

Математические методы верификации схем и программ

mk.cs.msu.ru → Лекционные курсы
→ Математические методы верификации схем и программ

Блок 2

Общие принципы дедуктивной верификации программ

Модельные императивные программы:
синтаксис,
операционная семантика

Лектор:

Подымов Владислав Васильевич
E-mail:
valdus@yandex.ru

Принципы дедуктивной верификации программ

1. Программа π вычисляет отношение R_π между данными, которые подаются на вход и получаются на выходе
2. Текст программы π — это формальное описание отношения R_π
3. Спецификация Φ программы — это формальное описание отношения R_Φ между данными программы
 - ▶ Отношение, описываемое спецификацией, — это требования, которым должно удовлетворять отношение, вычисляемое программой
4. Формальная верификация программы π относительно спецификации Φ — это строгое доказательство того, что программа π удовлетворяет требованиям Φ , то есть доказательство включения $R_\pi \subseteq R_\Phi$

Принципы дедуктивной верификации программ

Чтобы уметь формально верифицировать программы, нужно:

1. Строго описать,
 - ▶ какие записи мы считаем программами
(синтаксис программ)
 - ▶ как программы преобразуют входные данные в выходные
(семантику программ)
2. Выбрать формальный язык описания требований к программам
3. Предложить метод проверки того, удовлетворяет ли заданная программа предъявленным к ней требованиям

Напоминание: логика предикатов (сигнатура)

Сигнатурой логики предикатов $\langle Const, Func, Pred \rangle$ состоит из

- ▶ множества констант $Const$
- ▶ множества функциональных символов $Func$
(символов операций)
- ▶ множества предикатных символов $Pred$
(символов отношений)

Каждый функциональный и предикатный символ имеет вид $s^{(n)}$, где
 $n \in \mathbb{N} = \{1, 2, \dots\}$ — местность символа

Местность часто опускается в записи символа s

Var — счётное множество переменных

Напоминание: логика предикатов (термы, формулы)

Форма Бэкуса-Наура (**БНФ**), задающая синтаксис **термов** (выражений) и **формул** (условий, или булевых выражений):

$$\begin{aligned} t & ::= x \mid c \mid f(t_1, \dots, t_n), \\ \varphi & ::= P(t_1, \dots, t_n) \mid (\neg\varphi) \mid (\varphi \& \varphi) \mid (\varphi \vee \varphi) \mid \\ & \quad (\varphi \rightarrow \varphi) \mid (\exists x \varphi) \mid (\forall x \varphi), \end{aligned}$$

где φ — **формула**, t, t_1, \dots, t_n — **термы**, $P^{(n)} \in Pred$, $x \in Var$, $c \in Const$ и $f^{(n)} \in Func$

Term — множество всех термов (заданной сигнатуры)

Примеры

(сигнатура: $\langle \{3\}, \{+(^2), .(^2)\}, \{=(^2)\} \rangle$)

- ▶ $x + 3 \cdot y$ — терм в инфиксной записи
- ▶ $+(x, \cdot(3, y))$ — терм в функциональной записи
- ▶ $x + 3 \cdot y = 2 + z$ — формула в инфиксной записи
- ▶ $=(+((x, \cdot(3, y)), +(2, z)))$ — формула в функциональной записи

Напоминание: логика предикатов (термы, формулы)

Приоритеты логических операций в порядке убывания:

\exists , \forall и \neg ; затем $\&$; затем \vee ; затем \rightarrow

Ещё немного формул:

$$\forall x \exists y (y = x + 1)$$

$$\exists x \forall y P(x, y) \rightarrow \forall y \exists x P(x, y)$$

Императивные программы: синтаксис

Синтаксис императивных программ (сигнатуры σ) задаётся следующей формой Бэкуса-Наура:

$$\begin{aligned}\pi &::= instr \mid instr \pi \\ instr &::= \emptyset \mid \\ &\quad x := t; \mid \\ &\quad \text{if } C \text{ then } \pi \text{ else } \pi \text{ fi } \mid \\ &\quad \text{while } C \text{ do } \pi \text{ od}\end{aligned}$$

Здесь

- ▶ π — программа
- ▶ $instr$ — инструкция
(сверху вниз в БНФ:
пустая инструкция, присваивание, ветвление, цикл)
- ▶ x — переменная
- ▶ t — терм
- ▶ C — условие: формула, не содержащая кванторов

Императивные программы: операционная семантика

Два основных способа определения семантики программ:

денотационный и операционный

Денотационный:

- ▶ Значение каждой части программы — это математический объект: denotation (денотация, денотат)
- ▶ Денотация программы задаётся как композиция денотаций её составных частей
- ▶ Денотационной семантикой не определяется способ вычисления денотации на входных данных
- ▶ Хорошо подходит для описания значения функциональных программ
- ▶ Денотации функциональных программ — это функции

Императивные программы: операционная семантика

Два основных способа определения семантики программ:

денотационный и операционный

Операционный:

- ▶ Вычисление программы — это последовательное изменение состояния вычисления
- ▶ Программой задаётся отношение переходов, описывающее, какое состояние вычисления будет получено следующим из произвольного текущего состояния
- ▶ Значение программы — это функция преобразования входных данных в выходные, определяемая на основе транзитивного замыкания отношения переходов
- ▶ Хорошо подходит для описания значения императивных программ

Напоминание: логика предикатов (подстановки)

Подстановка — это отображение $\theta : Var \rightarrow Term$, такое что лишь для конечного числа переменных x верно неравенство $\theta(x) \neq x$

Связка — это выражение вида x/t , где $x \in Var$ и $t \in Term$

Подстановку можно задать конечным множеством связок:

$\theta = \{x_1/t_1, \dots, x_n/t_n\}$ — это подстановка следующего вида:

- ▶ $\theta(x_i) = t_i$ для $1 \leq i \leq k$
- ▶ $\theta(y) = y$ для $y \notin \{x_1, \dots, x_k\}$

Напоминание: логика предикатов

Свободные и связанные переменные

Связанное вхождение переменной в формулу — это вхождение переменной в область действия квантора, связывающего эту переменную

Свободное вхождение переменной — это вхождение, не являющееся связанным

Свободная переменная формулы — это переменная, имеющая свободное вхождение в формулу

Напоминание: логика предикатов

Применение подстановок, композиция подстановок

$E\theta$ — это результат применения подстановки θ к выражению E , то есть выражение, получающееся из E заменой

- ▶ каждого вхождения каждой переменной x на терм $\theta(x)$, если E — терм
- ▶ каждого свободного вхождения каждой свободной переменной x на терм $\theta(x)$, если E — формула

Например,

$$\begin{aligned}(x + x * y * z)\{x/y, y/z + 2\} &\equiv y + y * (z + 2) * z \\(x = y \vee \exists x (x = y))\{x/1, y/2\} &\equiv 1 = 2 \vee \exists x (x = 2)\end{aligned}$$

Композиция подстановок θ и η — это подстановка $\theta\eta$, такая что для любой переменной x верно равенство

$$x(\theta\eta) = (x\theta)\eta$$

Напоминание: логика предикатов

Интерпретации

Интерпретация (сигнатуры $\langle \text{Const}, \text{Func}, \text{Pred} \rangle$) состоит из

- ▶ предметной области D (множества предметов)
- ▶ оценки констант
 - ▶ оценка константы c — это предмет $\bar{c} \in D$
- ▶ оценки функциональных символов
 - ▶ оценка функционального символа $f^{(k)}$ — это функция $\bar{f} : D^k \rightarrow D$
- ▶ оценки предикатных символов
 - ▶ оценка предикатного символа $P^{(k)}$ — это предикат $\bar{P} : D^k \rightarrow \{\text{t}, \text{f}\}$

Например, предметная область (целочисленной) арифметической интерпретации \mathcal{I}_{ar} — все целые числа, и все символы в этой интерпретации оцениваются естественным образом:

$$\overline{2+2} \equiv 4$$

Напоминание: логика предикатов

Истинность формул

Формула φ истинна в интерпретации \mathcal{I} ($\mathcal{I} \models \varphi$), если имеет значение \top при

- ▶ оценке всех символов согласно интерпретации \mathcal{I} ,
- ▶ естественной трактовке логических связок и кванторов и
- ▶ замене всех свободных вхождений каждой свободной переменной на любой предмет

Например,

$$\mathcal{I}_{ar} \models x = x$$

$$\mathcal{I}_{ar} \not\models x = 1$$

$$\mathcal{I}_{ar} \models \forall x \exists y (y = x + 1)$$

$$\mathcal{I}_{ar} \models \exists y (y = x + 1)$$

$$\mathcal{I}_{ar} \not\models \exists y (x = 2 * y)$$

Императивные программы: операционная семантика

Состояние управления — это произвольная программа

Состояние данных (оценка переменных) — это подстановка, отображающая каждую переменную программы в терм, не содержащий переменных

Состояние вычисления — это пара $\langle \pi, \theta \rangle$, где π — состояние управления, а θ — состояние данных

