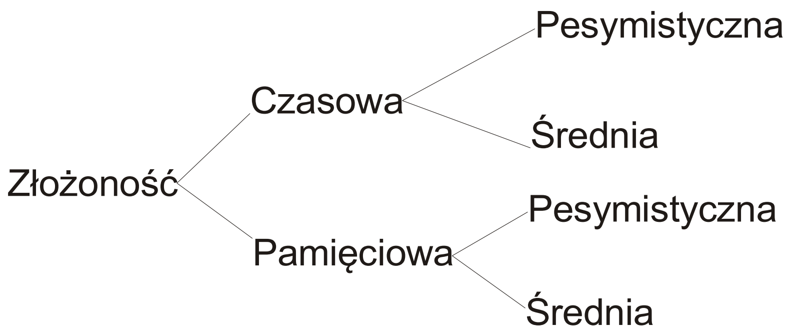
$$\{(0 < e = e_0)\}S\{e < e_0\}$$

(4) algorytm S ma własność stopu przy warunku początkowym $P \wedge B$

Teza:

Algorytm „while B do S” ma własność stopu względem warunku początkowego P (szkic dowodu. Nierówność e>=0 jest niezmiennikiem pętli. Wykonanie S zmniejsza wartość e. Zatem proces iteracyjny jest skończony).

Złożoność algorytmów



Dane:

- wyznacz największą spośród 3 liczb rzeczywistych

$$D = \{(x, y, z): x, y, z \in R\} = RxRxR = R^3$$

- Oblicz 2^n dla pewnej liczby naturalnej n.

$$D = \{n: n \in N\} = N$$

- Sprawdź czy dana liczba naturalna n jest liczbą pierwszą

$$D = \{n: n \in N\} = N$$

- Wyznacz największy wspólny dzielnik dwóch liczb naturalnych

$$D = \{(a, b): a, b \in N\} = NxN = N^2$$

- oblicz sumę płac pracowników w firmie

$$D = \{(n, p_1, \dots, p_n) : n \in N, p_1, \dots, p_n \in R\}$$

- wyznacz maksimum z ciągu liczb rzeczywistych

$$D = \{(n, x_1, \dots, x_n) : n \in N, x_1, \dots, x_n \in R\}$$

- oblicz iloczyn dwóch wektorów x, y o współrzędnych rzeczywistych.

$$D = \{(n, x_1, \dots, x_n, y_1, \dots, y_n) : n \in N, x_1, \dots x_n, y_1, \dots, y_n \in R\}$$

Pełna funkcja złożoności czasowej
Dany jest program P w Pascalu o zbiorze danych D.
Zakładamy, że:
- P jest poprawny
- obliczenie programu P na każdej danej z D jest skończone

Definiujemy funkcję t:D->N następująco:
t(d) = ilość operacji w obliczeniu programu P na danej $d \in D$
t – pełna funkcja złożoności czasowej programu P, pełna funkcja kosztu programu P

Przykłady:

Wyznacz maksimum z dwóch liczb rzeczywistych x,y
$D = \{(x, y) : x, y \in real\}$

Program max2	t(d)
Var x,y : real; Max : real; Begin Read(x,y)	2
If x>y then Max := x	2 lub 1
Else Max := y	0 lub 1
Writeln(Max) end	1 t(d) = 5

2) Oblicz n! dla n>=0

$D = \{n{:}n \in byte\}$

Program silnia;	t(d)
Var n:Byte;	
silnia : Longint;	
begin	
readln(n);	
if (n=0) or (n=1)	
then silnia := 1	1
else begin	
	3
silnia := 1	
	1
for i:=2 to n do	
silnia:=silnia*i	
	1
end	
	1*(n-1)
writeln(n)	
	2*(n-1)
end	
	1
	t(n) = 1+3+1 = 5 dla n=0 lub n=1
	t(n) = 1+3+1+(n-1)+2*(n-1)+1=3n+3 dla n>1

Rozmiar danych

Dowolną funkcję $r{:}\,D \to W$ nazywamy rozmiarem danych.

