Математическая логика

(mk.cs.msu.ru → Лекционные курсы → Математическая логика (группы 318, 241))

Лекция 15

Формальная верификация программ
Императивные программы
Корректность императивных программ
Логика Хоара
Автоматизация проверки правильности программ

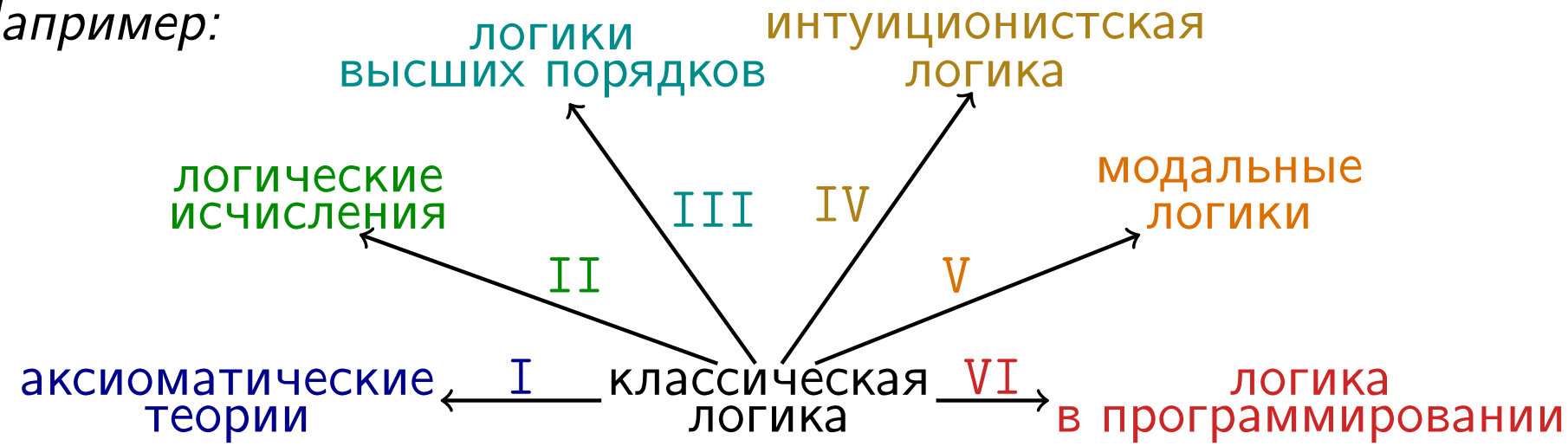
Лектор:
Подымов Владислав Васильевич

E-mail:
valdus@yandex.ru

ВМК МГУ, 2019, весенний семестр

Что изучает логика

Например:



I	Специальные интерпретации логических формул
II	Другие формы доказательного вывода
III	Расширенное применение логических операций
IV	Другая семантика логических операций
V	Другие логические операции
VI	Логические методы решения нелогических задач
...	...

Что изучает логика

Например:

I	Специальные интерпретации логических формул
II	Другие формы доказательного вывода
III	Расширенное применение логических операций
IV	Другая семантика логических операций
V	Другие логические операции
VI	Логические методы решения нелогических задач
...	...

Формальная верификация программ

“Любая нетривиальная программа
содержит хотя бы одну ошибку” (*автор неизвестен*)

Правильная программа ошибок не содержит

А что такое “правильная программа”?

Каждая программа разрабатывается для решения некоторой задачи, и программу можно назвать правильной, если эта задача действительно решается программой

А как убедиться, что написанная программа действительно решает поставленную задачу?

Формальная верификация программ

Подход к проверке правильности работы программы, про который знает каждый программист — это **тестирование**:

- придумывается показательный набор входных данных программы (тестовое покрытие)
- программа выполняется на тестовом покрытии
- результат выполнения программы сравнивается с результатом, ожидаемым согласно решаемой задаче

Основной недостаток такого подхода: **ошибки всё равно остаются** в программе, хотя их и становится меньше

Формальная верификация программ

В некоторых случаях протестировать код достаточно ("более-менее работает, а дальше пусть пользователь разбирается"), но далеко не всегда: даже от небольших программных ошибок серьёзно зависит прибыль компаний[1], успешность проектов мирового масштаба[2], быт людей[3] и даже их жизни[4]

---

[1] 1994. Процессор Intel, ошибка в реализации деления чисел с плавающей точкой $\Rightarrow$ замена дефектных процессоров, сотни миллионов \$ убытка

[2] 1962–сейчас. Ракеты и спутники, взорвавшиеся и исчезнувшие из-за программных ошибок: Фобос (пропущена кавычка), Ariane 5 (ошибка в округлении чисел), Mars Global Surveyor (перепутаны английская и метрическая системы мер), Hitomi (ошибка в программе стабилизации вращения), ...

