Hoareova logika

Hoareova logika

Hoareova logika či též **Floydova-Hoareova logika** je formální systém pro dokazování korektnosti programů v imperativních programovacích jazycích.

- Základním prvkem důkazů v této logice jsou tzv. **Hoareovy trojice** (**Hoare triples**) tvaru:

$$\{P\}\, c\, \{Q\}$$

kde:

- P — je tzv. **precondition**
- Q — je tzv. **postcondition**
- c — je **příkaz** daného programovacího jazyka

P a Q jsou logické formule vyjadřující se o stavech běžícího programu v daném programovacím jazyce — tyto formule se označují jako **assertions**.

Hoareovy trojice

Neformálně řečeno, Hoareova trojice

$$\{P\}\, c\, \{Q\}$$

vyjadřuje následující tvrzení:

Jestliže ve stavu, ve kterém platí podmínka P, bude proveden příkaz c a provádění tohoto příkazu po konečném počtu kroků skončí, dostane se program do stavu, ve kterém bude platit podmínka Q.

Poznámka: Jedná o tzv. **částečnou korektnost** (**partial correctness**) — není zaručeno, že provádění daného příkazu c někdy skončí, může běžet donekonečna

Hoareovy trojice

V některých variantách Hoareovy logiky se uvažuje tzv. **úplná korektnost** (**total correctness**), kdy Hoareova trojice

$$\{P\}\, c\, \{Q\}$$

vyjadřuje následující význam:

Jestliže ve stavu, ve kterém platí podmínka P, bude proveden příkaz c, provádění tohoto příkazu po konečném počtu kroků skončí a program se dostane do stavu, ve kterém bude platit podmínka Q.

Poznámka: V tomto kurzu se budeme soustředit primárně na jednodušší a základnější případ, kdy se uvažuje pouze částečná korektnost.

Hoareovy trojice

Příklady Hoareových trojic:

$$\{X < 5\}\ X := X + 1\ \{X < 6\}$$

$$\{(1 \leq X) \wedge (X \leq n)\}\ X := X - 1\ \{(0 \leq X) \wedge (X < n)\}$$

$$\{(X = m) \wedge (Y = n)\}\ T := X;\ X := Y;\ Y := T\ \{(X = n) \wedge (Y = m)\}$$

Assertions

Formule P a Q v Hoareově trojici $\{P\}\, c\, \{Q\}$ představují tzv. **assertions**:

- z formálního hlediska jsou to **predikáty** na stavech
  (tj. jsou vyhodnocovány na **stavech** systému, kterým přiřazují pravdivostní hodnotu)
- vymezují **podmnožinu** stavů, ve kterých je daná podmínka splněna
- mohou obsahovat:
  - logické spojky a kvantifikátory
  - libovolné matematické funkce a predikáty
    (např. libovolné aritmetické operace, relace na číslech, atd.)
  - proměnné programu
  - „logické“ proměnné reprezentující libovolné matematické objekty

Assertions

Proměnné programu:

- jejich hodnota je daná daným stavem, nad kterým je formule P vyhodnocována
- není možné je používat jako proměnné vázané kvantifikátory

Logické proměnné:

- mohou se vyskytovat jako proměnné vázené kvantifikátory
- pokud se stejná logická proměnná vyskytuje jako volná proměnná v precondition P i postcondition Q dané Hoareovy trojice

$$\{P\}\, c\, \{Q\}$$

je to chápáno tak, že v P i Q nabývá tato proměnná stejnou hodnotu a že je implicitně vázaná univerzálním kvantifikátorem přes celou Hoareovu trojici.

Assertions

Abychom tyto dva druhy proměnných odlišili, budeme používat následující konvenci:

- **proměnné programu** — budeme je označovat velkými písmeny (jako X, Y, Z, ...)
- **logické proměnné** — budeme je označovat malými písmeny (jako x, y, z, ...)

