Fundamentele Limbajelor de programare

Verificarea programelor imperative. Logica Hoare

Denisa Diaconescu[1]

Facultatea de Matematică și Informatică, DL Info

Anul II, Semestrul II, 2024/2025

[1]Cursul Program Verification, master anul I. Traducerea și adaptarea: T.F. Șerbănuță

Prezentare generală

1. Ce încercăm să rezolvăm?

2. Logica Hoare

Verificarea limbajelor imperative

- Limbajele imperative sunt construite în jurul ideii de stare a programului (date stocate în memorie).
- Programele imperative sunt secvențe de comenzi care modifică această stare.

Verificarea limbajelor imperative

- Limbajele imperative sunt construite în jurul ideii de stare a programului (date stocate în memorie).
- Programele imperative sunt secvențe de comenzi care modifică această stare.
- Pentru a demonstra proprietăți ale programelor imperative, avem nevoie de:
  1. O modalitate de a scrie afirmații despre starea programului.
  2. Reguli pentru manipularea și demonstrarea acestor afirmații.

Verificarea limbajelor imperative

- Limbajele imperative sunt construite în jurul ideii de stare a programului (date stocate în memorie).
- Programele imperative sunt secvențe de comenzi care modifică această stare.
- Pentru a demonstra proprietăți ale programelor imperative, avem nevoie de:
  1. O modalitate de a scrie afirmații despre starea programului.
  2. Reguli pentru manipularea și demonstrarea acestor afirmații.
- Acestea vor fi oferite de Logica Hoare.

C.A.R. (Tony) Hoare

Logica Hoare a fost introdusă de Tony Hoare.
Tot el a inventat algoritmul Quicksort în anul 1960 (avea 26 de ani).

Triplete Hoare

Un triplet Hoare $\{P\}$ S $\{Q\}$ are trei componente:

P o pre-condiție

S un fragment de cod

Q o post-condiție

Pre-condiția este o afirmație care se referă la starea de dinaintea execuției codului.

Post-condiția este o afirmație care se referă la starea de după execuția codului.

Sintaxă

Pre-condiția și post-condiția se construiesc din variabilele programului, numere, relații aritmetice, și vor folosi logică propozitională pentru a combina afirmații simple.

Exemplu

- $x = 3$, $\neg(x = y)$, $\neg(0 \leq x)$
- $x = 4 \land y = 2$
- $(\neg y \leq x) \to (x = 2 * y)$
- `true`, `false`

Dacă avem nevoie de o logică mai puternică, putem folosi logica de ordinul I.

Exemplu

Pentru a exprima că valoarea variabilei de program x este un pătrat perfect, putem scrie $\exists n.x = n * n$.

Semantica intuitivă

Reamintim că o stare este dată de valorile (întregi) ale variabilelor de program (locațiilor de memorie).

Semantica intuitivă

Reamintim că o stare este dată de valorile (întregi) ale variabilelor de program (locațiilor de memorie).

Triplet Hoare: $\{P\}$ S $\{Q\}$

- dacă P e adevărată în starea de dinaintea execuției lui S
- și execuția lui S se termină
- atunci Q e adevărată în starea de după execuția lui S

Corectitudine parțială

Logica Hoare exprimă corectitudine parțială!

- Un program este parțial corect dacă dă răspunsul așteptat atunci când se termină.
- Nu greșește niciodată, dar e posibil să nu dea nici un răspuns.

Corectitudine parțială

Logica Hoare exprimă corectitudine parțială!

- Un program este parțial corect dacă dă răspunsul așteptat atunci când se termină.
- Nu greșește niciodată, dar e posibil să nu dea nici un răspuns.

Exemplu

$$\{x = 1\}\ \texttt{while x=1 do y:=2}\ \{x = 3\}$$

este o afirmație adevărată conform logicii Hoare.

- dacă starea de dinaintea execuției satisface $x = 1$ și bucla while se termină, atunci starea de după satisface $x = 3$. Dar bucla while nu se termină!

Corectitudinea parțială e suficientă

De ce nu insistăm asupra terminării?

- Poate că nu vrem terminare.
- Simplifică logica (există variante și pentru corectitudine totală).
- Există metode specializate pentru demonstrarea terminării, dacă vrem.

Cum?

Exemplu

Tripletele Hoare ne permit să facem afirmații precum:

$$\{x > 0\}\ \texttt{y:=0-x}\ \{y < 0 \land x \neq y\}$$

Cum?

Exemplu

Tripletele Hoare ne permit să facem afirmații precum:

$$\{x > 0\}\ \texttt{y:=0-x}\ \{y < 0 \land x \neq y\}$$

Dacă $(x > 0)$ este adevărată înainte de execuția instrucțiunii `y:=0-x` atunci $(y < 0 \land x \neq y)$ este adevărată după.

Cum?

Exemplu

Tripletele Hoare ne permit să facem afirmații precum:

$$\{x > 0\}\ \texttt{y:=0-x}\ \{y < 0 \land x \neq y\}$$

Dacă $(x > 0)$ este adevărată înainte de execuția instrucțiunii `y:=0-x` atunci $(y < 0 \land x \neq y)$ este adevărată după.
Această afirmație este în mod intuitiv adevărată. Cum o demonstrăm?

Cum?

Exemplu

Tripletele Hoare ne permit să facem afirmații precum:

$$\{x > 0\}\ \texttt{y:=0-x}\ \{y < 0 \wedge x \neq y\}$$

Dacă $(x > 0)$ este adevărată înainte de execuția instrucțiunii `y:=0-x` atunci $(y < 0 \wedge x \neq y)$ este adevărată după.
Această afirmație este în mod intuitiv adevărată. Cum o demonstrăm?