[3] 2003. Полное отключение электричества в нескольких областях США и Канады $\Leftarrow$ ошибка в реализации взаимодействия программ оповещения об электрической нагрузке (race condition)

[4] 1980-е гг. Пять смертей при лечении рака аппаратом Therac-25 $\Leftarrow$ ошибочное увеличение мощности радиационного облучения при очень редком сочетании времён выполнения параллельных подпрограмм (race condition)

Формальная верификация программ

Есть и дугой подход к проверке правильности программ:

- формулируется набор требований к выполнению программы, означающих “программа функционирует правильно”
- **строго** доказывается или опровергается утверждение о том, что программа удовлетворяют этим требованиям

Требования такого вида, записанные на математическом языке, называются формальной спецификацей программы

Проверка соблюдения требований с помощью математических методов называется формальной верификацией программы

Если в качестве языка спецификации программ выбрать какой-либо **логический язык**, то для проверки правильности программы можно будет использовать **логические методы**

Попробуем описать такие язык и метод для **императивных программ**

Императивные программы: синтаксис

Далее считаются заданными *сигнатура* σ логики предикатов и множество *предметных переменных* Var

Синтаксис императивных программ зададим следующей БНФ:

$$
\begin{array}{lll}
\pi & ::= & stmt \mid stmt; \pi \\
stmt & ::= & \emptyset \mid \qquad \text{(пустая команда)} \\
 & & \mathtt{x} := t \mid \qquad \text{(присваивание)} \\
 & & \textbf{if } C \textbf{ then } \pi \textbf{ else } \pi \textbf{ fi} \mid \qquad \text{(ветвление)} \\
 & & \textbf{while } C \textbf{ do } \pi \textbf{ od}, \qquad \text{(цикл)}
\end{array}
$$

где π — программа,

$stmt$ — команда программы (или, по-другому, инструкция),

$\mathtt{x} \in \mathrm{Var}$,

t — *терм* и

C — условие: *бесквантор*ная формула, такая что $\mathrm{Var}_C \subseteq \mathrm{Var}$

Императивные программы: синтаксис

Пример: реализация алгоритма Эвклида вычисления наибольшего общего делителя чисел в переменных x, y в сигнатуре

$$\left\langle \{\ldots\}, \{-^{(2)}, \ldots\}, \{=^{(2)}, >^{(2)}, \ldots\} \right\rangle$$

while $\neg(\mathrm{x} = \mathrm{y})$ **do**
  if $\mathrm{x} > \mathrm{y}$ **then**
    $\mathrm{x} := \mathrm{x} - \mathrm{y}$
  else
    $\mathrm{y} := \mathrm{y} - \mathrm{x}$
  fi
od

Императивные программы: операционная семантика

Значение программы — это **вычисляемая ей функция** преобразования входных данных в выходные данные

Определение этой функции начнём с понятия состояния вычисления: “слепка” вычисления программы в заданный момент времени, которым описывается текущий результат вычисления (состояние данных) и указание на то, как этот результат будет изменяться программой дальше (состояние управления)

Состояние данных над переменными Var в интерпретации с предметной областью D — это отображение $\sigma : \mathrm{Var} \to \mathrm{D}$
Обозначение: $[\mathtt{x}_1/d_1, \ldots, \mathtt{x}_n/d_n]$, если $\mathrm{Var} = \{\mathtt{x}_1, \ldots, \mathtt{x}_n\}$

Состояние управления — это произвольная программа

Состояние вычисления — это пара $\langle \pi \mid \sigma \rangle$, где π — состояние управления и σ — состояние данных