Hoareovy trojice

Matematickou notací můžeme formálně zapsat to, že ve stavu σ platí podmínka vyjádřená formulí P, následujícím způsobem:

$$\sigma \models P$$

Význam Hoareovy trojice $\{P\}\, c\, \{Q\}$ pak můžeme vyjádřit následovně:

Pro každé stavy σ a σ' platí, že pokud:

- $\sigma \rightsquigarrow c \Downarrow \sigma'$ (tj. když provedením příkazu c přejde program ze stavu σ do stavu σ')
- $\sigma \models P$

pak $\sigma' \models Q$

Implikace

Kromě Hoareových trojic se v důkazech v rámci Hoareovy logiky vyskytují **implikace** tvaru:

$$P \Rightarrow Q$$

- Nejedná se o jednu formuli, ale o vyjádření relace mezi dvěma formulemi.
- Implikace $P \Rightarrow Q$ vyjadřuje tvrzení, že v každém stavu, ve kterém platí P, platí i Q.

  Formálně:

  V každém stavu σ, kde platí $\sigma \models P$, platí i $\sigma \models Q$.

Důkazy v Hoareově logice

Důkazy v Hoareově logice si můžeme představovat jako stromy, jejichž jednotlivé vrcholy jsou označeny:

- Hoareovými trojicemi tvaru $\{P\}\, c\, \{Q\}$
- Imlikacemi tvaru $P \Rightarrow Q$

Přičemž pro tento strom platí:

- Kořen stromu je dole a je označen Hoareovou trojicí reprezentující chování celého programu.
- Označení každého vrcholu musí odpovídat jednomu kroku důkazu, který je proveden podle některého z pravidel odpovídajících jednotlivým konstrukcím jazyka.

Pravidla Hoareovy logiky

Pokud chceme dokázat platnost dané Hoareovy trojice

$$\{P\}\, c\, \{Q\}$$

použijeme pravidlo, které závisí na tom, jak konkrétně vypadá příkaz c.

- Typicky platí, že pro každý konkrétní příkaz máme k dispozici právě jedno pravidlo.
- Některá pravidla nemají žádné premisy, některá mají jednu nebo více premis tvaru $\{P'\}\, c'\, \{Q'\}$ nebo $P' \Rightarrow Q'$.
- Tvar formulí P' a Q' a příkazů c' v těchto premisách může být u některých pravidel jednoznačně určen na základě formulí P a Q a příkazu c v původní trojici.
- Ne vždy tomu tak ale je.

Příkaz **skip**

Pravidlo pro příkaz `skip` je velmi jednoduché:

$$\{P\}\,\texttt{skip}\,\{P\}$$

Příkaz `skip` nemění stav (nemění hodnotu žádné proměnné).

Pokud tedy daná podmínka P platila před provedením tohoto příkazu, bude platit i po něm.

Sekvence

Pravidlo pro sekvenci tvořenou příkazy c_1 a c_2:

$$\frac{\{P\}\, c_1\, \{Q\} \qquad \{Q\}\, c_2\, \{R\}}{\{P\}\, c_1\ ;\ c_2\, \{R\}}$$

Poznámka: Všimněte si, že pokud chceme dokázat platnost trojice

$$\{P\}\, c_1\ ;\ c_2\, \{R\}$$

pomocí výše uvedeného pravidla, konkrétní tvar formule Q není touto výslednou trojicí jednoznačně určen.

Přiřazení

Jedním z nejdůležitějších pravidel Hoareovy logiky je pravidlo pro **přiřazení**:

$$\{P[a/X]\}\ X := a\ \{P\}$$

Zápis $P[a/X]$ v tomto pravidle označuje formuli, která vznikne z formule P, když za výskyty proměnné X dosadíme výraz a.

Tato operace na formulích se označuje jako **substituce**.

Příklad:

$$\{X + 1 > 5\}\ X := X + 1\ \{X > 5\}$$

Výsledkem substituce $(X > 5)[(X + 1)/X]$ je formule $(X + 1 > 5)$.

Přiřazení

Alternativní pravidla pro přiřazení:

$$\{\, P \wedge (X = m) \,\}\ X := a\ \{\, P[m/X] \wedge (X = (a[m/X])) \,\}$$

$$\{\, P \,\}\ X := a\ \{\, \exists m,\ P[m/X] \wedge (X = (a[m/X])) \,\}$$

Ve výše uvedených pravidlech je m nová logická proměnná, která se nevyskytuje ve formuli m.