Avem nevoie de niște reguli pentru a deduce (formal) astfel de triplete. Vom avea câte o regulă pentru fiecare din cele cinci feluri de instrucțiuni (plus încă două alte reguli).

Axioma pentru `skip` (Regula 0/6)

`skip` nu face nimic, deci e de așteptat ca tripletele Hoare pentru `skip` reflecte acest lucru.

Axioma pentru `skip`:

$$\boxed{\{Q\}\ \texttt{skip}\ \{Q\}}$$

Axioma atribuirii (Regula 1/6)

Atribuirile schimbă starea, deci e de așteptat ca tripletele Hoare pentru atribuire să reflecte această schimbare.

Axioma atribuirii (Regula 1/6)

Atribuirile schimbă starea, deci e de așteptat ca tripletele Hoare pentru atribuire să reflecte această schimbare.

Axioma atribuirii:

$$\boxed{\{Q(e)\}\ \texttt{x:=e}\ \{Q(x)\}}$$

($Q(x)$ este o proprietate asupra unei variabile x și $Q(e)$ indică aceeași proprietate în care toate aparițiile lui x au fost înlocuite de expresia e)

Axioma atribuirii (Regula 1/6)

Atribuirile schimbă starea, deci e de așteptat ca tripletele Hoare pentru atribuire să reflecte această schimbare.

Axioma atribuirii:

$$\boxed{\{Q(e)\}\ \texttt{x:=e}\ \{Q(x)\}}$$

($Q(x)$ este o proprietate asupra unei variabile x și $Q(e)$ indică aceeași proprietate în care toate aparițiile lui x au fost înlocuite de expresia e)

Dacă vrem ca x să aibă o anumită proprietate Q după atribuire, atunci acea proprietate trebuie să țină pentru expresia e atribuită lui x înainte de execuția atribuirii.

Axioma atribuirii

Exemplu

Regula inversă este falsă: $\{Q(x)\}$ `x:=e` $\{Q(e)\}$
Dacă am aplica această "axiomă" greșită pre-condiției $x = 0$ și fragmentului de cod `x:=1`, am obține

$$\{x = 0\}\ \texttt{x:=1}\ \{1 = 0\}$$

care se citește "dacă $x = 0$ initial și `x:=1` se temină, atunci $1 = 0$ după execuție".

Lucrăm invers, dinspre post-condiție spre pre-condiție

Ar putea părea natural să începem cu pre-condiția și să facem deducții înspre post-condiție, dar acesta nu e cel mai bun mod de a raționa folosind logica Hoare.

Dimpotrivă, pornim cu post-condiția și mergem ”înapoi”.

Lucrăm invers, dinspre post-condiție spre pre-condiție

Ar putea părea natural să începem cu pre-condiția și să facem deducții înspre post-condiție, dar acesta nu e cel mai bun mod de a raționa folosind logica Hoare.

Dimpotrivă, pornim cu post-condiția și mergem "înapoi".

Exemplu

Pentru a aplica axioma pentru atribuire,

$$\{Q(e)\}\ \texttt{x:=e}\ \{Q(x)\}$$

luăm post-condiția, o copiem în pre-condiție și apoi înlocuim toate aparițiile lui x cu e.
Observație: postcondiția poate avea una, mai multe, sau nici o apariție a lui x. Toate vor fi înlocuite cu e în pre-condiție!

Regula pentru atribuire

Axioma atribuirii: $\{Q(e)\}$ `x:=e` $\{Q(x)\}$

Exemplu

Presupunem că avem de executat `x:=2` și că post-condiția dorită este $y = x$.
O instanță a axiomei de atribuire:

Regula pentru atribuire

Axioma atribuirii: $\{Q(e)\}$ `x:=e` $\{Q(x)\}$

Exemplu

Presupunem că avem de executat `x:=2` și că post-condiția dorită este $y = x$.
O instanță a axiomei de atribuire:

$$\{y = 2\}\ \texttt{x:=2}\ \{y = x\}$$

Regula pentru atribuire

Exemplu

Să zicem că vrem să demonstrăm

$$\{y = 2\}\ \texttt{x:=y}\ \{x > 0\}?$$

Regula pentru atribuire

Exemplu

Să zicem că vrem să demonstrăm

$$\{y = 2\}\ \texttt{x:=y}\ \{x > 0\}?$$

Această aritmație este evident adevărată. Dar folosind axioma pentru atribuire obținem:

$$\{y > 0\}\ \texttt{x:=y}\ \{x > 0\}?$$

Și . . . $y > 0$ nu este echivalentă cu $y = 2$!

Afirmații tari și slabe

O afirmație P este mai tare decât Q dacă P o implică pe Q.
Dacă P este mai tare decât Q, atunci P are mai multe șanse să fie falsă decât Q.

Afirmații tari și slabe

O afirmație P este mai tare decât Q dacă P o implică pe Q.
Dacă P este mai tare decât Q, atunci P are mai multe șanse să fie falsă decât Q.

Exemplu (promisiune electorală)

- *Voi menține șomajul sub* 3% este mai tare decât
- *Voi menține șomajul sub* 15%

Afirmații tari și slabe

O afirmație P este mai tare decât Q dacă P o implică pe Q.
Dacă P este mai tare decât Q, atunci P are mai multe șanse să fie falsă decât Q.

Exemplu (promisiune electorală)

- *Voi menține șomajul sub* 3% este mai tare decât
- *Voi menține șomajul sub* 15%

Cea mai tare afirmație posibilă este $\perp$.
Cea mai slabă afirmație posibilă este $\top$.

Post-condiții tari

Exemplu

Triplul Hoare $\{x = 5\}$ `x:=x+1` $\{x = 6\}$ este mai informativ decât $\{x = 5\}$ `x:=x+1` $\{x > 0\}$.

