

Wstęp do Programowania potok funkcyjny

Marcin Kubica

2010/2011

Outline

- 1 Logika Hoare'a
 - Background
 - While-programy
 - Logika Hoare'a

Kilka podstawowych pojęć

Definition

- Programy imperatywne zmieniają **stan**, czyli wartości zmiennych.
- **Asercja** = warunek logiczny, który musi być spełniony w danym momencie obliczeń / w określonym miejscu w programie.
- **Warunek końcowy** = asercja opisująca stan po wykonaniu programu.
- **Warunek początkowy** = asercja opisująca stan przed wykonaniem programu.
- **Specyfikacja programu** = warunek początkowy + warunek końcowy.

Kilka podstawowych pojęć

Definition

- **Częściowa poprawność** = warunek początkowy + obliczenie kończy się bez błędów \implies warunek końcowy.
- **Własność stopu** = warunek początkowy \implies obliczenie kończy się bez błędów.
- **Całkowita poprawność** = Częściowa poprawność + własność stopu.

Formalizmy

Definition

Formalizmy do specyfikacji i weryfikacji programów imperatywnych:

- Logika Hoare'a — formuły postaci:
 $\{\text{warunek początkowy}\} \text{program} \{\text{warunek końcowy}\}$
oznaczają słabą poprawność,
- Transformatory predykatów Dijkstry — przekształcają warunek końcowy w najślabszy warunek początkowy,
- Logika dynamiczna (V. R. Pratt, D. Harel) — rozszerzenie logiki modalnej, $[p]\varphi$, $\langle p \rangle \varphi$.
- Logika algorytmiczna (G. Mirkowska, A. Salwicki) — program $\{\text{warunek końcowy}\}$; oznacza najślabszy warunek początkowy gwarantujący całkowitą poprawność.

While-programy

Definition

- Ograniczamy się do języka while-programów.
- Zakładamy, że wszystkie „zmiennne” są referencjami do liczb całkowitych.
(Rozszerzenie na inne typy proste nie jest trudne, ale komplikuje technalia.)
- Zakładamy, że każdy identyfikator reprezentuje inną referencję.
- Aliasing, lub typy złożone (tablice, wskaźniki) istotnie komplikuje problem specyfikacji i weryfikacji programów.

While-programy

Definition

Składnia while-programów:

$$\begin{aligned}
 \langle \text{while-program} \rangle &::= \langle \text{instrukcja} \rangle \mid \\
 &\quad \langle \text{instrukcja} \rangle \text{ ; } \langle \text{while-program} \rangle \\
 \langle \text{instrukcja} \rangle &::= \underline{\text{begin}} \langle \text{while-program} \rangle \underline{\text{end}} \mid \\
 &\quad \underline{\text{if}} \langle \text{warunek} \rangle \underline{\text{then}} \langle \text{instrukcja} \rangle \\
 &\quad \quad \quad [\underline{\text{else}} \langle \text{instrukcja} \rangle] \mid \\
 &\quad \underline{\text{while}} \langle \text{warunek} \rangle \\
 &\quad \quad \underline{\text{do}} \langle \text{while-program} \rangle \underline{\text{done}} \mid \\
 &\quad \langle \text{referencja} \rangle \text{ := } \langle \text{wyrażenie} \rangle \mid \\
 &\quad \underline{() } \\
 \langle \text{wyrażenie} \rangle &::= \text{wyrażenia arytmetyczne ze zmiennymi} \\
 \langle \text{warunek} \rangle &::= \text{warunki logiczne z } \langle \text{wyrażeniami} \rangle
 \end{aligned}$$

Semantyka while-programów

Definition

Dziedziny semantyczne:

- Id — zbiór identyfikatorów,
- Stan systemu, to wartościowanie zmiennych, funkcja przypisująca nazwom zmiennych ich wartości:
 $Stan = [Id \rightarrow \mathbb{Z}]$,
- Semantyka while-programu lub instrukcji to funkcja częściowa ze stanów w stany. (Obliczenia mogą się zapętlać, lub nie udać.)
 $\llbracket \text{while-program} \rrbracket : Stan \mapsto Stan$
- Semantyka wyrażenia, to funkcja ze stanu w liczby całkowite:
 $\llbracket \text{wyrażenie} \rrbracket : Stan \mapsto \text{int}$
- Semantyka warunku, to funkcja ze stanu w wartości logiczne:
 $\llbracket \text{warunek} \rrbracket : Stan \mapsto \text{bool}$

Semantyka while-programów

Definition

Semantyka while-programów:

$$\begin{aligned}
 \llbracket () \rrbracket(S) &= S \\
 \llbracket r := w \rrbracket(S) &= \text{fun } i \rightarrow \text{if } i = r \text{ then } \llbracket w \rrbracket(S) \text{ else } S(i) \\
 \llbracket P_1; P_2 \rrbracket(S) &= \llbracket P_2 \rrbracket(\llbracket P_1 \rrbracket(S)) \\
 \llbracket \text{if } w \text{ then } P_1 \text{ else } P_2 \rrbracket(S) &= \\
 &= \text{if } \llbracket w \rrbracket(S) \text{ then } \llbracket P_1 \rrbracket(S) \text{ else } \llbracket P_2 \rrbracket(S) \\
 \llbracket \text{if } w \text{ then } P \rrbracket &= \llbracket \text{if } w \text{ then } P \text{ else } () \rrbracket \\
 \llbracket \text{while } w \text{ do } P \text{ done} \rrbracket(S_0) &= S_n, \text{ gdzie: } \text{not } \llbracket w \rrbracket(S_n) \text{ ,} \\
 &\llbracket w \rrbracket(S_i), S_{i+1} = \llbracket P \rrbracket(S_i) \text{ dla } 0 \leq i < n \\
 &\text{w p.p. nieokreślone}
 \end{aligned}$$

Więcej na *Semantyce i weryfikacji programów*.

Formuły logiki Hoare'a

Definition

- Formuły Hoare'owskie to trójki postaci: $\{\varphi\}P\{\psi\}$, gdzie:
 - φ — warunek początkowy,
 - P — while-program,
 - ψ — warunek końcowy.
- $\{\varphi\}P\{\psi\}$ oznacza częściową poprawność P względem φ i ψ .
- Model = dziedzina algorytmiczna.
- Wynikanie semantyczne ze zbioru przesłanek: $A \models \{\varphi\}P\{\psi\}$.
- Możliwość udowodnienia na podstawie zbioru przesłanek:
 $A \vdash \{\varphi\} P \{\psi\}$

System dowodzenia

Definition

Aksjomaty:

- instrukcji pustej:

$$A \vdash \{\varphi\} () \{\varphi\}$$

- instrukcji przypisania

$$A \vdash \{\varphi[!x/w]\} x := w \{\varphi\}$$

Reguły:

- osłabiania.

$$\frac{A \models \varphi \implies \varphi', \quad A \vdash \{\varphi'\} P \{\psi'\}, \quad A \models \psi' \implies \psi}{A \vdash \{\varphi\} P \{\psi\}}$$

System dowodzenia

Definition

Reguły:

- średnika:

$$\frac{A \vdash \{\varphi\} P \{\xi\}, \quad A \vdash \{\xi\} Q \{\psi\}}{A \vdash \{\varphi\} P; Q \{\psi\}}$$

- instrukcji warunkowej:

$$\frac{A \vdash \{\varphi \wedge \alpha\} P \{\psi\}, \quad A \vdash \{\varphi \wedge \neg\alpha\} Q \{\psi\}}{A \vdash \{\varphi\} \text{ if } \alpha \text{ then } P \text{ else } Q \{\psi\}}$$

- pętli while:

$$\frac{A \vdash \{\varphi \wedge \alpha\} P \{\varphi\}}{A \vdash \{\varphi\} \text{ while } \alpha \text{ do } P \text{ done } \{\varphi \wedge \neg\alpha\}}$$