Σ — множество всех состояний вычисления

Императивные программы: операционная семантика

$\sigma\{x \leftarrow d\}$ — состояние данных, получающееся из состояния данных σ в результате *присваивания* переменной x значения d:

$$\sigma\{x \leftarrow d\}(\mathrm{x}) = d$$
$$\sigma\{x \leftarrow d\}(\mathrm{y}) = \sigma(\mathrm{y}), \text{ если } \mathrm{y} \neq \mathrm{x}$$

Шаг вычисления программы в интерпретации $\mathcal{I}$ описывается двуместным отношением переходов $\xrightarrow{\mathcal{I}}$ на множестве Σ:

- $\langle \mathrm{x} := t \mid \sigma \rangle \xrightarrow{\mathcal{I}} \langle \emptyset \mid \sigma\{x \leftarrow \overline{t}\} \rangle$
- $\langle \textbf{if } C \textbf{ then } \pi_1 \textbf{ else } \pi_2 \textbf{ fi} \mid \sigma \rangle \xrightarrow{\mathcal{I}} \langle \pi_1 \mid \sigma \rangle$, если $\mathcal{I} \models C\sigma$
- $\langle \textbf{if } C \textbf{ then } \pi_1 \textbf{ else } \pi_2 \textbf{ fi} \mid \sigma \rangle \xrightarrow{\mathcal{I}} \langle \pi_2 \mid \sigma \rangle$, если $\mathcal{I} \not\models C\sigma$
- $\langle \textbf{while } C \textbf{ do } \pi \textbf{ od} \mid \sigma \rangle \xrightarrow{\mathcal{I}} \langle \emptyset \mid \sigma \rangle$, если $\mathcal{I} \not\models C\sigma$
- $\langle \textbf{while } C \textbf{ do } \pi \textbf{ od} \mid \sigma \rangle \xrightarrow{\mathcal{I}} \langle \pi; \textbf{while } C \textbf{ do } \pi \textbf{ od} \mid \sigma \rangle$, если $\mathcal{I} \models C\sigma$
- $\langle \pi_1; \pi_2 \mid \sigma \rangle \xrightarrow{\mathcal{I}} \langle \pi_1'; \pi_2 \mid \sigma' \rangle$, если $\langle \pi_1 \mid \sigma \rangle \xrightarrow{\mathcal{I}} \langle \pi_1' \mid \sigma' \rangle$
- $\langle \emptyset; \pi \mid \sigma \rangle \xrightarrow{\mathcal{I}} \langle \pi \mid \sigma \rangle$

Императивные программы: операционная семантика

Трасса программы π в интерпретации $\mathcal{I}$ из состояния данных σ — это последовательность состояний вычисления вида

$$\langle \pi \mid \sigma \rangle \xrightarrow{\mathcal{I}} \langle \pi_1 \mid \sigma_1 \rangle \xrightarrow{\mathcal{I}} \langle \pi_2 \mid \sigma_2 \rangle \xrightarrow{\mathcal{I}} \dots$$

Вычислениями программы называются бесконечные трассы и трассы, оканчивающиейся состоянием управления $\emptyset$

Последнее состояние данных конечной трассы называется результатом выполнения этой трассы

Иными словами, если $\xRightarrow{\mathcal{I}}$ — *транзитивное замыкание* отношения $\xrightarrow{\mathcal{I}}$, то σ' — результат вычисления программы π в интерпретации $\mathcal{I}$ из состояния данных σ, если $\langle \pi \mid \sigma \rangle \xRightarrow{\mathcal{I}} \langle \emptyset \mid \sigma' \rangle$

Программой π в интерпретации $\mathcal{I}$ вычисляется частичная функция $\mathcal{I}[\pi] : \Sigma \to \Sigma$ следующего вида:

$$\mathcal{I}[\pi](\sigma) = \sigma' \quad \Leftrightarrow \quad \langle \pi \mid \sigma \rangle \xRightarrow{\mathcal{I}} \langle \emptyset \mid \sigma' \rangle$$

Императивные программы: операционная семантика

Пример

$$\pi\text{: } \textbf{while } \mathrm{x} > 0 \textbf{ do } \mathrm{x} := \mathrm{x} - 1 \textbf{ od}; \qquad \sigma = [\mathrm{x}/1];$$