Post-condiții tari

Exemplu

Triplul Hoare $\{x = 5\}$ `x:=x+1` $\{x = 6\}$ este mai informativ decât $\{x = 5\}$ `x:=x+1` $\{x > 0\}$.

Dacă post-condiția Q_1 este mai tare decât Q_2, atunci $\{P\}$ `S` $\{Q_1\}$ este o afirmație mai tare decât $\{P\}$ `S` $\{Q_2\}$.

Post-condiții tari

Exemplu

Triplul Hoare $\{x = 5\}$ `x:=x+1` $\{x = 6\}$ este mai informativ decât $\{x = 5\}$ `x:=x+1` $\{x > 0\}$.

Dacă post-condiția Q_1 este mai tare decât Q_2, atunci $\{P\}$ `S` $\{Q_1\}$ este o afirmație mai tare decât $\{P\}$ `S` $\{Q_2\}$.

Exemplu

Deoarece post-condiția $x = 6$ este mai tare decât $x > 0$ (întrucât $x = 6 \to x > 0$),
atunci $\{x = 5\}$ `x:=x+1` $\{x = 6\}$ este o afirmație mai tare decât $\{x = 5\}$ `x:=x+1` $\{x > 0\}$.

Pre-condiții slabe

Exemplu

Tripletul Hoare $\{x > 0\}$ `x:=x+1` $\{x > 1\}$ este mai informativ decât $\{x = 5\}$ `x:=x+1` $\{x > 1\}$.

Dacă o pre-condiție P_1 este mai slabă decât P_2, atunci $\{P_1\}$ `S` $\{Q\}$ este o afirmație mai tare decât $\{P_2\}$ `S` $\{Q\}$.

Exemplu

Deoarece pre-condiția $x > 0$ este mai slabă decât $x = 5$, atunci $\{x > 0\}$ `x:=x+1` $\{x > 1\}$ este o afirmație mai tare decât $\{x = 5\}$ `x:=x+1` $\{x > 1\}$.

Regula de întărire a pre-condiției (Regula 2/6)

Este sigur (corect) să întărim o pre-condiție.
Regula de întărire a pre-condiției:

$$\boxed{\frac{\{P_w\}\ \mathtt{S}\ \{Q\}}{\{P_s\}\ \mathtt{S}\ \{Q\}} \quad \text{dacă } P_s \to P_w}$$

Exemplu

$$\frac{\{y > 0\}\ \mathrm{x} := \mathrm{y}\ \{x > 0\}}{\{y = 2\}\ \mathrm{x} := \mathrm{y}\ \{x > 0\}} \quad \text{deoarece } y = 2 \to y > 0$$

Regula de slăbire a post-condiției (Regula 3/6)

Este sigur (corect) să *relaxăm* (slăbim) o post-condiție.
Regula de slăbire a post-condiției:

$$\boxed{\frac{\{P\}\ \mathtt{S}\ \{Q_s\}}{\{P\}\ \mathtt{S}\ \{Q_w\}} \quad \text{dacă } Q_s \to Q_w}$$

Exemplu

$$\frac{\{x > 2\}\ \mathtt{x := x + 1}\ \{x > 3\}}{\{x > 2\}\ \mathtt{x := x + 1}\ \{x > 1\}} \quad \text{deoarece } x > 3 \to x > 1$$

Regula pentru secvențiere (Regula 4/6)

Programele imperative conțin instrucțiuni, care modifică starea secvențial.

Regula de secvențiere:

$$\boxed{\dfrac{\{P\}\mathrm{S}_1\{Q\} \quad \{Q\}\mathrm{S}_2\{R\}}{\{P\}\mathrm{S}_1;\mathrm{S}_2\{R\}}}$$

Regula pentru secvențiere (Regula 4/6)

Programele imperative conțin instrucțiuni, care modifică starea secvențial.
Regula de secvențiere:

$$\boxed{\frac{\{P\}\mathtt{S}_1\{Q\} \quad \{Q\}\mathtt{S}_2\{R\}}{\{P\}\mathtt{S}_1;\mathtt{S}_2\{R\}}}$$

Exemplu

$$\frac{\{x > 2\}\mathtt{x} := \mathtt{x} + 1\{x > 3\} \quad \{x > 3\}\mathtt{x} := \mathtt{x} + 2\{x > 5\}}{\{x > 2\}\mathtt{x} := \mathtt{x} + 1 \; ; \; \mathtt{x} := \mathtt{x} + 2\{x > 5\}}$$

Cum obținem condiția din mijloc?

De obicei, când aplicăm o regulă de forma

$$\boxed{\dfrac{\{P\}\mathrm{S}_1\{Q\} \quad \{Q\}\mathrm{S}_2\{R\}}{\{P\}\mathrm{S}_1;\mathrm{S}_2\{R\}}}$$

pre-condiția P și post-condiția R ne vor fi date; totuși, de unde obținem Q?

Cum obținem condiția din mijloc?

De obicei, când aplicăm o regulă de forma

$$\boxed{\frac{\{P\}\mathtt{S}_1\{Q\} \quad \{Q\}\mathtt{S}_2\{R\}}{\{P\}\mathtt{S}_1;\mathtt{S}_2\{R\}}}$$

pre-condiția P și post-condiția R ne vor fi date; totuși, de unde obținem Q? Pornind de la ținta noastră, R, și mergând înapoi!

$$\frac{\{x > 2\}\mathtt{x} := \mathtt{x} + 1\{Q\} \quad \{Q\}\mathtt{x} := \mathtt{x} + 2\{x > 5\}}{\{x > 2\}\mathtt{x} := \mathtt{x} + 1; \mathtt{x} := \mathtt{x} + 2\{x > 5\}}$$

Cum scriem o demonstrație?