System dowodzenia

Definition

Wzmocnione reguły gwarantujące własność stopu:

- instrukcji przypisania

$$\frac{A \models \varphi[!x/w] \implies w \text{ jest określone}}{A \vdash \{\varphi[!x/w]\} x := w \{\varphi\}}$$

- instrukcji warunkowej:

$$\frac{A \models \varphi \implies \alpha \text{ jest określone,} \\ A \vdash \{\varphi \wedge \alpha\} P \{\psi\}, \quad A \vdash \{\varphi \wedge \neg\alpha\} Q \{\psi\}}{A \vdash \{\varphi\} \text{ if } \alpha \text{ then } P \text{ else } Q \{\psi\}}$$

System dowodzenia

Definition

Wzmocnione reguły gwarantujące własność stopu:

- pętli while:

$$\begin{array}{l}
 A \models \varphi \implies \alpha \text{ jest określone, } A \models (\varphi \wedge \mu \leq 0) \implies \neg\alpha, \\
 n \text{ nie występuje w } P, \quad A \vdash \{\varphi \wedge \alpha \wedge \mu = n\} P \{\varphi \wedge \mu < n\} \\
 \hline
 A \vdash \{\varphi\} \text{ while } \alpha \text{ do } P \text{ done } \{\varphi \wedge \neg\alpha\}
 \end{array}$$

Minimum

Example

Oto program obliczający minimum z dwóch liczb:

$$\begin{aligned} & \{!x = !x_0 \wedge !y = !y_0\} \\ & \text{if } !x < !y \text{ then } z := !x \text{ else } z := !y \\ & \{!z = \min(!x_0, !y_0)\} \end{aligned}$$

Z reguły dla instrukcji warunkowej mamy dwie formuły do udowodnienia:

- 1 $\{!x = !x_0 \wedge !y = !y_0 \wedge !x < !y\} z := !x \{!z = \min(!x_0, !y_0)\}$
- 2 $\{!x = !x_0 \wedge !y = !y_0 \wedge \neg(!x < !y)\} z := !y \{!z = \min(!x_0, !y_0)\}$

Minimum

Example

$$\textcircled{1} \{!x = !x_0 \wedge !y = !y_0 \wedge !x < !y\} z := !x \{!z = \min(!x_0, !y_0)\}$$

Korzystając z:

$$(!x = !x_0 \wedge !y = !y_0 \wedge !x < !y) \implies !x = \min(!x_0, !y_0)$$

oraz reguły osłabiania otrzymujemy:

$$\{!x = \min(!x_0, !y_0)\} z := !x \{!z = \min(!x_0, !y_0)\}$$

co jest aksjomatem przypisania.

$$\textcircled{2} \{!x = !x_0 \wedge !y = !y_0 \wedge \neg(!x < !y)\} z := !y \{!z = \min(!x_0, !y_0)\}$$

Korzystając z reguły osłabiania oraz z tego, że:

$$(!x = !x_0 \wedge !y = !y_0 \wedge \neg(!x < !y)) \implies !y = \min(!x_0, !y_0)$$

otrzymujemy:

$$\{!y = \min(!x_0, !y_0)\} z := !y \{!z = \min(!x_0, !y_0)\}$$

co jest aksjomatem przypisania. □

Algorytm Euklidesa

Example

Specyfikacja:

$$\{!x > 0 \wedge !x = !x_0 \wedge !y > 0 \wedge !y = !y_0\} E \{!x = NWD(!x_0, !y_0)\}$$

Implementacja:

$$E \left\{ \begin{array}{l} \text{while } !x \neq !y \text{ do} \\ \quad \left\{ \begin{array}{l} \text{if } !x > !y \text{ then} \\ \quad x := !x - !y \\ \text{else} \\ \quad y := !y - !x \end{array} \right. \end{array} \right.$$

