Logika dla informatyków

Andrzej M. Borzyszkowski

Instytut Informatyki
Uniwersytet Gdański

inf.ug.edu.pl/~amb

Weryfikacja software'u – logika Hoare'a

Weryfikacja software'u

- specyfikacja R w języku naturalnym
- przekształcona w formułę φ_R logiki
- program P w języku programowania
- taki, że P spełnia φ_R
- dowód spełnialności

Język programowania:

- wyrażenia arytmetyczne:
  $E ::= n \mid x \mid (-E) \mid (E + E) \mid (E - E) \mid (E * E)$
- wyrażenia boolowskie:
  $B ::= \text{true} \mid \text{false} \mid (E < E) \mid (!B) \mid (B \,\&\, B) \mid (B \mid B)$
- programy: $C ::= x = E \mid C; C \mid \text{if } B \text{ then } \{C\} \text{ else } \{C\} \mid \text{while } B \ \{C\}$

Notacja T. Hoare'a

- stan programu = wartościowanie zmiennych
- znaczenie programu = przekształcanie wartościowania zmiennych
- tj. przy danym stanie wyjściowym otrzymujemy stan końcowy
- osobno trzeba dyskutować problem *czy* stan końcowy zostanie osiągnięty – problem stopu
- notacja $(\!|\varphi|\!)\ P\ (\!|\psi|\!)$ znaczy: program P, startując w stanie spełniającym φ, kończy w stanie spełniającym ψ
- np. $(\!|x > 0|\!)\ P\ (\!|(y{-}1) \cdot (y{-}1) < x \leq y \cdot y|\!)$ oznacza, że program P oblicza pierwiastek kwadratowy z liczby dodatniej x zaokrąglony w górę, pierwiastek zapamiętany jest w zmiennej y

Definicja

Stan ℓ spełnia formułę φ, jeśli $\mathcal{M} \models_\ell \varphi$ w modelu $\mathcal{M}$ dla liczb naturalnych z wartościowaniem zmiennych ℓ

Poprawność częściowa i całkowita

Definicja

Poprawność częściowa: "program P, startując w stanie spełniającym φ, kończy w stanie spełniającym ψ", o ile osiągnie stan końcowy, oznaczenie $\models_{par} (\!|\varphi|\!)\ P\ (\!|\psi|\!)$

- program, który się zapętla jest z definicji poprawny – nie wyprodukuje fałszywego wyniku
- np. $(\!|\,|\!)$ while true {x=0} $(\!|0 = 1|\!)$

Definicja

Poprawność całkowita: "program P, startując w stanie spełniającym φ, kończy w stanie spełniającym ψ", a stan końcowy na pewno osiąga, oznaczenie $\models_{tot} (\!|\varphi|\!)\ P\ (\!|\psi|\!)$

- czyli poprawność częściowa plus warunek stopu
- program pętlący się nie spełnia żadnej specyfikacji

Przykład: silnia po raz pierwszy

- Poniższy program *Sil*1 liczy silnię x, odczytujemy wynik w y, liczby są całkowite

```
y=1;
z=0;
while (z!=x){
  z=z+1;
  y=y*z;
}
```

- ale program ten nie zwróci żadnego wyniku, jeśli $x < 0$
- $(\!|\,|\!)$ *Sil*1 $(\!|y = x!|\!)$ jest prawdziwe w semantyce częściowej poprawności
- $(\!|x \geq 0|\!)$ *Sil*1 $(\!|y = x!|\!)$ jest prawdziwe w semantyce całkowitej poprawności
- gdyby warunek w pętli był sformułowany $z < x$, to program kończyłby działanie zawsze, dla wartości $x < 0$ kończyłby działanie z $y = 1$

Reguły dla częściowej i całkowitej poprawności

- Celem będzie zdefiniowanie reguł takich, by wyprowadzalne były zdania o częściowej i całkowitej poprawności:
  $\vdash_{par} (\!|\varphi|\!)\ P\ (\!|\psi|\!)$
  $\vdash_{tot} (\!|\varphi|\!)\ P\ (\!|\psi|\!)$
- Pożądane własności:
  - zdrowość (poprawność reguł):
    jeśli $\vdash_{par} (\!|\varphi|\!)\ P\ (\!|\psi|\!)$, to $\models_{par} (\!|\varphi|\!)\ P\ (\!|\psi|\!)$
    jeśli $\vdash_{tot} (\!|\varphi|\!)\ P\ (\!|\psi|\!)$, to $\models_{tot} (\!|\varphi|\!)\ P\ (\!|\psi|\!)$
  - pełność: przeciwne implikacje
  - dowód poprawności reguł polega na sprawdzeniu każdej reguły (i jest łatwy)
  - dowód pełności reguł jest o wiele trudniejszy