Exemplu

Să zicem că vrem să demonstrăm

$$\{x = 3\}\ \texttt{x:=x+1;\ x:=x+2}\ \{x > 5\}.$$

Cum scriem o demonstrație?

Exemplu

Să zicem că vrem să demonstrăm

$$\{x = 3\}\ \texttt{x:=x+1; x:=x+2}\ \{x > 5\}.$$

1. $\{x + 2 > 5\}\ \texttt{x:=x+2}\ \{x > 5\}$ (Atribuire)

Observați numerotarea pașilor și justificările!

Cum scriem o demonstrație?

Exemplu

Să zicem că vrem să demonstrăm

$$\{x = 3\}\ \texttt{x:=x+1;}\ \ \texttt{x:=x+2}\ \{x > 5\}.$$

1. $\{x + 2 > 5\}$ `x:=x+2` $\{x > 5\}$ (Atribuire)
2. $\{(x + 1) + 2 > 5\}$ `x:=x+1` $\{x + 2 > 5\}$ (Atribuire)

Observați numerotarea pașilor și justificările!

Cum scriem o demonstrație?

Exemplu

Să zicem că vrem să demonstrăm

$$\{x = 3\}\ \texttt{x:=x+1; x:=x+2}\ \{x > 5\}.$$

1. $\{x + 2 > 5\}\ \texttt{x:=x+2}\ \{x > 5\}$ (Atribuire)
2. $\{(x + 1) + 2 > 5\}\ \texttt{x:=x+1}\ \{x + 2 > 5\}$ (Atribuire)
3. $\{(x + 1) + 2 > 5\}\ \texttt{x:=x+1 ; x:=x+2}\ \{x > 5\}$ (1,2, Seventiere)

Observați numerotarea pașilor și justificările!

Cum scriem o demonstrație?

Exemplu

Să zicem că vrem să demonstrăm

$$\{x = 3\}\ \texttt{x:=x+1;}\ \ \texttt{x:=x+2}\ \{x > 5\}.$$

1. $\{x + 2 > 5\}$ `x:=x+2` $\{x > 5\}$ (Atribuire)
2. $\{(x + 1) + 2 > 5\}$ `x:=x+1` $\{x + 2 > 5\}$ (Atribuire)
3. $\{(x + 1) + 2 > 5\}$ `x:=x+1 ; x:=x+2` $\{x > 5\}$ (1,2, Secvențiere)
4. $\{x = 3\}$ `x:=x+1; x:=x+2` $\{x > 5\}$. (3, Întărirea pre-condiției)
   deoarece $x = 3 \to (x + 1) + 2 > 5$

Observați numerotarea pașilor și justificările!

Regula pentru instrucțiunea `if` (Regula 5/6)

Regula pentru `if`:

$$\boxed{\dfrac{\{P \land b\}\ \mathtt{S_1}\ \{Q\} \quad \{P \land \neg b\}\ \mathtt{S_2}\ \{Q\}}{\{P\}\ \mathtt{if\ b\ then\ S_1\ else\ S_2}\ \{Q\}}}$$

Regula pentru instrucțiunea `if` (Regula 5/6)

Regula pentru `if`:

$$\frac{\{P \land b\}\ \mathtt{S_1}\ \{Q\} \quad \{P \land \neg b\}\ \mathtt{S_2}\ \{Q\}}{\{P\}\ \mathtt{if\ b\ then\ S_1\ else\ S_2}\ \{Q\}}$$

- Când se execută o instrucțiune `if`, se va executa ori $\mathtt{S_1}$ ori $\mathtt{S_2}$.
- De aceea, ca în urma execuției să fie validă afirmația Q, trebuie ca ambele ramuri $\mathtt{S_1}$ și $\mathtt{S_2}$ să facă Q adevărată.
- Asemănător, dacă pre-condiția pentru `if` este P, ea trebuie să fie o pre-condiție pentru ambele ramuri $\mathtt{S_1}$ și $\mathtt{S_2}$.
- Alegerea dintre $\mathtt{S_1}$ și $\mathtt{S_2}$ depinde de validitatea expresiei b în starea inițială; deci putem presupune b ca o pre-condiție pentru $\mathtt{S_1}$ și $\neg b$ ca o pre-condiție pentru $\mathtt{S_2}$.

Regula pentru If

Regula pentru `if`:

$$\boxed{\dfrac{\{P \land b\}\ \mathtt{S}_1\ \{Q\} \quad \{P \land \neg b\}\ \mathtt{S}_2\ \{Q\}}{\{P\}\ \mathtt{if\ b\ then\ S}_1\ \mathtt{else\ S}_2\ \{Q\}}}$$

Exemplu

Să zicem că vrem să demonstrăm

$$\{x > 3\}\ \mathtt{if\ x>2\ then\ y:=1\ else\ y:=-1}\ \{y > 0\}$$

Regula pentru If

Regula pentru `if`:

$$\boxed{\dfrac{\{P \wedge b\}\ \mathtt{S}_1\ \{Q\} \quad \{P \wedge \neg b\}\ \mathtt{S}_2\ \{Q\}}{\{P\}\ \mathtt{if\ b\ then\ S}_1\ \mathtt{else\ S}_2\ \{Q\}}}$$

Exemplu

Să zicem că vrem să demonstrăm

$$\{x > 3\}\ \mathtt{if\ x>2\ then\ y:=1\ else\ y:=-1}\ \{y > 0\}$$

Regula pentru `if` ne spune că e suficient să demonstrăm:

I $\{(x > 3) \wedge (x > 2)\}\ \mathtt{y:=1}\ \{y > 0\}$

II $\{(x > 3) \wedge \neg(x > 2)\}\ \mathtt{y:=-1}\ \{y > 0\}$

Regula pentru If

Exemplu (cont.)