Algorytm Euklidesa

Example

- Pokażemy pełną poprawność.
- Zauważmy, że wszystkie wyrażenia i warunki są dobrze określone.
- Dodatkowe przesłanki — tylko w regule dla pętli `while`.
- Zaczynamy od zastosowania reguły dla pętli `while`:
 - Funkcja miary: $!x+!y$.
 - Niezmiennik:

$$\varphi \equiv !x > 0 \wedge !y > 0 \wedge NWD(!x, !y) = NWD(!x_0, !y_0)$$
 - $(\varphi \wedge !x+!y \leq 0) \implies !x = !y$ — trywialnie spełnione.
 - Trzeba pokazać:

$$\{\varphi \wedge !x \neq !y \wedge !x+!y = n\} \text{ I } \{\varphi \wedge !x+!y < n\}$$

Algorytm Euklidesa

Example

$$\{\varphi \wedge !x \neq !y \wedge !x + !y = n\} \text{ I } \{\varphi \wedge !x + !y < n\}$$

Stosując regułę dla if-then-else, mamy dwa przypadki:

- 1 $\{\varphi \wedge !x \neq !y \wedge !x + !y = n \wedge !x > !y\}$
 $x := !x - !y$
 $\{\varphi \wedge !x + !y < n\}$
- 2 $\{\varphi \wedge !x \neq !y \wedge !x + !y = n \wedge \neg(!x > !y)\}$
 $y := !y - !x$
 $\{\varphi \wedge !x + !y < n\}$

Algorytm Euklidesa

Example

$$\textcircled{1} \quad \{\varphi \wedge !x \neq !y \wedge !x + !y = n \wedge !x > !y\}$$

$$x := !x - !y$$

$$\{\varphi \wedge !x + !y < n\}$$

Oslabiamy do aksjomatu przypisania:

$$\left\{ \begin{array}{l} !x - !y > 0 \wedge !y > 0 \wedge !x - !y + !y < n \wedge \\ NWD(!x - !y, !y) = NWD(!x_0, !y_0) \end{array} \right\}$$

$$x := !x - !y$$

$$\{\varphi \wedge !x + !y < n\}$$

korzystając z implikacji:

$$\left(\begin{array}{l} \varphi \wedge !x \neq !y \wedge \\ !x + !y = n \wedge !x > !y \end{array} \right) \Rightarrow \left(\begin{array}{l} !x - !y > 0 \wedge !y > 0 \wedge \\ NWD(!x - !y, !y) = NWD(!x_0, !y_0) \wedge \\ !x - !y + !y < n \end{array} \right)$$

Algorytm Euklidesa

Example

$$\begin{aligned} & \textcircled{2} \quad \{\varphi \wedge !x \neq !y \wedge !x + !y = n \wedge \neg(!x > !y)\} \\ & \quad y := !y - !x \\ & \quad \{\varphi \wedge !x + !y < n\} \end{aligned}$$

Oslabiamy do aksjomatu przypisania:

$$\begin{aligned} & \left\{ \begin{array}{l} !x > 0 \wedge !y - !x > 0 \wedge !x + !y - !x < n \wedge \\ NWD(!x, !y - !x) = NWD(!x_0, !y_0) \end{array} \right\} \\ & \quad y := !y - !x \\ & \quad \{\varphi \wedge !x + !y < n\} \end{aligned}$$

korzystając z implikacji:

$$\left(\begin{array}{l} \varphi \wedge !x \neq !y \wedge \\ !x + !y = n \wedge \neg(!x > !y) \end{array} \right) \implies \left(\begin{array}{l} !x > 0 \wedge !y - !x > 0 \wedge \\ NWD(!x, !y - !x) = NWD(!x_0, !y_0) \wedge \\ !x + !y - !x < n \end{array} \right)$$



Algorytm mnożenia rosyjskich chłopów

Example

Specyfikacja:

$$\{!x = !x_0 \wedge !x \geq 0 \wedge !y = !y_0 \wedge !y \geq 0\} P \{!z = !x_0 \cdot !y_0\}$$

