

# Verificarea și Validarea Sistemelor Soft

## Curs 8. Corectitudinea programelor (Floyd. Hoare. Dijkstra) Partea a II-a

Lector dr. Camelia Chisăliță-Crețu

Universitatea Babeș-Bolyai  
Cluj-Napoca

10 Aprilie 2024

- 1 Metode pentru demonstrarea corectitudinii
  - Instrucțiuni cu santinelă. Non-determinism
  - Derivarea formală a programelor
- 2 Dezvoltarea algoritmilor corecți din specificații
  - Rafinare
  - Reguli de rafinare
- 3 Analiză statică. Analiză dinamică
- 4 Urmează...
- 5 Bibliografie

# Instrucțiuni cu santinelă

- **instrucțiune cu santinelă** (*engl. guarded command*)
  - o listă de instrucțiuni prefixată de o expresie booleană;
  - dacă expresia booleană este inițial evaluată la true atunci lista instrucțiunilor este eligibilă pentru execuție;
  - sintaxă:
    - $\langle \textit{guarded command} \rangle ::= \langle \textit{guard} \rangle \rightarrow \langle \textit{guarded list} \rangle$
    - $\langle \textit{guard} \rangle ::= \langle \textit{boolean expression} \rangle$
    - $\langle \textit{guarded list} \rangle ::= \langle \textit{statement} \rangle \{ ; \langle \textit{statement} \rangle \}$
    - $\langle \textit{guarded command set} \rangle ::=$   
 $\langle \textit{guarded command} \rangle \{ \square \langle \textit{guarded command} \rangle \}$
    - $\langle \textit{alternative construct} \rangle ::= \textbf{if} \langle \textit{guarded command set} \rangle \textbf{fi}$
    - $\langle \textit{repetitive construct} \rangle ::= \textbf{do} \langle \textit{guarded command set} \rangle \textbf{od}$
    - $\langle \textit{statement} \rangle ::= \langle \textit{alternative construct} \rangle \mid$   
 $\langle \textit{repetitive construct} \rangle \mid \textit{"other statements"}$

# Non-determinism. Exemple.

## ● Exemplu . Maximul a două numere:

**if**  $x \geq y \rightarrow m := x$

$\square$   $y \geq x \rightarrow m := y$

**fi**

## Cea mai slabă precondiție

- **Hoare** – introduce precondiția suficientă astfel încât algoritmul să obțină rezultate corecte;
  - totuși nu există certitudinea că algoritmul se va termina;
- **Dijkstra** – introduce precondiția necesară și suficientă astfel încât algoritmul să permită obținerea rezultatului corect;
  - cea mai slabă precondiție (*engl.* **weakest precondition, wp**)
    - $wp(S, R)$ , unde
      - $S$  – mulțime de instrucțiuni;
      - $R$  – predicat (condiție) asupra stării programului;
      - pornind execuția instrucțiunilor  $S$  dintr-o stare  $p$ , execuția se termină și starea în care se ajunge satisface pe  $R$ ;
    - $wp$  - transformă o precondiție într-o postcondiție  $R$  (*engl.* **predicate transformer**).

# Proprietățile *wp* [Dij75]

## 1 legi:

- Legea miracolului exclus;
- Legea monotoniei;
- Legea conjuncției;
- Legea disjuncției.

## 2 operatori:

- atribuire ( $:=$ );
- concatenare ( $;$ );

## 3 structuri:

- secvențială;
- alternativă;
- repetitivă.

# Proprietățile $wp$ [Fre10]

## 1 Legea miracolului exclus

pentru orice  $S$ , pentru toate stările, unde  $R = FALSE$  are loc:

$$wp(S, FALSE) = FALSE;$$

## 2 Legea monotoniei

pentru orice  $S$  și orice două post-condiții, astfel încât pentru toate stările  $P \Rightarrow Q$ , pentru toate stările are loc:

$$wp(S, P) \Rightarrow wp(S, Q);$$

## 3 Legea conjuncției

pentru orice  $S$  și orice două post-condiții  $P$  și  $Q$ , pentru toate stările:

$$wp(S, P) \wedge wp(S, Q) = wp(S, P \wedge Q);$$

## 4 Legea disjuncției

pentru orice  $S$  determinist și orice post-condiții  $P$  și  $Q$ , pentru toate stările:

$$(wp(S, P) \vee wp(S, Q)) \Rightarrow wp(S, P \vee Q).$$

# Operatorul de atribuire și concatenare

## ● operatorul de atribuire ( $:=$ )

- semantica expresiei  $x := E$  se poate descrie prin:
  - $\text{wp}("x := E", R) = R_E^x$ , unde
    - $R_E^x$  – o copie a predicatului  $R$ , pentru care, fiecare apariție a variabilei  $x$  este înlocuită de  $E$ .

## ● operatorul de concatenare ( $;$ )

- semantica expresiei de concatenare  $;$  se poate descrie prin:
  - $\text{wp}("S1; S2", R) = \text{wp}(S1, \text{wp}(S2, R));$ 
    - $S1, S2$  – blocuri de instrucțiuni;
    - $R$  – postcondiție.



