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

ВМК МГУ, 2022/2023, осенний семестр

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)

Сигнатура 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):

$$\begin{array}{lcl} t & ::= & \mathrm{x} \mid \mathbf{c} \mid \mathbf{f}(t_1, \ldots, t_n), \\ \varphi & ::= & \mathrm{P}(t_1, \ldots, t_n) \mid (\neg\varphi) \mid (\varphi \,\&\, \varphi) \mid (\varphi \vee \varphi) \mid \\ & & (\varphi \to \varphi) \mid (\exists \mathrm{x}\varphi) \mid (\forall \mathrm{x}\varphi), \end{array}$$

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

Term — множество всех термов (заданной сигнатуры)

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

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

Напоминание: логика предикатов (термы, формулы)

Приоритеты логических операций в порядке убывания:

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

Ещё немного формул:

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

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

Императивные программы: синтаксис

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

$$
\begin{array}{lcl}
\pi & ::= & \mathit{instr} \mid \mathit{instr}\ \pi \\
\mathit{instr} & ::= & \emptyset \mid \\
 & & \mathtt{x} := t; \mid \\
 & & \textbf{if}\ C\ \textbf{then}\ \pi\ \textbf{else}\ \pi\ \textbf{fi} \mid \\
 & & \textbf{while}\ C\ \textbf{do}\ \pi\ \textbf{od}
\end{array}
$$

Здесь

- π — программа
- *instr* — инструкция (по-другому, команда)
  (сверху вниз в БНФ:
  пустая инструкция, присваивание, ветвление, цикл)
- $\mathtt{x}$ — 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 $\theta : \mathit{Var} \to \mathit{Term}$, такое что лишь для конечного числа переменных $\mathtt{x}$ верно неравенство $\theta(\mathtt{x}) \neq \mathtt{x}$

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

Подстановку можно задать конечным множеством связок:
$\theta = \{\mathtt{x_1}/t_1, \ldots, \mathtt{x_n}/t_n\}$ — это подстановка следующего вида:

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

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

Например,

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

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

$$\mathtt{x}(\theta\eta) = (\mathtt{x}\theta)\eta$$

Напоминание: логика предикатов

Интерпретации

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

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

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

$$\overline{\mathbf{2}}\,\overline{+}\,\overline{\mathbf{2}} \quad \equiv \quad 4$$

Напоминание: логика предикатов

Истинность формул

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

$$
\begin{array}{lcl}
\mathcal{I}_{ar} & \models & \mathrm{x} = \mathrm{x} \\
\mathcal{I}_{ar} & \not\models & \mathrm{x} = \mathbf{1} \\
\mathcal{I}_{ar} & \models & \forall \mathrm{x}\, \exists \mathrm{y}\, (\mathrm{y} = \mathrm{x} + \mathbf{1}) \\
\mathcal{I}_{ar} & \models & \exists \mathrm{y}\, (\mathrm{y} = \mathrm{x} + \mathbf{1}) \\
\mathcal{I}_{ar} & \not\models & \exists \mathrm{y}\, (\mathrm{x} = \mathbf{2} * \mathrm{y})
\end{array}
$$

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 $\rightarrow$ на множестве состояний вычисления для программы π в интерпретации $\mathcal{I}$ определяется так:

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

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 $\to$ на множестве состояний вычисления для программы π в интерпретации $\mathcal{I}$ определяется так:

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

Императивные программы: операционная семантика

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

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

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

($\to^*$ — *транзитивное замыкание отношения* $\to$)

Императивные программы: пример

π: **while** $\mathtt{x} > 0$ **do** $\mathtt{x} := \mathtt{x} - 1$; **od** $\qquad \theta = \{\mathtt{x}/1\}$

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

$$\langle \textbf{while } \mathtt{x} > 0 \textbf{ do } \mathtt{x} := \mathtt{x} - 1; \textbf{ od} \mid \{\mathtt{x}/1\}\rangle$$
$$\downarrow$$
$$\langle \mathtt{x} := \mathtt{x} - 1; \textbf{ while } \mathtt{x} > 0 \textbf{ do } \mathtt{x} := \mathtt{x} - 1; \textbf{ od} \mid \{\mathtt{x}/1\}\rangle$$

Пояснение:

$$(\mathtt{x} > 0)\{\mathtt{x}/1\} = (1 > 0)$$
$$\mathcal{I} \models (1 > 0)$$

Императивные программы: пример

π: **while** $\mathtt{x} > 0$ **do** $\mathtt{x} := \mathtt{x} - 1$; **od** $\qquad \theta = \{\mathtt{x}/1\}$