$\mathcal{I}$ — "естественная" арифметическая интерпретация

Вычисление π в $\mathcal{I}$ на σ выглядит так:

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

Пояснение:

$$\mathcal{I} \models (\mathrm{x} > 0)[\mathrm{x}/1]$$

Императивные программы: операционная семантика

Пример

$$\pi\text{: } \textbf{while } \mathrm{x} > 0 \textbf{ do } \mathrm{x} := \mathrm{x} - 1 \textbf{ od}; \qquad \sigma = [\mathrm{x}/1];$$

$\mathcal{I}$ — "естественная" арифметическая интерпретация

Вычисление π в $\mathcal{I}$ на σ выглядит так:

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

Пояснение:

$$\langle \textbf{while } \mathrm{x} > 0 \textbf{ do } \pi \textbf{ od} \mid [\mathrm{x}/1] \rangle \xrightarrow{\mathcal{I}} \langle \pi; \textbf{while } \mathrm{x} > 0 \textbf{ do } \pi \textbf{ od} \mid [\mathrm{x}/1] \rangle$$

Императивные программы: операционная семантика

Пример

$$\pi\text{: } \textbf{while } \mathrm{x} > 0 \textbf{ do } \mathrm{x} := \mathrm{x} - 1 \textbf{ od}; \qquad \sigma = [\mathrm{x}/1];$$

$\mathcal{I}$ — "естественная" арифметическая интерпретация

Вычисление π в $\mathcal{I}$ на σ выглядит так:

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

Пояснение:

$$\langle \mathrm{x} := \mathrm{x} - 1 \mid [\mathrm{x}/1] \rangle \xrightarrow{\mathcal{I}} \langle \emptyset \mid [\mathrm{x}/1]\{\mathrm{x} \leftarrow 1 - 1\} \rangle = \langle \emptyset \mid [\mathrm{x}/0] \rangle$$

Императивные программы: операционная семантика

Пример

$$\pi\colon \textbf{while } \mathrm{x} > 0 \textbf{ do } \mathrm{x} := \mathrm{x} - 1 \textbf{ od}; \qquad \sigma = [\mathrm{x}/1];$$

$\mathcal{I}$ — "естественная" арифметическая интерпретация

Вычисление π в $\mathcal{I}$ на σ выглядит так:

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

Пояснение:

$$\langle \mathrm{x} := \mathrm{x} - 1; \pi \mid [\mathrm{x}/1] \rangle \xrightarrow{\mathcal{I}} \langle \emptyset; \pi \mid [\mathrm{x}/0] \rangle$$

Императивные программы: операционная семантика

Пример

$$\pi\colon \textbf{while } \mathrm{x} > 0 \textbf{ do } \mathrm{x} := \mathrm{x} - 1 \textbf{ od}; \qquad \sigma = [\mathrm{x}/1];$$

$\mathcal{I}$ — "естественная" арифметическая интерпретация

Вычисление π в $\mathcal{I}$ на σ выглядит так:

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

Пояснение:

$$\langle \emptyset; \pi \mid \sigma \rangle \xrightarrow{\mathcal{I}} \langle \pi \mid \sigma \rangle$$

Императивные программы: операционная семантика

Пример

π: **while** $\mathrm{x} > 0$ **do** $\mathrm{x} := \mathrm{x} - 1$ **od**; $\quad\quad \sigma = [\mathrm{x}/1]$;

$\mathcal{I}$ — "естественная" арифметическая интерпретация

Вычисление π в $\mathcal{I}$ на σ выглядит так:

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

Пояснение:

$$\mathcal{I} \not\models (\mathrm{x} > 0)[\mathrm{x}/0]$$

Императивные программы: операционная семантика

Пример

$$\pi\text{: } \textbf{while } \mathrm{x} > 0 \textbf{ do } \mathrm{x} := \mathrm{x} - 1 \textbf{ od}; \qquad \sigma = [\mathrm{x}/1];$$

$\mathcal{I}$ — "естественная" арифметическая интерпретация

Вычисление π в $\mathcal{I}$ на σ выглядит так:

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

Пояснение:

$$\langle \textbf{while } \mathrm{x} > 0 \textbf{ do } \pi \textbf{ od} \mid [\mathrm{x}/0] \rangle \xrightarrow{\mathcal{I}} \langle \emptyset \mid [\mathrm{x}/0] \rangle$$