- oblicz sumę płac pracowników w firmie.

$$D = \{(n, p_1, \dots, p_n){:}\, n \in N, p_1, \dots, p_n \in R\}$$

$W = \{n : n \in N\} = N$

- wyznacz maksimum z ciągu liczb rzeczywistych

$D = \{(n, x_1, \dots, x_n) : n \in N, x_1, \dots, x_n \in R\}$

$W = \{n : n \in N\} = N D = \{(n, k, m, A, B) : n, m, k \in N, A \in Mac_{n,k}(R), B \in Mac_{n,k}(R)\}$

$W = \{(n, k, m) : n, k, m \in N\} = N^3 \; lub \; W = \{n : n = \max\{n, k, m\}\} = N$

Pesymistyczna złożoność czasowa

- J – zbiór operacji jednostkowych Pascala(niekoniecznie wszystkich)
- P – program w Pascalu
- D – zbiór danych programu P
- W – zbiór rozmiarów danych programu P
- t – pełna funkcja złożoności czasowej programu P

funkcję $T : W \to N$ zdefiniowaną następująco:

$T(w) = \sup\{t(d) : d \in D \wedge r(d) = w\}$

nazywamy pesymistyczną złożonością czasową programu.

Notacja „O"

Niech $g : N \to R$

$O(g) = \{f : N \to R : istenij\text{ą} \; sta\text{ł}e : c \in R, c > 0 \; i \; n_0 \in N, n > 0 \; takie \; \text{ż}e |f(n)| \leq c * |g(n)| \; dla \; wszystkich \; naturalnych \; n \geq n_0\}$

Dane są dwie funkcje $f, g : N \to R$

Mówimy, że „funkcja f jest co najwyżej rzędu funkcji g" jeśli $f \in O(g)$

Oznaczenie: f(n) = O(g(n))

Notacja „Ω"

Niech $g : N \to R$

$\mathbf{\Omega}(g) = \{f : N \to R : istenij\text{ą} \; sta\text{ł}e : c \in R, c > 0 \; i \; n_0 \in N, n > 0 \; takie \; \text{ż}e |f(n)| \leq c * |g(n)| \; dla \; wszystkich \; naturalnych \; n \geq n_0\}$

Dane są dwie funkcje $f, g : N \to R$

Mówimy, że „funkcja f jest co najmniej rzędu funkcji g" jeśli $f \in \mathbf{\Omega}(g)$

Oznaczenie: f(n) = O(g(n))

Notacja "θ"

Niech $g : N \to R$

$\theta(g) = O(g) \cap \Omega(g)$

dane są dwie funkcje $f, g : N \to R$

Mówimy, że „funkcja f jest dokładnie rzędu funkcji g" jeśli $f \in \theta(g)$

Oznaczenie: f(n) = θ(g(n))

Własności notacji „O"

- $\forall(k > 0)\, kf = O(f)$
- $n^r = O(n^s)\ jeśli\ 0 \leq r \leq s$
- $jeśli\ f = O(g), to\ f + g = O(g)$
- $Jeśli\ f\ jest\ wielomianem\ stopnia\ d, to\ f = O(n^d)$
- $Jeśli\ f = O(g)\ i\ g = O(h), to\ f = O(h)$
- $Jeśli\ f = O(g)\ i\ h = O(r),\ to\ fh = O(gr)$
- $\forall(b > 1\ and\ k \geq 0)\ n^k = O(b^n)$
- $\forall(b > 1\ and\ k > 0)\ \log_b n = O(n^k)$

Podsumowanie

Dla algorytmu A i zbioru operacji J:

znajdujemy funkcję $g: W \rightarrow R$, gdzie W jest zbiorem rozmiaru danych i rzeczywistą stałą c>0 taką, że $T_{A,J,W}(W) \leq c * |g(w)|$ dla prawie wszystkich $w \in W$,

Wtedy piszemy T(w) = O(g(w)) („T jest co najwyżej rzędu g")

lub T = O(g)