Problem ze zmiennymi

- Program *Sil*2 też liczy silnię x, odczytujemy wynik w y

```
y=1;
while (x!=0){
  y=y*x;
  x=x−1
}
```

- ale na zakończenie zmienna x będzie miała wartość 0
- $(\!|x \geq 0|\!)$ *Sil*2 $(\!|y = x!|\!)$ jest po prostu fałszem – y jest silnią *oryginalnej* wartości x, a tymczasem $x = 0$.
- trzeba oryginalną wartość x zapamiętać:
  $\models_{tot} (\!|x \geq 0 \wedge x = x_0|\!)$ *Sil*2 $(\!|y = x_0!|\!)$
- def.: *zmienna logiczna*: zmienna występująca w formule Hoare'a, nie związana kwantyfikatorem, i nie występująca w programie.

Reguły logiki Hoare'a dla częściowej poprawności

- złożenie:
$$\frac{(\!|\varphi|\!)\ P_1\ (\!|\eta|\!) \qquad (\!|\eta|\!)\ P_2\ (\!|\psi|\!)}{(\!|\varphi|\!)\ P_1; P_2\ (\!|\psi|\!)}$$
- przypisanie:
$$\frac{}{(\!|\psi[E/x]|\!)\ x = E\ (\!|\psi|\!)}$$
- w formułach nie powinno się używać kwantyfikatora wiążącego zmienne inne niż logiczne
- wówczas podstawienie w formule jest bezproblemowe
- $\models_{tot} (\!|2 = 2|\!)\ x = 2\ (\!|x = 2|\!)$
- $\models_{tot} (\!|2 = y|\!)\ x = 2\ (\!|x = y|\!)$
- $\models_{tot} (\!|2 = 3|\!)\ x = 2\ (\!|x = 3|\!)$ dla każdego stanu, że $2 = 3$ program kończy działanie itd. Implikacja spełniona jest w próżni.

Reguły logiki Hoare'a c.d.

- instrukcja warunkowa:
$$\frac{(\!|\varphi \wedge B|\!)\ P_1\ (\!|\psi|\!) \qquad (\!|\varphi \wedge \neg B|\!)\ P_2\ (\!|\psi|\!)}{(\!|\varphi|\!)\ \text{if } B \text{ then } \{P_1\} \text{ else } \{P_2\}\ (\!|\psi|\!)}$$
- pętla:
$$\frac{(\!|\varphi \wedge B|\!)\ P\ (\!|\varphi|\!)}{(\!|\varphi|\!)\ \text{while } B\ \{P\}\ (\!|\varphi \wedge \neg B|\!)}$$
- częściowa poprawność: jeśli program while B $\{P\}$ zakończy działanie, to będzie spełniony warunek $\neg B$
- nie możemy zagwarantować całkowitej poprawności, warunek wyjścia z pętli nie musi zostać osiągnięty
- φ jest *niezmiennikiem* pętli (tzn. jeśli doszło do wykonania ciała pętli, to warunek się nie zmienił)

Reguła wiążąca logiki predykatów i Hoare'a

- $$\frac{\vdash_A \varphi' \to \varphi \qquad (\!|\varphi|\!)\ P\ (\!|\psi|\!) \qquad \vdash_A \psi \to \psi'}{(\!|\varphi'|\!)\ P\ (\!|\psi'|\!)}$$
gdzie $\vdash_A$ oznacza wyprowadzalność w logice predykatów dotyczącej arytmetyki.

Zapis dowodów w logice Hoare'a

- każdy dowód logiki można zapisać jako drzewo
- można zapisać w postaci linearnej
- ale nawet wówczas fragmenty programu byłyby wielkrotnie kopiowane – w każdej regule przesłanki i konkluzja zawierają te same elementy
- idea: nie kopiować elementów programistycznych, a formuły logiczne wstawiać do środka programu, program traktuje je jako komentarze
- np. reguła dla złożenia pozwala na zapis:

$(\!|\varphi|\!)$
$P_1;$
$(\!|\eta|\!)$
P_2
$(\!|\psi|\!)$

- reguła dla wiążąca z logiką predykatów pozwala na zapis

$(\!|\varphi'|\!)$
$(\!|\varphi|\!)$
P
$(\!|\psi|\!)$
$(\!|\psi'|\!)$

Zapis dowodów w logice Hoare'a c.d.