Pentru I. $\{(x > 3) \wedge (x > 2)\}$ `y:=1` $\{y > 0\}$ avem demonstrația:

1. $\{1 > 0\}$ `y:=1` $\{y > 0\}$ (Atribuire)

Regula pentru If

Exemplu (cont.)

Pentru I. $\{(x > 3) \land (x > 2)\}$ `y:=1` $\{y > 0\}$ avem demonstrația:

1. $\{1 > 0\}$ `y:=1` $\{y > 0\}$ (Atribuire)
2. $\{(x > 3) \land (x > 2)\}$ `y:=1` $\{y > 0\}$ (1, Întărirea pre-condiției)
   deoarece $(x > 3) \land (x > 2) \to 1 > 0$

Regula pentru If

Exemplu (cont.)

Pentru I. $\{(x > 3) \wedge (x > 2)\}$ `y:=1` $\{y > 0\}$ avem demonstrația:

1. $\{1 > 0\}$ `y:=1` $\{y > 0\}$ (Atribuire)
2. $\{(x > 3) \wedge (x > 2)\}$ `y:=1` $\{y > 0\}$ (1, Întărirea pre-condiției)
   deoarece $(x > 3) \wedge (x > 2) \to 1 > 0$

Pentru II. $\{(x > 3) \wedge \neg(x > 2)\}$ `y:=-1` $\{y > 0\}$ avem demonstrația:

3. $\{-1 > 0\}$ `y:=-1` $\{y > 0\}$ (Atribuire)

Regula pentru If

Exemplu (cont.)

Pentru I. $\{(x > 3) \wedge (x > 2)\}$ `y:=1` $\{y > 0\}$ avem demonstrația:

1. $\{1 > 0\}$ `y:=1` $\{y > 0\}$ (Atribuire)
2. $\{(x > 3) \wedge (x > 2)\}$ `y:=1` $\{y > 0\}$ (1, Întărirea pre-condiției)
   deoarece $(x > 3) \wedge (x > 2) \rightarrow 1 > 0$

Pentru II. $\{(x > 3) \wedge \neg(x > 2)\}$ `y:=-1` $\{y > 0\}$ avem demonstrația:

3. $\{-1 > 0\}$ `y:=-1` $\{y > 0\}$ (Atribuire)
4. $\{(x > 3) \wedge \neg(x > 2)\}$ `y:=-1` $\{y > 0\}$ (3, Întărirea pre-condiției)
   deoarece $(x > 3) \wedge \neg(x > 2) \rightarrow -1 > 0$

Regula pentru If

Exemplu (cont.)

Pentru I. $\{(x > 3) \land (x > 2)\}$ `y:=1` $\{y > 0\}$ avem demonstrația:

1. $\{1 > 0\}$ `y:=1` $\{y > 0\}$ (Atribuire)
2. $\{(x > 3) \land (x > 2)\}$ `y:=1` $\{y > 0\}$ (1, Întărirea pre-condiției)
   deoarece $(x > 3) \land (x > 2) \to 1 > 0$

Pentru II. $\{(x > 3) \land \neg(x > 2)\}$ `y:=-1` $\{y > 0\}$ avem demonstrația:

3. $\{-1 > 0\}$ `y:=-1` $\{y > 0\}$ (Atribuire)
4. $\{(x > 3) \land \neg(x > 2)\}$ `y:=-1` $\{y > 0\}$ (3, Întărirea pre-condiției)
   deoarece $(x > 3) \land \neg(x > 2) \to -1 > 0$

Putem concluziona:

5. $\{x > 3\}$ `if x>2 then y:=1 else y:=-1` $\{y > 0\}$ (2, 4, If)

Regula pentru If

Exercițiu:

Cum ați scrie o regulă pentru o instrucțiune condițională fără else?

`if b then S`

Regula pentru While (Regula 6/6)

Regula pentru While:

$$\boxed{\dfrac{\{P \wedge b\}\ \mathtt{S}\ \{P\}}{\{P\}\ \mathtt{while\ b\ do\ S}\ \{P \wedge \neg b\}}}$$

Regula pentru While (Regula 6/6)

Regula pentru While:

$$\boxed{\frac{\{P \wedge b\}\ \mathtt{S}\ \{P\}}{\{P\}\ \mathtt{while\ b\ do\ S}\ \{P \wedge \neg b\}}}$$

- P este numit invariantul buclei
- P este adevărată înainte de a intra în buclă, și este păstrată adevărată de execuția corpului buclei, S (deși nu neapărat și în timpul execuției lui S).
- Dacă execuția buclei se temină condiția de control trebuie să fie falsă, deci $\neg b$ apare în post-condiție.
- Pentru a executa corpul buclei, S, b trebuie să fie adevărată, deci b apare în pre-condiția lui S.