(Protože se jedná o logickou proměnnou, nevyskytuje se ani ve výrazu a, protože ten může obsahovat jen proměnné programu.)

Přiřazení

Příklad:

$$\{(X > 4) \wedge X = m\}\ X := X + 1\ \{(m > 4) \wedge (X = m + 1)\}$$

$$\{X > 4\}\ X := X + 1\ \{\exists m, (m > 4) \wedge (X = m + 1)\}$$

Poznámka: Obě výše uvedená alternativní pravidla pro přiřazení je možno odvodit ze standardního Hoareova pravidla pro přiřazení.

Naopak toto standardní pravidlo je možno odvodit z libovolného z těchto alternativních pravidel.

(Při těchto odvozeních je třeba použít kromě daných pravidel pro přiřazení také pravidla pro konsekvenci, která budou popsána za chvíli.)

Konsekvence

Ne vždy jsou formule v Hoareových trojicích, které vyjdou jako výsledek použití pravidel pro jednotlivé příkazy, přesně ve tvaru, který je co nejjednodušší, nejpřirozenější nebo který by se hodil pro další kroky důkazu.

Formule v Hoareových trojicích je možno změnit pomocí následujících pravidel pro konsekvence:

- **Zesílení precondition:**

$$\frac{P \Rightarrow P' \qquad \{P'\}\, c\, \{Q\}}{\{P\}\, c\, \{Q\}}$$

- **Zeslabení postcondition:**

$$\frac{\{P\}\, c\, \{Q'\} \qquad Q' \Rightarrow Q}{\{P\}\, c\, \{Q\}}$$

Konsekvence

Obě výše uvedená pravidla je možné zkombinovat do jednoho společného pravidla:

$$\frac{P \Rightarrow P' \qquad \{P'\}\, c\, \{Q'\} \qquad Q' \Rightarrow Q}{\{P\}\, c\, \{Q\}}$$

- Toto pravidlo je možné odvodit z výše uvedených dvou pravidel.
- Naopak z tohoto zkombinovaného pravidla je možné odvodit daná dvě pravidla.

  (Použitím faktu, že pro libovolnou formuli P platí $P \Rightarrow P$.)

Poznámka: Všimněte si, že všechna výše uvedená pravidla pro konsekvence zahrnují jako speciální případ i případy, kdy jsou formule P a P' (resp. Q a Q') logicky ekvivalentní.

Příkaz **if**

Pravidlo pro příkaz **if** vypadá následovně:

$$\frac{\{P \wedge b\}\ c_1\ \{Q\} \qquad \{P \wedge \neg b\}\ c_2\ \{Q\}}{\{P\}\ \texttt{if}\ b\ \texttt{then}\ c_1\ \texttt{else}\ c_2\ \{Q\}}$$

Poznámka: Podmínka b je zde použita zároveň jako podformule.

Příkaz **while**

Pravidlo pro příkaz **while** vypadá následovně:

$$\frac{\{P \wedge b\}\ c\ \{P\}}{\{P\}\ \texttt{while}\ b\ \texttt{do}\ c\ \{P \wedge \neg b\}}$$

Poznámka: Formule P se označuje jako **invariant** daného cyklu.

Existenční kvantifikátory

Někdy může být užitečné také následující pravidlo:

$$\frac{\{P\}\ c\ \{Q\}}{\{\exists x, P\}\ c\ \{\exists x, Q\}}$$

Pravidla Hoareovy logiky

Pravidla Hoareovy logiky není třeba brát jako axiomy:

- Platnost těchto pravidel je možné dokázat z definované sémantiky jazyka Imp a definice toho, kdy platí Hoareovy trojice.
- Důkazy korektnosti těchto pravidel jsou typicky jediným místem, kde je třeba se explicitně odkazovat na sémantiku daného jazyka.
- Při následném používání daných pravidel pro vytváření důkazů korektnosti konkrétních programů pak typicky není třeba se dále na tuto sémantiku odvolávat.