Definicja

Dla danego programu P i formuły ψ najsłabszym warunkiem wstępnym (weakest precondition) jest takie φ, że

- $(\!|\varphi|\!)\ P\ (\!|\psi|\!)$, *oraz*
- *jeśli* $(\!|\varphi'|\!)\ P\ (\!|\psi|\!)$, *to* $\varphi' \to \varphi$

- podstawowym składnikiem programu jest przypisanie, reguła dla przypisania "działa wstecz"
- np. dla przypisania $x = E \quad wp(\psi) = \psi[E/x]$
- aby udowodnić $(\!|\varphi|\!)\ P\ (\!|\psi|\!)$ wystarczy udowodnić $\varphi \to wp(\psi)$

Przykłady z przypisaniem

$(\!|y < 3|\!)$
$(\!|y + 1 < 4|\!)$
$y = y + 1$
$(\!|y < 4|\!)$

$(\!|\ |\!)$
$(\!|x + y = x + y|\!)$
$z = x;$
$(\!|z + y = x + y|\!)$
$z = z + y;$
$(\!|z = x + y|\!)$
$u = z;$
$(\!|u = x + y|\!)$

Instrukcja warunkowa

- szukamy nasłabszego φ tak by $(\!|\varphi|\!)$ if B then $\{P_1\}$ else $\{P_2\}$ $(\!|\psi|\!)$
- załóżmy, że $(\!|\varphi_1|\!)\ P_1\ (\!|\psi|\!)$ oraz $(\!|\varphi_2|\!)\ P_2\ (\!|\psi|\!)$
- wówczas $\varphi = (B \to \varphi_1) \wedge (\neg B \to \varphi_2)$ jest najsłabszym warunkiem dla instrukcji warunkowej
- odpowiada do dokładniej regule pochodnej dla instrukcji warunkowej

$$\frac{(\!|\varphi_1|\!)\ P_1\ (\!|\psi|\!) \qquad (\!|\varphi_2|\!)\ P_2\ (\!|\psi|\!)}{(\!|(B \to \varphi_1) \wedge (\neg B \to \varphi_2)|\!)\ \text{if } B \text{ then } \{P_1\} \text{ else } \{P_2\}\ (\!|\psi|\!)}$$

Pętla

- jest to jedyne miejsce, w którym trzeba dokonać wysiłku umysłowego – *nie ma* automatycznego wyliczenia jednej formuły logicznej z innej
- celem jest dowód $(\!|\varphi|\!)$ while B $\{P\}$ $(\!|\psi|\!)$
- trzeba znaleźć niezmiennik η taki, że
  - $\varphi \to \eta$
  - $(\!|\eta \wedge B|\!)\ P\ (\!|\eta|\!)$
  - $\eta \wedge \neg B \to \psi$

Dowód poprawności programu liczącego silnię

- $(\!|x \geq 0|\!)$ *Sil*1 $(\!|y = x!|\!)$
- ciałem pętli jest

```
z = z + 1;
y = y * z;
```

- niezmiennikiem pętli jest formuła $y = z!$ (nie potrzebujemy nawet dodatkowe warunku wejścia do pętli)
- jedną konieczną implikacją jest by w połączeniu z warunkiem wyjścia z pętli otrzymać $y = x!$
- z drugiej strony niezmiennik musi być być prawdziwy w stanie *przed* wejściem do pętli

Dowód dla silni c.d.

