Programozási nyelvek és paradigmák

Szerződésalapú programozás

Kozsik Tamás (2020)

`check`-utasítás

- Design by Contract

`check` *assertion* `end`

check-utasítás

- Design by Contract

`check` *assertion* `end`

```
check
   size_is_not_too_large: size <= capacity
   -- Size is not too large because ...
end
```

Ciklus szerződése

```
gcd( a, b: INTEGER ): INTEGER
local
   number: INTEGER
do
   from
      Result := a
      number := b
   invariant 0 < Result; 0 < number;
             -- gcd(a,b) = gcd(Result,number)
   variant Result + number
   until Result = number loop
      if Result > number
      then Result := Result - number
      else number := number - Result
      end
   end
end -- gcd
```

Rutin szerződése

```
gcd( a, b: INTEGER ): INTEGER
require
   0 < a; 0 < b
local
   number: INTEGER
do
   from ...
   invariant ...
   variant ...
   until ...
   loop ...  end
ensure
   Result > 0; a \\ Result = 0; b \\ Result = 0;
   -- for all n: (a \\ n = 0 and b \\ n = 0)
   --                                  implies n <= Result
end -- gcd
```

Elő- és utófeltételes specifikáció

$$A \;=\; a : \mathbb{Z} \times b : \mathbb{Z} \times \mathrm{Result} : \mathbb{Z}$$

$$B \;=\; a' : \mathbb{Z} \times b' : \mathbb{Z}$$

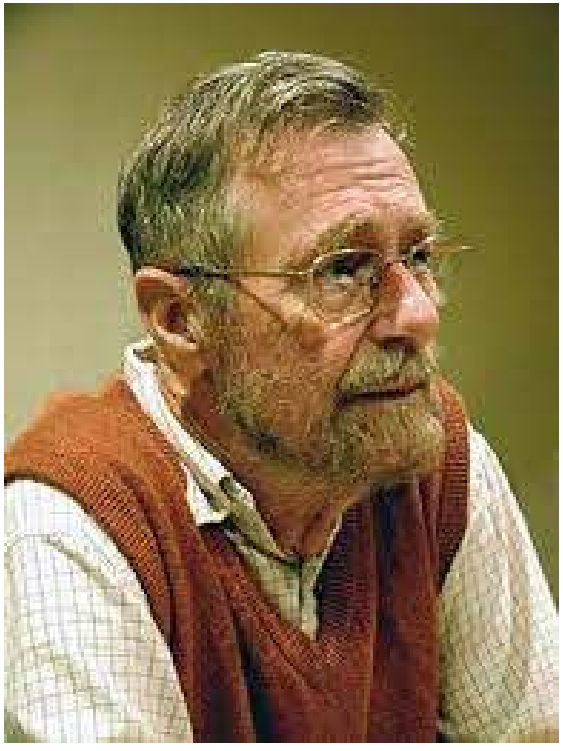
$$Q_{a',b'} \Rightarrow lf(\mathrm{GCD}, R_{a',b'})$$

ahol $Q_{a',b'},\ R_{a',b'} :\ A \to \mathbb{L}$, és

$$Q_{a',b'} \;=\; (a = a' > 0 \wedge b = b' > 0)$$

$$\begin{aligned} R_{a',b'} \;=\;& Q_{a',b'} \wedge (\mathrm{Result} > 0 \wedge a|\mathrm{Result} \wedge b|\mathrm{Result}) \\ & \wedge \Big(\forall c \in \mathbb{Z} : (a|c \wedge b|c) \;\to\; c \leq \mathrm{Result}\Big) \end{aligned}$$

Leggyengébb előfeltétel (weakest precondition)