Императивные программы: операционная семантика

Пример

π: **while** $\mathrm{x} > 0$ **do** $\mathrm{x} := \mathrm{x} - 1$ **od**; $\quad\quad \sigma = [\mathrm{x}/1]$;

$\mathcal{I}$ — "естественная" арифметическая интерпретация

Вычисление π в $\mathcal{I}$ на σ выглядит так:

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

Пояснение:

$[\mathrm{x}/0]$ — результат вычисления

Задача верификации программ

Требования правильности выполнения программы могут быть записаны в виде двух отношений на состояниях данных:

- предусловие φ — отношение, разделяющее входные данные программы на *правильные* и *неправильные*
- постусловие ψ — отношение, которым описывается *правильный* вид выходных данных после выполнения программы

Принято рассматривать два вида правильности выполнения программы относительно заданных предусловия и постусловия:

- частичная корректность: результат любого **конечного** вычисления программы на правильных входных данных правилен
- полная корректность: любое вычисление программы на правильных входных данных **конечно**, и результат этого вычисления правилен

Остановимся подробнее на **частичной** корректности программ

Задача верификации программ

Тройка Хоара (по-другому — триплет Хоара) — это запись вида $\{\varphi\}\,\pi\,\{\psi\}$, где

- φ — формула *логики предикатов*, называемая предусловием
- π — *программа*
- ψ — формула *логики предикатов*, называемая постусловием

Триплет $\{\varphi\}\,\pi\,\{\psi\}$ выполним в интерпретации $\mathcal{I}$ ($\mathcal{I} \models \{\varphi\}\,\pi\,\{\psi\}$), если для любых состояний данных σ, σ' верно следующее:

если $\mathcal{I} \models \varphi\sigma$ и значение $\sigma' = \mathcal{I}[\pi](\sigma)$ определено, то $\mathcal{I} \models \psi\sigma'$

Программа π частично корректна в интерпретации $\mathcal{I}$ относительно предусловия φ и постусловия ψ, если $\mathcal{I} \models \{\varphi\}\,\pi\,\{\psi\}$

Логика Хоара

Для проверки проверки частичной корректности программ можно адаптировать *метод семантических таблиц*: определить понятие *вывода* (дерева, построенного согласно *правилам вывода*), и свести проверку частичной корректности программы к построению особенного вывода, называемого *успешным*

Правила вывода будут выглядеть так:[1]

$$\frac{\Phi}{\Psi}, \quad \frac{\Phi}{\Psi, \Omega}, \quad \frac{\Phi}{\varphi} \quad \text{или} \quad \frac{\Phi}{\varphi, \Omega, \psi},$$

где Φ, Ψ, Ω — триплеты Хоара и φ, ψ — формулы логики предикатов

Содержательное прочтение: триплет Φ выполним в $\mathcal{I} \Leftrightarrow$ в $\mathcal{I}$ выполнимы все триплеты и истинны все формулы, записанные под чертой

[1] Hoare C.A.R. An axiomatic basis for computer programming. 1969

Логика Хоара

Вот эти правила:

$$R_{\emptyset}: \frac{\{\varphi\}\,\emptyset\,\{\varphi\}}{\textbf{true}} \qquad R_{:=}: \frac{\{\varphi\,\{x/t\}\}\,x := t\,\{\varphi\}}{\textbf{true}}$$

(переменная x свободна для t в φ)

$$R_{\textbf{if}}: \frac{\{\varphi\}\,\textbf{if } C \textbf{ then } \pi_1 \textbf{ else } \pi_2 \textbf{ fi}\,\{\psi\}}{\{\varphi \,\&\, C\}\,\pi_1\,\{\psi\}, \{\varphi \,\&\, \neg C\}\,\pi_2\,\{\psi\}}$$

$$R_{\textbf{while}}: \frac{\{\varphi\}\,\textbf{while } C \textbf{ do } \pi \textbf{ od}\,\{\varphi \,\&\, \neg C\}}{\{\varphi \,\&\, C\}\,\pi\,\{\varphi\}}$$

$$R_{;}: \frac{\{\varphi\}\,\pi_1; \pi_2\,\{\psi\}}{\{\varphi\}\,\pi_1\,\{\chi\}, \{\chi\}\,\pi_2\,\{\psi\}}$$

$$R_{inf}: \frac{\{\varphi\}\,\pi\,\{\psi\}}{\varphi \to \varphi', \{\varphi'\}\,\pi\,\{\psi'\}, \psi' \to \psi}$$