$(\!|1 = 0!|\!)$
$y = 1;$
$(\!|y = 0!|\!)$
$z = 0;$
$(\!|y = z!|\!)$ niezmiennik pętli trzeba wymyśleć
while$(z \mathrel{!}= x)\{$
$(\!|y = z! \wedge z \neq x|\!)$
$(\!|y \cdot (z+1) = (z+1)!|\!)$
$z = z + 1;$
$(\!|y \cdot z = z!|\!)$
$y = y * z;$
$(\!|y = z!|\!)$ i musi zachodzić na końcu ciała pętli
$\}$
$(\!|y = z! \wedge \neg(z \neq x)|\!)$
$(\!|y = x!|\!)$

Największy wspólny dzielnik

obliczamy $NWD(a, b)$
while $b : a, b = b, a\%b$; *return* a

wszystkie liczby są naturalne

niezmiennikiem pętli jest $N = NWD(a, b)$

implikacja na początku pętli zachodzi ponieważ dla $b > 0$ $NWD(a, b) = NWD(b, a\%b)$

końcowa implikacja zachodzi ponieważ $NWD(a, 0) = a$

$(\!|N = NWD(a, b)|\!)$
while$(b)\{$
$(\!|N = NWD(a, b) \wedge b > 0|\!)$
$(\!|N = NWD(b, a\%b)|\!)$
$c = a;$
$(\!|N = NWD(b, c\%b)|\!)$
$a = b;$
$(\!|N = NWD(a, c\%b)|\!)$
$b = c\%b;$
$(\!|N = NWD(a, b)|\!)$
$\}$
$(\!|N = NWD(a, b) \wedge b = 0|\!)$
$(\!|N = a|\!)$

Szybkie potęgowanie

obliczamy b^n, wynik w r

$r = 1;$ *while*$(n)\{$
if $(n\%2)\{n - -; r = b * r;\}$
else $\{n = n/2; b = b * b;\}$
$\}$

warunek *if* jest niekonieczny, dla liczb parzystych ten wzór też działa

warunek *else* jest istotny, parzystość n gwarantuje, że $n = 2 \cdot (n/2)$ i zachodzi implikacja

niezmiennikiem jest $Pow = r \cdot b^n$

$(\!|Pow = b^n|\!)$
$r = 1;$
$(\!|Pow = r \cdot b^n|\!)$
while$(n)\{$ $(\!|Pow = r \cdot b^n \wedge n \neq 0|\!)$
if $(n\%2)$ $(\!|Pow = r \cdot b^n \wedge n\%2 \neq 0|\!)$
$(\!|Pow = (b \cdot r) \cdot b^{n-1}|\!)$
$\{n - -;$ $(\!|Pow = (b \cdot r) \cdot b^n|\!)$
$r = b * r;\}$ $(\!|Pow = r \cdot b^n|\!)$
else $(\!|Pow = r \cdot b^n \wedge n\%2 = 0|\!)$
$(\!|Pow = r \cdot (b^2)^{n/2}|\!)$
$\{n = n/2;$ $(\!|Pow = r \cdot (b \cdot b)^n|\!)$
$b = b * b;\}$ $(\!|Pow = r \cdot b^n|\!)$
$(\!|Pow = r \cdot b^n|\!)$
$\}$
$(\!|Pow = r \cdot b^n \wedge n = 0|\!)$
$(\!|Pow = r|\!)$

Całkowita poprawność

- jedynym problemem jest pętla i możliwość nieskończonych obliczeń bez wyjścia z pętli
- dowód, że pętla musi się skończyć polega na wprowadzeniu licznika/ogranicznika
- czyli taka wielkość, że maleje z każdym wykonaniem ciała pętli i ma tylko skończenie wiele możliwych wartości
- w konkretnych przykładach nie muszą to być liczby ale np. podrzewa, podformuły i wiele innych sytuacji, w których działa dowód przez indukcję

$$\frac{(\!|\eta \wedge B \wedge 0 \leq E = E_0|\!)\ P\ (\!|\eta \wedge 0 \leq E < E_0|\!)}{(\!|\eta \wedge 0 \leq E|\!)\ \text{while}\ B\ \{P\}\ (\!|\eta \wedge \neg B|\!)}$$

Całkowita poprawność c.d.

- w przykładzie *Sil*1 wielkością zmieniającą się jest $x - z$
- program *Sil*1 nie zapętla sie jedynie przy założeniu, że $x \geq 0$
- dla NWD zmniejsza się $\max(a, b)$ jeśli liczby były dodatnie
- dla potęgowania zmniejsza się n jeśli na początku $n \geq 0$
- *nie istnieje* automatyczny sposób sprawdzania, czy program gwarantuje zakończenie działań (problem stopu)
- np. program Collatza

```
c=x;
while (c!=1){
  if (c%2==0) {c=c/2;}
  else {c=3*c+1;}
}
```

- nie wiadomo, czy $\models_{tot} (\!|x > 0|\!)\ \mathit{Collatz}\ (\!|\,|\!)$