Aplicarea regulii pentru While

$$\frac{\{P \land b\}\ \texttt{S}\ \{P\}}{\{P\}\ \texttt{while b do S}\ \{P \land \neg b\}} \qquad \{P\}\ \texttt{while b do S}\ \{Q\}$$

- Partea cea mai grea este descoperirea invariantului.
- Este nevoie de intuiție. Nu există algoritm care să găsească invariantul.
- Post-condiția obținută după aplicarea regulii e de forma $P \land \neg b$.
  E posibil ca post-condiția dorită, Q, să fie diferită!
- Dacă $(P \land \neg b) \to Q$, putem folosi regula de slăbire a post-condiției.

$$\frac{\dfrac{\{P \land b\}\ \texttt{S}\ \{P\}}{\{P\}\ \texttt{while b do S}\ \{P \land \neg b\}}}{\{P\}\ \texttt{while b do S}\ \{Q\}} \quad \text{deoarece } P \land \neg b \to Q$$

Regula pentru While

Exemplu

Să zicem că vrem să găsim o pre-condiție P astfel încât

$$\{P\}\ \texttt{while (n>0) do n:=n-1}\ \{n = 0\}$$

Regula pentru While

Exemplu

Să zicem că vrem să găsim o pre-condiție P astfel încât

$$\{P\}\ \texttt{while (n>0) do n:=n-1}\ \{n = 0\}$$

Vrem un invariant P astfel încât

- P este menținut adevărat de corpul buclei
- $P \land \neg(n > 0) \to (n = 0)$

Regula pentru While

Exemplu

Să zicem că vrem să găsim o pre-condiție P astfel încât

$$\{P\}\ \texttt{while (n>0) do n:=n-1}\ \{n = 0\}$$

Vrem un invariant P astfel încât

- P este menținut adevărat de corpul buclei
- $P \land \neg(n > 0) \rightarrow (n = 0)$

$P \equiv n \geq 0$ pare o alegere rezonabilă pentru un astfel de invariant.
Premiza regulii pentru while este satisfăcută de axioma de atribuire.

Regula pentru While

Exemplu (cont.)

Demonstrăm că

$$\{n \geq 0\} \ \texttt{while (n>0) do n:=n-1} \ \{n = 0\}$$

1. $\{n - 1 \geq 0\}$ `n:=n-1` $\{n \geq 0\}$ (Atribuire)

Regula pentru While

Exemplu (cont.)

Demonstrăm că

$$\{n \geq 0\}\ \texttt{while (n>0) do n:=n-1}\ \{n = 0\}$$

1. $\{n - 1 \geq 0\}$ `n:=n-1` $\{n \geq 0\}$ (Atribuire)
2. $\{n \geq 0 \land n > 0\}$ `n:=n-1` $\{n \geq 0\}$ (1, Întărirea pre-condiției)
   deoarece $n \geq 0 \land n > 0 \to n - 1 \geq 0$

Regula pentru While

Exemplu (cont.)

Demonstrăm că

$$\{n \geq 0\}\ \texttt{while (n>0) do n:=n-1}\ \{n = 0\}$$

1. $\{n - 1 \geq 0\}$ `n:=n-1` $\{n \geq 0\}$ (Atribuire)
2. $\{n \geq 0 \land n > 0\}$ `n:=n-1` $\{n \geq 0\}$ (1, Întărirea pre-condiției)
   deoarece $n \geq 0 \land n > 0 \to n - 1 \geq 0$
3. $\{n \geq 0\}$ `while (n>0) do n:=n-1` $\{n \geq 0 \land \neg(n > 0)\}$ (2, While)

Regula pentru While

Exemplu (cont.)

Demonstrăm că

$$\{n \geq 0\}\ \texttt{while (n>0) do n:=n-1}\ \{n = 0\}$$

1. $\{n - 1 \geq 0\}$ `n:=n-1` $\{n \geq 0\}$ (Atribuire)
2. $\{n \geq 0 \land n > 0\}$ `n:=n-1` $\{n \geq 0\}$ (1, Întărirea pre-condiției)
   deoarece $n \geq 0 \land n > 0 \to n - 1 \geq 0$
3. $\{n \geq 0\}$ `while (n>0) do n:=n-1` $\{n \geq 0 \land \neg(n > 0)\}$ (2, While)
4. $\{n \geq 0\}$ `while (n>0) do n:=n-1` $\{n = 0\}$ (3, Slăbirea post-condiției)
   deoarece $n \geq 0 \land \neg(n > 0) \to n = 0$

Sistemul de reguli pentru Logica Hoare

Skip: $\boxed{\{Q\}\ \texttt{skip}\ \{Q\}}$

Atribuire: $\boxed{\{Q(e)\}\ \texttt{x:=e}\ \{Q(x)\}}$

Întărire pre: $\boxed{\dfrac{\{P_w\}\ \texttt{S}\ \{Q\}}{\{P_s\}\ \texttt{S}\ \{Q\}}\quad \text{dacă } P_s \to P_w}$

Slăbire post: $\boxed{\dfrac{\{P\}\ \texttt{S}\ \{Q_s\}}{\{P\}\ \texttt{S}\ \{Q_w\}}\quad \text{dacă } Q_s \to Q_w}$

Secvențiere: $\boxed{\dfrac{\{P\}\texttt{S}_1\{Q\}\quad \{Q\}\texttt{S}_2\{R\}}{\{P\}\texttt{S}_1;\texttt{S}_2\{R\}}}$

If: $\boxed{\dfrac{\{P \wedge b\}\ \texttt{S}_1\ \{Q\}\quad \{P \wedge \neg b\}\ \texttt{S}_2\ \{Q\}}{\{P\}\ \texttt{if b then S}_1\ \texttt{else S}_2\ \{Q\}}}$

While: $\boxed{\dfrac{\{P \wedge b\}\ \texttt{S}\ \{P\}}{\{P\}\ \texttt{while b do S}\ \{P \wedge \neg b\}}}$

Un program simplu

Exemplu

Suma primelor n numere naturale impare este n^2.

Program:

```
i := 0;
s := 0;
while (i ≠ n) do
  i := i+1;
  s := s+(2*i-1)
```

Țintă: $\{\top\}$ `Program` $\{s = n^2\}$

Un program simplu

Exemplu (cont.)

Să vedem câteva exemple:

- $1 = 1 = 1^2$
- $1 + 3 = 4 = 2^2$
- $1 + 3 + 5 = 9 = 3^2$
- $1 + 3 + 5 + 7 = 16 = 4^2$

Pare să meargă. Să vedem dacă putem și demonstra!