Логика Хоара

Лемма(*о корректности правил вывода Хоара*). Для любой интерпретации $\mathcal{I}$ и любого правила вывода логики Хоара

$$\frac{\Phi}{\Psi}, \quad \frac{\Phi}{\Psi, \Omega}, \quad \frac{\Phi}{\varphi}, \quad \frac{\Phi}{\varphi, \Psi, \psi}$$

верно следующее:

если $\mathcal{I} \models \Psi$, $\mathcal{I} \models \Omega$ и формулы φ, ψ истинны в $\mathcal{I}$, то $\mathcal{I} \models \Phi$

Доказательство.

Подробно рассмотрим только правило $R_{:=}$: $\dfrac{\{\varphi\{\mathtt{x}/t\}\}\,\mathtt{x} := t\,\{\varphi\}}{\textbf{true}}$

Пусть σ — произвольное состояние данных, такое что $\mathcal{I} \models \varphi\{\mathtt{x}/t\}\,\sigma$

Шаг вычисления для программы $\mathtt{x} := t$ устроен так:

$$\langle \mathtt{x} := t \mid \sigma \rangle \xrightarrow{\mathcal{I}} \langle \emptyset \mid \sigma\{x \leftarrow \overline{t}\} \rangle$$

Так как $\mathcal{I} \models \varphi\{\mathtt{x}/t\}\,\sigma$, верно и $\mathcal{I} \models \varphi(\sigma\{x \leftarrow \overline{t}\})$

Следовательно, $\mathcal{I} \models \{\varphi\{\mathtt{x}/t\}\}\,\mathtt{x} := t\,\{\varphi\}$

Корректность остальных правил можете попробовать обосновать самостоятельно

Логика Хоара

Вывод триплета $\{\varphi\}\,\pi\,\{\psi\}$ — это дерево следующего вида:

- вершины размечены триплетами и формулами
- корень помечен триплетом $\{\varphi\}\,\pi\,\{\psi\}$
- дети вершины определяются так же, как и в дереве табличного вывода для логики предикатов
- все листья помечены формулами

Успешный вывод триплета $\{\varphi\}\,\pi\,\{\psi\}$ в интерпретации $\mathcal{I}$ — это конечный вывод, все листья которого помечены формулами, истинными в $\mathcal{I}$

Логика Хоара

Теорема(*о корректности логики Хоара*). Если существует успешный вывод триплета $\{\varphi\}\,\pi\,\{\psi\}$ в интерпретации $\mathcal{I}$, то программа π частично корректна в $\mathcal{I}$ относительно предусловия φ и постусловия ψ

Доказательство.

Рассмотрим произвольный успешный вывод для $\{\varphi\}\,\pi\,\{\psi\}$ в $\mathcal{I}$

Применив *лемму о корректности правил вывода* конечное число раз, получим выполнимость триплета $\{\varphi\}\,\pi\,\{\psi\}$ в $\mathcal{I}$

Значит, по *определению частичной корректности*, программа π частично корректна в $\mathcal{I}$ относительно φ и ψ ▼

Логика Хоара

Пример: алгоритм Эвклида.

Рассмотрим такую программу π:

while $\neg(\mathtt{x} = \mathtt{y})$ **do if** $\mathtt{x} > \mathtt{y}$ **then** $\mathtt{x} := \mathtt{x} - \mathtt{y}$ **else** $\mathtt{y} := \mathtt{y} - \mathtt{x}$ **fi od**

Докажем, что в результате выполнения π в "естественной" арифметической интерпретации в переменную $\mathtt{x}$ записывается **наибольший общий делитель** (НОД) положительных целых чисел, хранящихся в $\mathtt{x}$ и $\mathtt{y}$ в начале выполнения программы