Implementacja:

$$P \left\{ \begin{array}{l} z := 0; \\ \text{while } !x > 0 \text{ do} \\ \quad \left\{ \begin{array}{l} \text{if } !x \bmod 2 = 0 \text{ then begin} \\ \quad x := !x/2; \\ \quad y := !y \cdot 2 \\ \quad \text{end else begin} \\ \quad \quad x := !x - 1; \\ \quad \quad z := !z + !y \\ \quad \quad \text{end} \end{array} \right. \\ \text{done} \end{array} \right.$$

Algorytm mnożenia rosyjskich chłopów

Example

- Pokażemy pełną poprawność.
- Wyrażenia oraz warunki są zawsze dobrze określone.
- Dodatkowe przesłanki — tylko w regule dla pętli `while`.
- Niezmiennik pętli `while`:

$$\varphi \equiv !x \cdot !y + !z = !x_0 \cdot !y_0 \wedge !x \geq 0 \wedge !y \geq 0$$

Algorytm mnożenia rosyjskich chłopów

Example

$$\{!x = !x_0 \wedge !x \geq 0 \wedge !y = !y_0 \wedge !y \geq 0\} P \{!z = !x_0 \cdot !y_0\}$$

Z reguły dla średnika uzyskujemy dwie formuły do udowodnienia:

$$\textcircled{1} \{!x = !x_0 \wedge !x \geq 0 \wedge !y = !y_0 \wedge !y \geq 0\} z := 0 \{\varphi\}$$

Możemy osłabić do aksjomatu przypisania:

$$\{!x \cdot !y + 0 = !x_0 \cdot !y_0 \wedge !x \geq 0 \wedge !y \geq 0\} z := 0 \{\varphi\}$$

$$\textcircled{2} \{\varphi\} W \{!z = !x_0 \cdot !y_0\}$$

Algorytm mnożenia rosyjskich chłopów

Example

$$\textcircled{2} \quad \{\varphi\} W \{!z = !x_0 \cdot !y_0\}$$

Zauważmy, że $(\varphi \wedge !x \leq 0) \implies (!x = 0 \wedge !z = !x_0 \cdot !y_0)$.
 Osłabiamy do: $\{\varphi\} W \{\varphi \wedge !x \leq 0\}$.

Stosujemy regułę dla pętli while:

- Funkcja miary: $!x$.
- $(\varphi \wedge !x \leq 0) \implies \neg(!x > 0)$.
- $\{\varphi \wedge !x > 0 \wedge !x = n\} I \{\varphi \wedge !x < n\}$

Algorytm mnożenia rosyjskich chłopów

Example

$$\{\varphi \wedge !x > 0 \wedge !x = n\} \text{ I } \{\varphi \wedge !x < n\}$$

Stosując regułę instrukcji warunkowej dostajemy dwie formuły:

A) $\{\varphi \wedge !x > 0 \wedge !x = n \wedge !x \bmod 2 = 0\}$

$$x := !x/2; y := !y \cdot 2$$
$$\{\varphi \wedge !x < n\}$$

B) $\{\varphi \wedge !x > 0 \wedge !x = n \wedge !x \bmod 2 \neq 0\}$

$$x := !x - 1; z := !z + !y$$
$$\{\varphi \wedge !x < n\}$$

Algorytm mnożenia rosyjskich chłopów

Example

$$\text{A) } \{ \varphi \wedge !x > 0 \wedge !x = n \wedge !x \bmod 2 = 0 \}$$

$$x := !x/2; y := !y \cdot 2$$

$$\{ \varphi \wedge !x < n \}$$

Z reguły dla średnika sprowadzamy do aksjomatu przypisania:

$$\{ !x \cdot 2 \cdot !y + !z = !x_0 \cdot !y_0 \wedge !x \geq 0 \wedge 2 \cdot !y \geq 0 \wedge !x < n \}$$

$$y := !y \cdot 2$$

$$\{ \varphi \wedge !x < n \}$$

oraz:

$$\{ \varphi \wedge !x > 0 \wedge !x = n \wedge !x \bmod 2 = 0 \}$$

$$x := !x/2$$

$$\{ !x \cdot 2 \cdot !y + !z = !x_0 \cdot !y_0 \wedge !x \geq 0 \wedge 2 \cdot !y \geq 0 \wedge !x < n \}$$