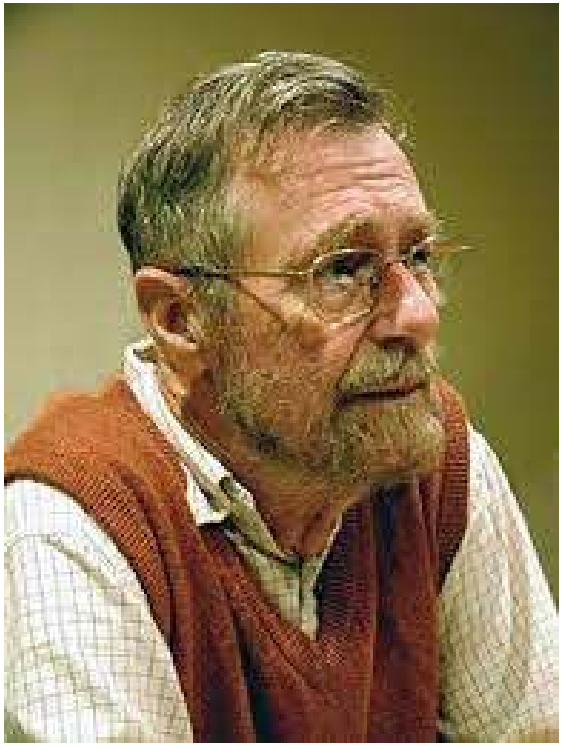


Edsger Wybe Dijkstra

$lf(S, P)$ vagy $wp(S, P)$

Ezen tulajdonságú állapotból lefuttatva az S programot, a program garantáltan véget ér, és P tulajdonságú állapotba kerül.

Leggyengébb előfeltétel (weakest precondition)



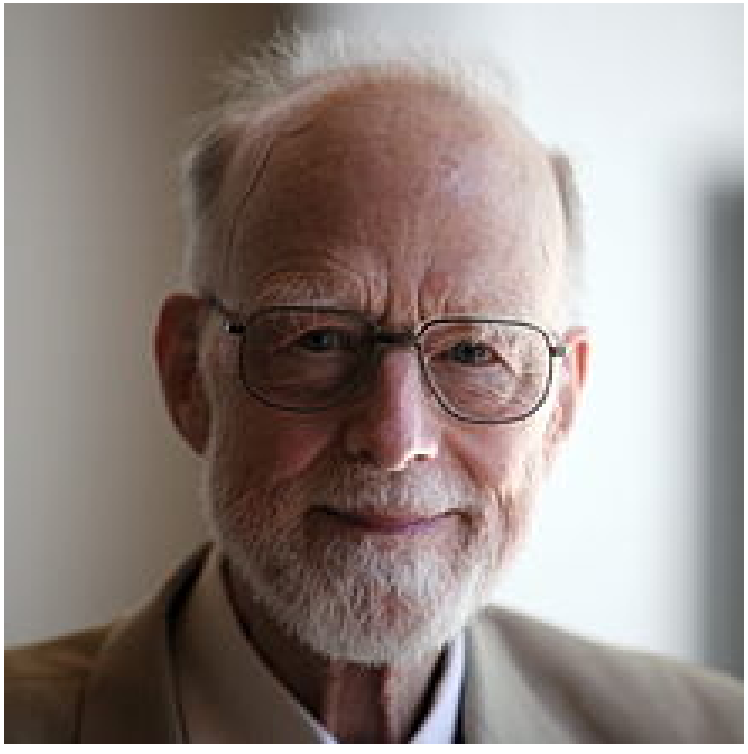
Edsger Wybe Dijkstra

$lf(S, P)$ vagy $wp(S, P)$

Ezen tulajdonságú állapotból lefuttatva az S programot, a program garantáltan véget ér, és P tulajdonságú állapotba kerül.

$$lf(S, P) = \left\{ a \in \mathcal{D}_{p(S)} \mid p(S)(a) \subseteq P \right\}$$

Hoare-hármas (Hoare-triple)