Anotované programy

Zápis důkazů v Hoareově logice ve formě stromů nebo ve formě posloupnosti Hoareových trojic je i pro velice malé programy poměrně nešikovný a ne příliš přehledný:

- stromy jsou hodně velké
- zápis příkazů i jednotlivých formulí může být dost dlouhý

O něco elegantnější je zápis důkazů ve formě **anotovaného programu**:

- Vypadá jako běžný zápis programu, kde jsou mezi jednotlivé řádky programu doplněny formule, reprezentující podmínky, které mají v daném místě platit.

Anotované programy

$\{\,\mathsf{True}\,\} \Rightarrow$

$\{\, a = b \cdot 0 + a \,\}$

$X := a$

$\{\, a = b \cdot 0 + X \,\}$

$Y := 0$

$\{\, a = b \cdot Y + X \,\}$

while $b \leq X$ **do**

$\{\, a = b \cdot Y + X \;\wedge\; b \leq X \,\} \Rightarrow$

$\{\, a = b \cdot (Y + 1) + (X - b) \,\}$

$X := X - b$

$\{\, a = b \cdot (Y + 1) + X \,\}$

$Y := Y + 1$

$\{\, a = b \cdot Y + X \,\}$

end

$\{\, a = b \cdot Y + X \;\wedge\; \neg(b \leq X) \,\} \Rightarrow$

$\{\, a = b \cdot Y + X \;\wedge\; X < b \,\}$

Anotované programy

- Z anotovaného programu je možné snadno extrahovat:
  - původní (neanotovaný) program
  - strukturu stromu důkazu
  - množinu implikací tvaru $P \Rightarrow Q$, jejichž platnost je třeba dokázat, aby byl důkaz hotov
- Ve skutečnosti ani není obecně třeba uvést všechny anotace:
  - Je možné vytvořit variantu anotovaných programů, kde je třeba uvádět anotace jen na některých „klíčových“ místech kódu
    — typicky na začátku, na konci a invarianty cyklů
  - všechny ostatní anotace se pak dají dopočítat automaticky

Implementace Hoareovy logiky v Coqu

V Coqu mohou být assertions reprezentovány například následujícím typem:

Definition Assertion := state → **Prop**

kde state je datový typ reprezentující stavy běžícího programu.

- Pro vyjádření toho, že ve stavu st platí podmínka P (tj. st $\models P$), kde
  - P je typu Assertion (tj. P : Assertion)
  - st je typu state (tj. st : state)

  můžeme pak v Coqu použít následující zápis:

P st

Hoareovy trojice v Coqu

Hoareovy trojice mohou být reprezentovány v Coqu jako následující ternární predikát:

```
Definition hoare_triple
           (P : Assertion) (c : com) (Q : Assertion) : Prop :=
  ∀ st st' : state,
    st =[ c ]⇒ st'  →  P st  →  Q st'.
```

Navíc místo zápisu

```
hoare_triple P c Q
```

můžeme používat následující uživatelsky definovanou notaci:

```
{{P}} c {{Q}}
```

Assertions reprezentované v Coqu

V principu je sice možné definovat assertion P zcela libovolně, například jako výraz tvaru

$$\textbf{fun}\ (\mathrm{st} : \mathrm{state}) \Rightarrow \quad \cdots$$

a explicitně se v něm odkazovat na daný stav st.

V praxi se ale něčemu takovému chceme typicky vyhnout a místo toho zapisovat tyto predikáty jako běžné **logické formule**:

- V takovém případě je typicky potřeba definovat nové verze všech logických spojek a kvatifikátorů, které budou pracovat s hodnotami typu Assertion, nikoli typu **Prop** jako běžné spojky a kvantifikátory.
- Pomocí uživatelsky definovaných notací a koercí je možné dosáhnout toho, že zápis assertions vypadá velmi podobně jako běžně zapisovaná tvrzení v Coqu.

Pravidla Hoareovy logiky v Coqu

V Coqu mohou být jednotlivá pravidla dokázána jako teorémy:

- Například pravidlo pro příkaz `skip` může být vyjádřeno následovně:

Theorem hoare_skip : $\forall P$: Assertion,
$\{\{P\}\}$ `skip` $\{\{P\}\}$.

Platnost tohoto teorému je možné dokázat použitím definice sémantiky příkazu `skip` v jazyce Imp a definice hoare_triple.