Wstęp do Programowania potok funkcyjny

Marcin Kubica

2010/2011

Outline

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 $\Longrightarrow$ warunek końcowy.
- **Własność stopu** = warunek początkowy $\Longrightarrow$ 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:
  {warunek początkowy} program {warunek końcowy}
  oznaczają słabą poprawność,
- Transformatory predykatów Dijkstry — przekształcają warunek końcowy w najsłabszy 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 {warunek końcowy};
  oznacza najsłabszy warunek początkowy gwarantujący całkowitą poprawność.

While-programy

Definition

- Ograniczamy się do języka while-programów.
- Zakładamy, że wszystkie „zmienne" są referencjami do liczb całkowitych.
  (Rozszerzenie na inne typy proste nie jest trudne, ale komplikuje technikalia.)
- 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:

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

Semantyka while-programów

Definition

Dziedziny semantyczne:

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

Semantyka while-programów

Definition

Semantyka while-programów:

$$
\begin{aligned}
[\![()]\!](S) &= S \\
[\![r := w]\!](S) &= \texttt{fun } i \to \texttt{if } i = r \texttt{ then } [\![w]\!](S) \texttt{ else } S(i) \\
[\![P_1; P_2]\!](S) &= [\![P_2]\!]([\![P_1]\!](S)) \\
[\![\texttt{if } w \texttt{ then } P_1 \texttt{ else } P_2]\!](S) &= \\
&= \texttt{if } [\![w]\!](S) \texttt{ then } [\![P_1]\!](S) \texttt{ else } [\![P_2]\!](S) \\
[\![\texttt{if } w \texttt{ then } P]\!] &= [\![\texttt{if } w \texttt{ then } P \texttt{ else } ()]\!]
\end{aligned}
$$

$$
[\![\texttt{while } w \texttt{ do } P \texttt{ done}]\!](S_0) = S_n, \text{ gdzie : } \begin{array}{l} \texttt{not } [\![w]\!](S_n) \\ [\![w]\!](S_i),\ S_{i+1} = [\![P]\!](S_i) \text{ dla } 0 \leq i < n \\ \text{w p.p. nieokreślone} \end{array},
$$

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 podstawiew zbioru przesłanek: $A \vdash \{\varphi\}\ \mathtt{P}\ \{\psi\}$

System dowodzenia

Definition

Aksjomaty:

- instrukcji pustej:

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

- instrukcji przypisania

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

Reguły:

- osłabiania.

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

System dowodzenia

Definition

Reguły:

- średnika:

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

- instrukcji warunkowej:

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

- pętli while:

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

System dowodzenia

Definition

Wzmocnione reguły gwarantujące własność stopu:

- instrukcji przypisania

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

- instrukcji warunkowej:

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

System dowodzenia

Definition

Wzmocnione reguły gwarantujące własność stopu:

- pętli while:

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

Minimum

Example

Oto program obliczający minimum z dwóch liczb:

$$\{!x = !x_0 \wedge !y = !y_0\}$$
$$\texttt{if } !x < !y \texttt{ then } z := !x \texttt{ else } z := !y$$
$$\{!z = \min(!x_0, !y_0)\}$$

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

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.
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:

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

Implementacja:

$$E \begin{cases} \texttt{while } !x \neq !y \texttt{ do} \\ \quad I \begin{cases} \texttt{if } !x > !y \texttt{ then} \\ \quad\quad x := !x - !y \\ \texttt{else} \\ \quad\quad y := !y - !x \end{cases} \end{cases}$$

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

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

Algorytm Euklidesa

Example

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

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

1. $\{\varphi \wedge !x \neq !y \wedge !x+!y = n \wedge !x > !y\}$
   $x := !x - !y$
   $\{\varphi \wedge !x+!y < n\}$

   Osłabiamy do aksjomatu przypisania:

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

   korzystając z implikacji:

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

Algorytm Euklidesa

Example

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

   Osłabiamy do aksjomatu przypisania:

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

   korzystając z implikacji:

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

□

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:

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

Algorytm mnożenia rosyjskich chłopów

Example

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

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

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:

1. $\{!x = !x_0 \wedge !x \geq 0 \wedge !y = !y_0 \wedge !y \geq 0\}\ z := 0\ \{\varphi\}$
   Możemy osłabić do aksjomatu przypisania:

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

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

Algorytm mnożenia rosyjskich chłopów

Example

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

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

Algorytm mnożenia rosyjskich chłopów

Example

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

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

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

$$\{\varphi \wedge !x > 0 \ \wedge \ \ \wedge !x = n ! x \bmod 2 = 0\}$$
$$x := x/2$$
$$\{!x \cdot 2 \cdot !y + !z = !x_0 \cdot !y_0 \wedge !x \geq 0 \ \wedge \ 2 \cdot !y \geq 0 \wedge !x < n\}$$

Korzystamy z implikacji:

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

Osłabiamy 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:

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

osłabiamy do aksjomatu przypisania:

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