Sir Charles Antony Richard Hoare

- Axiomatikus szemantika
- Parciális és totális helyesség

$$\{\text{előfeltétel}\}\ \text{program}\ \{\text{utófeltétel}\}$$

Elő- és utófeltételes specifikáció Hoare-hármassal

$\forall (a', b') \in \mathbb{Z} \times \mathbb{Z}$:

$$\{Q_{a',b'}\}\ \mathrm{GCD}\ \{R_{a',b'}\}$$

ahol

$$Q_{a',b'} \ = \ (a = a' > 0 \wedge b = b' > 0)$$

$$\begin{aligned} R_{a',b'} \ = \quad & Q_{a',b'} \wedge (\mathrm{Result} > 0 \wedge a|\mathrm{Result} \wedge b|\mathrm{Result}) \\ & \wedge \Big(\forall c \in \mathbb{Z} : (a|c \wedge b|c) \ \rightarrow \ c \leq \mathrm{Result}\Big) \end{aligned}$$

Rutinok Eiffelben – példa

```
gcd( a, b: INTEGER ): INTEGER
require
   0 < a; 0 < b
local
   ...
do
   ...
ensure
   Result > 0; a \\ Result = 0; b \\ Result = 0;
   -- for all n: (a \\ n = 0 and b \\ n = 0)
   --                                  implies n <= Result
end -- gcd
```

Rutinok Eiffelben

```
rutin
    require
        REQ
    local
        ...
    do
        ...
    ensure
        ENS
    end
```

ahol

- REQ, ENS: $A \to$ BOOLEAN
- A: formális paraméterek és az aktuális objektum mezői

Egy rutin mikor felel meg a szerződésének?

Ezt később még finomítani fogjuk...

$$\{\texttt{REQ}\}\ \texttt{rutin}\ \{\texttt{ENS}\}$$

Ciklus levezetési szabálya

Tekintsük a `while` π `do` B programot. Legyen Q az előfeltétele, R az utófeltétele, I az invariánsa, t a terminálló (*variáns*-) függvénye.

Ha

1. $Q \Rightarrow I$
2. $I \wedge \neg\pi \Rightarrow R$
3. $I \wedge \pi \Rightarrow lf(B, I)$
4. $I \Rightarrow t \geq 0$
5. $\forall t_0 : \ I \wedge \pi \wedge t = t_0 \Rightarrow lf(B, t \leq t_0 - 1)$

akkor

$$Q \Rightarrow lf(\texttt{while}\ \pi\ \texttt{do}\ B, R)$$

Ciklus levezetési szabálya – Hoare-hármassal

Tekintsük a `while` π `do` B programot. Legyen Q az előfeltétele, R az utófeltétele, I az invariánsa, t a terminálló (*variáns*-) függvénye.

Ha

1. $Q \Rightarrow I$
2. $I \wedge \neg\pi \Rightarrow R$
3. $\{I \wedge \pi\}\ B\ \{I\}$
4. $I \Rightarrow t \geq 0$
5. $\forall t_0 :\ \{I \wedge \pi \wedge t = t_0\}\ B\ \{t \leq t_0 - 1)\}$

akkor

$$\{Q\}\ \texttt{while}\ \pi\ \texttt{do}\ B\ \{R\}$$

Ciklus Eiffelben – példa

```
...
from
   Result := a
   number := b
invariant 0 < Result; 0 < number;
          -- gcd(a,b) = gcd(Result,number)
variant Result + number
until Result = number loop
   if Result > number
   then Result := Result - number
   else number := number - Result
   end
end
...
```

Ciklus Eiffelben

```
...
from INIT
invariant INV
variant VAR
until COND
loop BODY
end
...
```

ahol

- `INIT`, `BODY`: összetett utasítás
- `INV`, `COND`: $A \rightarrow$ `BOOLEAN`
- `VAR`: $A \rightarrow$ `INTEGER`
- és A: formális paraméterek, lokális változók, az aktuális objektum mezői

Egy ciklus mikor felel meg a szerződésének?

1. $\{true\}$ `INIT` $\{$`INV`$\}$
2. –
3. $\{$`INV` $\wedge \neg$`COND`$\}$ `BODY` $\{$`INV`$\}$
4. `INV` $\Rightarrow$ `VAR` ≥ 0
5. $\forall v:$ $\{$`INV` $\wedge \neg$`COND` $\wedge$ `VAR` $= v\}$ `BODY` $\{$`VAR` $\leq v - 1)\}$