Отношение переходов → на множестве состояний вычисления для программы π в интерпретации \mathcal{I} определяется так:

- ▶ $\langle x := t; | \theta \rangle \rightarrow \langle \emptyset | \{x/t\}\theta \rangle$
- ▶ если $\mathcal{I} \models C\theta$, то
 $\langle \text{if } C \text{ then } \pi_1 \text{ else } \pi_2 \text{ fi} | \theta \rangle \rightarrow \langle \pi_1 | \theta \rangle$
- ▶ если $\mathcal{I} \not\models C\theta$, то
 $\langle \text{if } C \text{ then } \pi_1 \text{ else } \pi_2 \text{ fi} | \theta \rangle \rightarrow \langle \pi_2 | \theta \rangle$

Императивные программы: операционная семантика

Состояние управления — это произвольная программа

Состояние данных (оценка переменных) — это подстановка, отображающая каждую переменную программы в терм, не содержащий переменных

Состояние вычисления — это пара $\langle \pi, \theta \rangle$, где π — состояние управления, а θ — состояние данных

Отношение переходов → на множестве состояний вычисления для программы π в интерпретации \mathcal{I} определяется так:

- ▶ если $\mathcal{I} \models C\theta$, то
 $\langle \text{while } C \text{ do } \pi \text{ od} \mid \theta \rangle \rightarrow \langle \pi \text{ while } C \text{ do } \pi \text{ od} \mid \theta \rangle$
- ▶ если $\mathcal{I} \not\models C\theta$, то
 $\langle \text{while } C \text{ do } \pi \text{ od} \mid \theta \rangle \rightarrow \langle \emptyset \mid \theta \rangle$
- ▶ если $\langle \pi_1 \mid \theta \rangle \rightarrow \langle \pi'_1 \mid \eta \rangle$, то
 $\langle \pi_1 \pi_2 \mid \theta \rangle \rightarrow \langle \pi'_1 \pi_2 \mid \eta \rangle$
- ▶ $\langle \emptyset \pi \mid \theta \rangle \rightarrow \langle \pi \mid \theta \rangle$

Императивные программы: операционная семантика

Трасса программы π на оценке θ в интерпретации \mathcal{I} — это последовательность состояний вычисления вида

$$\langle \pi \mid \theta \rangle \rightarrow \langle \pi_1 \mid \theta_1 \rangle \rightarrow \langle \pi_2 \mid \theta_2 \rangle \rightarrow \dots$$

Вычисление программы π на оценке θ в интерпретации \mathcal{I} — это максимальная по длине трасса π на θ в \mathcal{I}

Результат конечного вычисления программы π на оценке θ в интерпретации \mathcal{I} — это оценка данных последнего состояния вычисления π на θ в \mathcal{I}

Иными словами, если $\langle \pi \mid \theta \rangle \rightarrow^* \langle \emptyset \mid \eta \rangle$, то η — результат вычисления π на θ в \mathcal{I}

(\rightarrow^* — транзитивное замыкание отношения \rightarrow)

Императивные программы: пример

π : while $x > 0$ do $x := x - 1$; od

$$\theta = \{x/1\}$$

Вычисление π на θ в \mathcal{I}_{ar} :

$$\begin{aligned} & \langle \text{while } x > 0 \text{ do } x := x - 1; \text{ od} \mid \{x/1\} \rangle \\ & \quad \downarrow \\ & \langle x := x - 1; \text{while } x > 0 \text{ do } x := x - 1; \text{ od} \mid \{x/1\} \rangle \end{aligned}$$

Пояснение:

$$\begin{aligned} (x > 0) \{x/1\} &= (1 > 0) \\ \mathcal{I} &\models (1 > 0) \end{aligned}$$

Императивные программы: пример

π : while $x > 0$ do $x := x - 1$; od

$$\theta = \{x/1\}$$

Вычисление π на θ в \mathcal{I}_{ar} :

$$\begin{array}{c} \langle \text{while } x > 0 \text{ do } x := x - 1; \text{ od} \mid \{x/1\} \rangle \\ \downarrow \\ \langle x := x - 1; \text{while } x > 0 \text{ do } x := x - 1; \text{ od} \mid \{x/1\} \rangle \\ \downarrow \\ \langle \emptyset \text{ while } x > 0 \text{ do } x := x - 1; \text{ od} \mid \{x/1 - 1\} \rangle \end{array}$$

Пояснение:

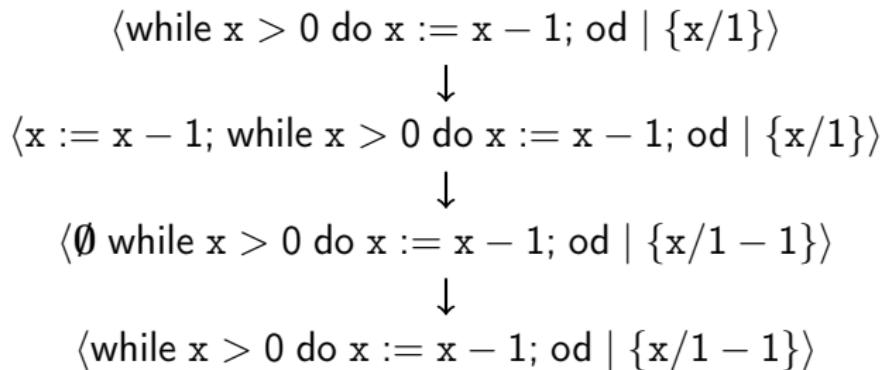
$$\{x/x - 1\}\{x/1\} = \{x/1 - 1\}$$

Императивные программы: пример

π : while $x > 0$ do $x := x - 1$; od

$\theta = \{x/1\}$

Вычисление π на θ в \mathcal{I}_{ar} :



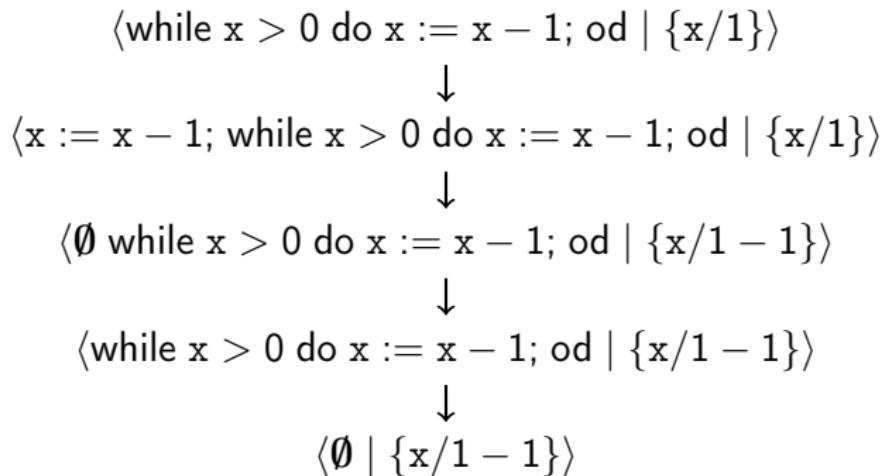
Пояснение:

Императивные программы: пример

π : while $x > 0$ do $x := x - 1$; od

$$\theta = \{x/1\}$$

Вычисление π на θ в \mathcal{I}_{ar} :



Пояснение:

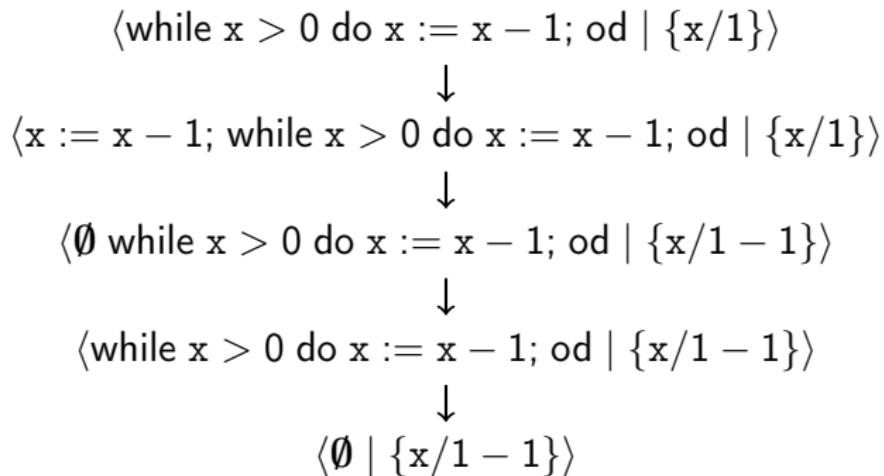
$$\begin{aligned} (x > 0) \{x/1 - 1\} &= (1 - 1 > 0) \\ \mathcal{I} &\not\models (1 - 1 > 0) \end{aligned}$$

Императивные программы: пример

π : while $x > 0$ do $x := x - 1$; od

$\theta = \{x/1\}$

Вычисление π на θ в \mathcal{I}_{ar} :



Пояснение:

$\{x/1 - 1\}$ — результат вычисления