Tintă: $\{\top\}$ `Program` $\{s = n^2\}$

Un program simplu

Exemplu (cont.)

Mai întâi, avem nevoie de un invariant P.

$$\boxed{\dfrac{\{P \wedge b\}\ \mathtt{S}\ \{P\}}{\{P\}\ \mathtt{while\ b\ do\ S}\ \{P \wedge \neg b\}}}$$

```
while (i ≠ n) do
   i := i+1;
   s := s+(2*i-1)
```

$\{s = n^2\}$

Din regula pentru while, vrem ca $P \wedge (i = n) \to (s = n^2)$, ca să putem aplica regula de slăbire a post-condiției.

La fiecare trecere prin corpul buclei, i e incrementat și s trece la următorul pătrat perfect.

Un program simplu

Exemplu (cont.)

Mai întâi, avem nevoie de un invariant P.

$$\frac{\{P \wedge b\}\ \mathtt{S}\ \{P\}}{\{P\}\ \mathtt{while\ b\ do\ S}\ \{P \wedge \neg b\}}$$

```
while (i ≠ n) do
   i := i+1;
   s := s+(2*i-1)
```
$\{s = n^2\}$

Din regula pentru while, vrem ca $P \wedge (i = n) \to (s = n^2)$, ca să putem aplica regula de slăbire a post-condiției.

La fiecare trecere prin corpul buclei, i e incrementat și s trece la următorul pătrat perfect.

Invariantul $P \equiv (s = i^2)$ pare rezonabil.

Un program simplu

Verificăm că $P \equiv (s = i^2)$ este invariant: deducem $\{P \land (i \neq n)\}$ S $\{P\}$

Un program simplu

Verificăm că $P \equiv (s = i^2)$ este invariant: deducem $\{P \wedge (i \neq n)\}$ `S` $\{P\}$

1. $\{s + (2 * i - 1) = i^2\}$ `s:=s+(2*i-1)` $\{s = i^2\}$ (Atribuire)

Un program simplu

Verificăm că $P \equiv (s = i^2)$ este invariant: deducem $\{P \wedge (i \neq n)\}$ `S` $\{P\}$

1. $\{s + (2 * i - 1) = i^2\}$ `s:=s+(2*i-1)` $\{s = i^2\}$ (Atribuire)

2. $\{s + (2 * (i + 1) - 1) = (i + 1)^2\}$ `i:=i+1` $\{s + (2 * i - 1) = i^2\}$ (Atribuire)

Un program simplu

Verificăm că $P \equiv (s = i^2)$ este invariant: deducem $\{P \wedge (i \neq n)\}$ `S` $\{P\}$

1. $\{s + (2 * i - 1) = i^2\}$ `s:=s+(2*i-1)` $\{s = i^2\}$ (Atribuire)

2. $\{s + (2 * (i + 1) - 1) = (i + 1)^2\}$ `i:=i+1` $\{s + (2 * i - 1) = i^2\}$ (Atribuire)

3. $\{s + (2 * (i + 1) - 1) = (i + 1)^2\}$ `i:=i+1; s:=s+(2*i-1)` $\{s = i^2\}$ (1, 2, Secvențiere)

Un program simplu

Verificăm că $P \equiv (s = i^2)$ este invariant: deducem $\{P \wedge (i \neq n)\}$ `S` $\{P\}$

1. $\{s + (2 * i - 1) = i^2\}$ `s:=s+(2*i-1)` $\{s = i^2\}$ (Atribuire)

2. $\{s + (2 * (i + 1) - 1) = (i + 1)^2\}$ `i:=i+1` $\{s + (2 * i - 1) = i^2\}$ (Atribuire)

3. $\{s + (2 * (i + 1) - 1) = (i + 1)^2\}$ `i:=i+1; s:=s+(2*i-1)` $\{s = i^2\}$
   (1, 2, Secvențiere)

4. $\{s = i^2 \wedge i \neq n\}$ `i:=i+1; s:=s+(2*i-1)` $\{s = i^2\}$
   (3 Întărirea pre-condiției)
   deoarece $s = i^2 \wedge i \neq n \rightarrow s + (2 * (i + 1) - 1) = (i + 1)^2$

Un program simplu

Exemplu (cont.)

Completarea demonstrației $\{\top\}$ `Program` $\{s = n^2\}$

...

4. $\{s = i^2 \land i \neq n\}$ `i:=i+1; s:=s+(2*i-1)` $\{s = i^2\}$

Un program simplu

Exemplu (cont.)

Completarea demonstrației $\{\top\}$ `Program` $\{s = n^2\}$

...

4. $\{s = i^2 \wedge i \neq n\}$ `i:=i+1; s:=s+(2*i-1)` $\{s = i^2\}$
5. $\{s = i^2\}$ `while ... s:=s+(2*i-1)` $\{s = i^2 \wedge \neg(i \neq n)\}$ (4, While)

Un program simplu

Exemplu (cont.)

Completarea demonstrației $\{\top\}$ `Program` $\{s = n^2\}$

...

4. $\{s = i^2 \land i \neq n\}$ `i:=i+1; s:=s+(2*i-1)` $\{s = i^2\}$
5. $\{s = i^2\}$ `while ... s:=s+(2*i-1)` $\{s = i^2 \land \neg(i \neq n)\}$ (4, While)
6. $\{s = i^2\}$ `while ... s:=s+(2*i-1)` $\{s = n^2\}$ (5, Slăbirea post-condiției)

deoarece $s = i^2 \land \neg(i \neq n) \to s = n^2$

Un program simplu

Exemplu (cont.)