## Structura alternativă

### Definition

1. Fie  $IF$  descrisă prin **if**  $B_1 \rightarrow SL_1 \square \dots \square B_n \rightarrow SL_n$  **fi**. Fie  $BB$  descrisă prin  $(\exists i : 1 \leq i \leq n : B_i)$ , atunci  
 $wp(IF, R) = (BB \wedge (\forall i : 1 \leq i \leq n : B_i \Rightarrow wp(SL_i, R)))$ .

### Theorem

#### Substituția simplă

1. Pentru  $(\forall i : 1 \leq i \leq n : (Q \wedge B_i) \Rightarrow wp(SL_i, R))$  pentru toate stările, atunci  $(Q \wedge BB) \Rightarrow wp(IF, R)$  are loc în toate stările.

- $t : SSet \rightarrow Z$ ,  $SSet$  – domeniul stărilor;
- Fie  $wdec(S, t)$  – cea mai slabă precondiție definită pentru  $S$ , pentru care funcția  $t$  descrește în starea finală, față de cea inițială.

### Theorem

1. Pentru  $(\forall i : 1 \leq i \leq n : (Q \wedge B_i) \Rightarrow wdec(SL_i, t))$ , pentru toate stările se poate spune că  $(Q \wedge BB) \Rightarrow wdec(IF, t)$  are loc în toate stările.

# Construcția repetitivă

## Definition

2. Fie  $DO$  descrisă prin **do**  $B_1 \rightarrow SL_1 \square \dots \square B_n \rightarrow SL_n$  **od**. Fie  $H_0(R) = (R \wedge \neg BB)$  și pentru  $k > 0$ ,  $H_k(R) = (wp(IF, H_{k-1}(R))) \vee H_0(R)$ , atunci, prin definiție,  $wp(DO, R) = (\exists k : k \geq 0 : H_k(R))$ .

## Theorem

3. Dacă pentru toate stările avem  $(P \wedge BB) \Rightarrow (wp(IF, P) \wedge wdec(IF, t) \wedge t \geq 0)$  atunci pentru toate stările avem  $P \Rightarrow wp(DO, P \wedge \neg BB)$ .

## Definition

3.  $T$  este condiția satisfăcută de toate stările și  $wp(S, T)$  este cea mai slabă precondiție care garantează terminarea programului  $S$ .

## Theorem

4. Dacă  $(P \wedge BB) \Rightarrow wp(IF, P)$  pentru toate stările, atunci  $(P \wedge wp(DO, T) \Rightarrow wp(DO, P \wedge \neg BB))$  pentru toate stările.

# Rafinare

- Date de intrare:  $X$ ;                      pre-condiție:  $\varphi(X)$   
  Date de ieșire:  $Z$ ;                      post-condiție:  $\psi(X, Z)$
- program abstract  
   $Z : [\varphi, \psi]$
- rafinare  
   $\prec$  – are semnificația *se rescrie prin...*  
   $Z = P_0 \prec P_1 \prec P_2 \prec \dots \prec P_{n-1} \prec P_n$
- reguli de rafinare
  - regula atribuirii;
  - regula compunerii secvențiale;
  - regula alternanței;
  - regula iterației.

## Rafinare [Fre10]

- Regula atribuirii:

$[\varphi(v/e), \psi] \prec v := e$

- Regula compunerii secvențiale:

$[\eta_1, \eta_2] \prec [\eta_1, \gamma]$   
 $[\gamma, \eta_2]$

( $\gamma$  - predicat auxiliar  
(*engl. middle predicate*))

- Regula alternanței:

$cond = c_1 \vee c_2 \vee \dots \vee c_n;$

$[\eta_1, \eta_2] \prec$

**if**  $c_1 \rightarrow [\eta_1 \wedge c_1, \eta_2]$

$\square$   $c_2 \rightarrow [\eta_1 \wedge c_2, \eta_2]$

$\vdots$

$\square$   $c_n \rightarrow [\eta_1 \wedge c_n, \eta_2]$

**fi**

- Regula iterației:

$cond = c_1 \vee c_2 \vee \dots \vee c_n$

$[\eta, \eta \wedge \neg cond] \prec$

**do**  $c_1 \rightarrow [\eta \wedge c_1, \eta \wedge TC]$

$\square$   $c_2 \rightarrow [\eta \wedge c_2, \eta \wedge TC]$

$\vdots$

$\square$   $c_n \rightarrow [\eta \wedge c_n, \eta \wedge TC]$

**od**

# Exemple

- **Rafinare.pdf.**

# Instrumente software pentru analiza statică și analiza dinamică

- ESC2Java - Extended Static Checker to Java - **Seminar 06**;
- JML- Java Modeling Language - **Seminar 06**;

## Pentru examen...

- teoria Dijkstra
  - **rafinare**: definiții reguli;
  - rafinare algoritmi din specificații  
(link: [Rafinare.pdf](#))  
(2 probleme – [Seminar 06](#)):
    - împărțire întreagă (cât și rest);
    - rădăcină pătrată;

# Bibliografie I

- [Dij75] E. Dijkstra.  
Guarded commands, nondeterminacy and formal derivation of programs.  
*CACM*, 8(18):453–457, 1975.
- [Fre10] M. Frentiu.  
*Verificarea și validarea sistemelor soft*.  
Presa Universitară Clujeană, 2010.