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

$$\langle \textbf{while } \mathtt{x} > 0 \textbf{ do } \mathtt{x} := \mathtt{x} - 1; \textbf{ od} \mid \{\mathtt{x}/1\}\rangle$$
$$\downarrow$$
$$\langle \mathtt{x} := \mathtt{x} - 1; \textbf{ while } \mathtt{x} > 0 \textbf{ do } \mathtt{x} := \mathtt{x} - 1; \textbf{ od} \mid \{\mathtt{x}/1\}\rangle$$
$$\downarrow$$
$$\langle \emptyset \textbf{ while } \mathtt{x} > 0 \textbf{ do } \mathtt{x} := \mathtt{x} - 1; \textbf{ od} \mid \{\mathtt{x}/1 - 1\}\rangle$$

Пояснение:

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

Императивные программы: пример

π: **while** $\mathtt{x} > 0$ **do** $\mathtt{x} := \mathtt{x} - 1$; **od** $\qquad\qquad \theta = \{\mathtt{x}/1\}$

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

$$\langle \textbf{while } \mathtt{x} > 0 \textbf{ do } \mathtt{x} := \mathtt{x} - 1; \textbf{ od} \mid \{\mathtt{x}/1\}\rangle$$
$$\downarrow$$
$$\langle \mathtt{x} := \mathtt{x} - 1; \textbf{ while } \mathtt{x} > 0 \textbf{ do } \mathtt{x} := \mathtt{x} - 1; \textbf{ od} \mid \{\mathtt{x}/1\}\rangle$$
$$\downarrow$$
$$\langle \emptyset \textbf{ while } \mathtt{x} > 0 \textbf{ do } \mathtt{x} := \mathtt{x} - 1; \textbf{ od} \mid \{\mathtt{x}/1 - 1\}\rangle$$
$$\downarrow$$
$$\langle \textbf{while } \mathtt{x} > 0 \textbf{ do } \mathtt{x} := \mathtt{x} - 1; \textbf{ od} \mid \{\mathtt{x}/1 - 1\}\rangle$$

Пояснение:

Императивные программы: пример

π: **while** $\mathtt{x} > 0$ **do** $\mathtt{x} := \mathtt{x} - 1$; **od** $\qquad \theta = \{\mathtt{x}/1\}$

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

$$\langle \textbf{while } \mathtt{x} > 0 \textbf{ do } \mathtt{x} := \mathtt{x} - 1; \textbf{ od} \mid \{\mathtt{x}/1\}\rangle$$
$$\downarrow$$
$$\langle \mathtt{x} := \mathtt{x} - 1; \textbf{ while } \mathtt{x} > 0 \textbf{ do } \mathtt{x} := \mathtt{x} - 1; \textbf{ od} \mid \{\mathtt{x}/1\}\rangle$$
$$\downarrow$$
$$\langle \emptyset \textbf{ while } \mathtt{x} > 0 \textbf{ do } \mathtt{x} := \mathtt{x} - 1; \textbf{ od} \mid \{\mathtt{x}/1-1\}\rangle$$
$$\downarrow$$
$$\langle \textbf{while } \mathtt{x} > 0 \textbf{ do } \mathtt{x} := \mathtt{x} - 1; \textbf{ od} \mid \{\mathtt{x}/1-1\}\rangle$$
$$\downarrow$$
$$\langle \emptyset \mid \{\mathtt{x}/1-1\}\rangle$$

Пояснение:

$$(\mathtt{x} > 0)\{\mathtt{x}/1-1\} = (1-1>0)$$
$$\mathcal{I} \not\models (1-1>0)$$

Императивные программы: пример

π: **while** $\mathrm{x} > 0$ **do** $\mathrm{x} := \mathrm{x} - 1$; **od** $\qquad \theta = \{\mathrm{x}/1\}$

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

$$\langle \textbf{while } \mathrm{x} > 0 \textbf{ do } \mathrm{x} := \mathrm{x} - 1; \textbf{ od} \mid \{\mathrm{x}/1\}\rangle$$
$$\downarrow$$
$$\langle \mathrm{x} := \mathrm{x} - 1; \textbf{ while } \mathrm{x} > 0 \textbf{ do } \mathrm{x} := \mathrm{x} - 1; \textbf{ od} \mid \{\mathrm{x}/1\}\rangle$$
$$\downarrow$$
$$\langle \emptyset \textbf{ while } \mathrm{x} > 0 \textbf{ do } \mathrm{x} := \mathrm{x} - 1; \textbf{ od} \mid \{\mathrm{x}/1 - 1\}\rangle$$
$$\downarrow$$
$$\langle \textbf{while } \mathrm{x} > 0 \textbf{ do } \mathrm{x} := \mathrm{x} - 1; \textbf{ od} \mid \{\mathrm{x}/1 - 1\}\rangle$$
$$\downarrow$$
$$\langle \emptyset \mid \{\mathrm{x}/1 - 1\}\rangle$$

Пояснение:

$\{\mathrm{x}/1 - 1\}$ — результат вычисления