Completarea demonstrației $\{\top\}$ `Program` $\{s = n^2\}$

...

4. $\{s = i^2 \wedge i \neq n\}$ `i:=i+1; s:=s+(2*i-1)` $\{s = i^2\}$
5. $\{s = i^2\}$ `while ... s:=s+(2*i-1)` $\{s = i^2 \wedge \neg(i \neq n)\}$ (4, While)
6. $\{s = i^2\}$ `while ... s:=s+(2*i-1)` $\{s = n^2\}$ (5, Slăbirea post-condiției)

   deoarece $s = i^2 \wedge \neg(i \neq n) \rightarrow s = n^2$
7. $\{0 = i^2\}$ `s:= 0` $\{s = i^2\}$ (Atribuire)

Un program simplu

Exemplu (cont.)

Completarea demonstrației $\{\top\}$ `Program` $\{s = n^2\}$

...

4. $\{s = i^2 \wedge i \neq n\}$ `i:=i+1; s:=s+(2*i-1)` $\{s = i^2\}$
5. $\{s = i^2\}$ `while ... s:=s+(2*i-1)` $\{s = i^2 \wedge \neg(i \neq n)\}$ (4, While)
6. $\{s = i^2\}$ `while ... s:=s+(2*i-1)` $\{s = n^2\}$ (5, Slăbirea post-condiției)

   deoarece $s = i^2 \wedge \neg(i \neq n) \rightarrow s = n^2$
7. $\{0 = i^2\}$ `s:= 0` $\{s = i^2\}$ (Atribuire)
8. $\{0 = 0^2\}$ `i:= 0` $\{0 = i^2\}$ (Atribuire)

Un program simplu

Exemplu (cont.)

Completarea demonstrației $\{\top\}$ `Program` $\{s = n^2\}$

...

4. $\{s = i^2 \wedge i \neq n\}$ `i:=i+1; s:=s+(2*i-1)` $\{s = i^2\}$
5. $\{s = i^2\}$ `while ... s:=s+(2*i-1)` $\{s = i^2 \wedge \neg(i \neq n)\}$ (4, While)
6. $\{s = i^2\}$ `while ... s:=s+(2*i-1)` $\{s = n^2\}$ (5, Slăbirea post-condiției)

   deoarece $s = i^2 \wedge \neg(i \neq n) \to s = n^2$
7. $\{0 = i^2\}$ `s:= 0` $\{s = i^2\}$ (Atribuire)
8. $\{0 = 0^2\}$ `i:= 0` $\{0 = i^2\}$ (Atribuire)
9. $\{\top\}$ `i:= 0` $\{0 = i^2\}$ (8, Întărirea pre-condiției)

   deoarece $\top \to 0 = 0^2$

Un program simplu

Exemplu (cont.)

Completarea demonstrației $\{\top\}$ `Program` $\{s = n^2\}$

...

4. $\{s = i^2 \wedge i \neq n\}$ `i:=i+1; s:=s+(2*i-1)` $\{s = i^2\}$
5. $\{s = i^2\}$ `while ... s:=s+(2*i-1)` $\{s = i^2 \wedge \neg(i \neq n)\}$ (4, While)
6. $\{s = i^2\}$ `while ... s:=s+(2*i-1)` $\{s = n^2\}$ (5, Slăbirea post-condiției)

   deoarece $s = i^2 \wedge \neg(i \neq n) \rightarrow s = n^2$
7. $\{0 = i^2\}$ `s:= 0` $\{s = i^2\}$ (Atribuire)
8. $\{0 = 0^2\}$ `i:= 0` $\{0 = i^2\}$ (Atribuire)
9. $\{\top\}$ `i:= 0` $\{0 = i^2\}$ (8, Întărirea pre-condiției)

   deoarece $\top \rightarrow 0 = 0^2$
10. $\{\top\}$ `i := 0; s := 0` $\{s = i^2\}$ (7,9, Secvențiere)

Un program simplu

Exemplu (cont.)

Completarea demonstrației $\{\top\}$ `Program` $\{s = n^2\}$

...

4. $\{s = i^2 \wedge i \neq n\}$ `i:=i+1; s:=s+(2*i-1)` $\{s = i^2\}$
5. $\{s = i^2\}$ `while ... s:=s+(2*i-1)` $\{s = i^2 \wedge \neg(i \neq n)\}$ (4, While)
6. $\{s = i^2\}$ `while ... s:=s+(2*i-1)` $\{s = n^2\}$ (5, Slăbirea post-condiției)

   deoarece $s = i^2 \wedge \neg(i \neq n) \rightarrow s = n^2$
7. $\{0 = i^2\}$ `s:= 0` $\{s = i^2\}$ (Atribuire)
8. $\{0 = 0^2\}$ `i:= 0` $\{0 = i^2\}$ (Atribuire)
9. $\{\top\}$ `i:= 0` $\{0 = i^2\}$ (8, Întărirea pre-condiției)

   deoarece $\top \rightarrow 0 = 0^2$
10. $\{\top\}$ `i := 0; s := 0` $\{s = i^2\}$ (7,9, Secvențiere)
11. $\{\top\}$ `Program` $\{s = n^2\}$ (10, 6, Secvențiere)

Bibliografie

- Lecture Notes on "Formal Methods for Software Engineering", Australian National University, Rajeev Goré.
- Mike Gordon, "Specification and Verification I", chapters 1 and 2.
- Michael Huth, Mark Ryan, "Logic in Computer Science: Modeling and Reasoning about Systems", 2nd edition, Cambridge University Press, 2004.
- Krzysztof R. Apt, Frank S. de Boer, Ernst-Rüdiger Olderog, "Verification of Sequential and Concurrent Programs", 3rd edition, Springer.