Algorytm mnożenia rosyjskich chłopów

Example

$$\{\varphi \wedge !x > 0 \wedge !x = n \wedge !x \bmod 2 = 0\}$$

$$x := !x/2$$

$$\{!x \cdot 2 \cdot !y + !z = !x_0 \cdot !y_0 \wedge !x \geq 0 \wedge 2 \cdot !y \geq 0 \wedge !x < n\}$$

Korzystamy z implikacji:

$$\left(\begin{array}{l} \varphi \wedge !x > 0 \wedge \\ !x = n \wedge \\ !x \bmod 2 = 0 \end{array} \right) \implies \left(\begin{array}{l} (!x/2) \cdot 2 \cdot !y + !z = !x_0 \cdot !y_0 \wedge \\ !x/2 \geq 0 \wedge 2 \cdot !y \geq 0 \wedge !x/2 < n \end{array} \right)$$

Oślabiamy do aksjomatu przypisania:

$$\{(!x/2) \cdot 2 \cdot !y + !z = !x_0 \cdot !y_0 \wedge !x/2 \geq 0 \wedge 2 \cdot !y \geq 0 \wedge !x/2 < n\}$$

$$x := !x/2$$

$$\{!x \cdot 2 \cdot !y + !z = !x_0 \cdot !y_0 \wedge !x \geq 0 \wedge 2 \cdot !y \geq 0 \wedge !x < n\}$$

Algorytm mnożenia rosyjskich chłopów

Example

B) $\{\varphi \wedge !x > 0 \wedge !x = n \wedge !x \bmod 2 \neq 0\}$
 $x := !x - 1; z := !z + !y$
 $\{\varphi \wedge !x < n\}$

Z reguły dla średnika, otrzymujemy aksjomat przypisania:

$$\{!x \cdot !y + !z + !y = !x_0 \cdot !y_0 \wedge !x \geq 0 \wedge !y \geq 0 \wedge !x < n\}$$

$$z := !z + !y$$

$$\{\varphi \wedge !x < n\}$$

oraz:

$$\{\varphi \wedge !x > 0 \wedge !x = n \wedge !x \bmod 2 \neq 0\}$$

$$x := !x - 1$$

$$\{!x \cdot !y + !z + !y = !x_0 \cdot !y_0 \wedge !x \geq 0 \wedge !y \geq 0 \wedge !x < n\}$$

Algorytm mnożenia rosyjskich chłopów

Example

$$\{\varphi \wedge !x > 0 \wedge !x = n \wedge !x \bmod 2 \neq 0\}$$

$$x := !x - 1$$

$$\{!x \cdot !y + !z + !y = !x_0 \cdot !y_0 \wedge !x \geq 0 \wedge !y \geq 0 \wedge !x < n\}$$

Korzystając z implikacji:

$$\left(\begin{array}{l} \varphi \wedge !x > 0 \wedge \\ !x = n \wedge \\ !x \bmod 2 \neq 0 \end{array} \right) \implies \left(\begin{array}{l} (!x - 1) \cdot !y + !z + !y = !x_0 \cdot !y_0 \wedge \\ !x - 1 \geq 0 \wedge !y \geq 0 \wedge !x - 1 < n \end{array} \right)$$

osłabiamy do aksjomatu przypisania:

$$\{(!x - 1) \cdot !y + !z + !y = !x_0 \cdot !y_0 \wedge !x - 1 \geq 0 \wedge !y \geq 0 \wedge !x - 1 < n\}$$

$$x := !x - 1$$

$$\{!x \cdot !y + !z + !y = !x_0 \cdot !y_0 \wedge !x \geq 0 \wedge !y \geq 0 \wedge !x < n\} \quad \square$$

Deser