Предусловие φ: числа $\mathtt{x}$ и $\mathtt{y}$ положительны, и $\mathtt{z}$ — переменная, обозначающая НОД $\mathtt{x}$ и $\mathtt{y}$ в начале выполнения программы

$$\begin{aligned} \mathtt{u}|\mathtt{v} &: && \exists \mathtt{w}\,(\mathtt{v} = \mathtt{u} \times \mathtt{w}) \\ \gcd(\mathtt{u},\mathtt{v},\mathtt{w}) &: && (\mathtt{w}|\mathtt{u})\,\&\,(\mathtt{w}|\mathtt{v})\,\&\,\forall \mathtt{r}\,((\mathtt{r}|\mathtt{u})\,\&\,(\mathtt{r}|\mathtt{v}) \to (\mathtt{r} \le \mathtt{w})) \\ \varphi &: && \mathtt{x} > 0\,\&\,\mathtt{y} > 0\,\&\,\gcd(\mathtt{x},\mathtt{y},\mathtt{z}) \end{aligned}$$

Постусловие ψ: в $\mathtt{x}$ записан требуемый НОД

$$\psi\colon \mathtt{x} = \mathtt{z}$$

Для обоснования правильности π достаточно построить успешный табличный вывод для триплета $\{\varphi\}\,\pi\,\{\psi\}$

Логика Хоара

$\{\mathtt{x} > \mathbf{0} \,\&\, \mathtt{y} > \mathbf{0} \,\&\, \gcd(\mathtt{x}, \mathtt{y}, \mathtt{z})\}$
while $\neg(\mathtt{x} = \mathtt{y})$ **do if** $\mathtt{x} > \mathtt{y}$ **then** $\mathtt{x} := \mathtt{x} - \mathtt{y}$ **else** $\mathtt{y} := \mathtt{y} - \mathtt{x}$ **fi od**
$\{\mathtt{x} = \mathtt{z}\}$

χ_1: $\mathtt{x} > \mathbf{0} \,\&\, \mathtt{x} > \mathbf{0} \,\&\, \gcd(\mathtt{x}, \mathtt{y}, \mathtt{z}) \to \mathtt{x} > \mathbf{0} \,\&\, \mathtt{y} > \mathbf{0} \,\&\, \gcd(\mathtt{x}, \mathtt{y}, \mathtt{z})$

χ_2: $\mathtt{x} > \mathbf{0} \,\&\, \mathtt{y} > \mathbf{0} \,\&\, \gcd(\mathtt{x}, \mathtt{y}, \mathtt{z}) \,\&\, \neg\neg(\mathtt{x} = \mathtt{y}) \to \mathtt{x} = \mathtt{z}$

$\{\mathtt{x} > \mathbf{0} \,\&\, \mathtt{y} > \mathbf{0} \,\&\, \gcd(\mathtt{x}, \mathtt{y}, \mathtt{z})\}$
while $\neg(\mathtt{x} = \mathtt{y})$ **do if** $\mathtt{x} > \mathtt{y}$ **then** $\mathtt{x} := \mathtt{x} - \mathtt{y}$ **else** $\mathtt{y} := \mathtt{y} - \mathtt{x}$ **fi od**
$\{\mathtt{x} > \mathbf{0} \,\&\, \mathtt{y} > \mathbf{0} \,\&\, \gcd(\mathtt{x}, \mathtt{y}, \mathtt{z}) \,\&\, \neg\neg(\mathtt{x} = \mathtt{y})\}$

R_{inf}: $\dfrac{\{\varphi\}\,\pi\,\{\psi\}}{\varphi \to \varphi', \{\varphi'\}\,\pi\,\{\psi'\}, \psi' \to \psi}$

χ_1 истинна в $\mathcal{I}$
χ_2 истинна в $\mathcal{I}$

Логика Хоара

$\{\mathtt{x} > \mathbf{0} \,\&\, \mathtt{y} > \mathbf{0} \,\&\, \gcd(\mathtt{x}, \mathtt{y}, \mathtt{z})\}$
while $\neg(\mathtt{x} = \mathtt{y})$ **do if** $\mathtt{x} > \mathtt{y}$ **then** $\mathtt{x} := \mathtt{x} - \mathtt{y}$ **else** $\mathtt{y} := \mathtt{y} - \mathtt{x}$ **fi od**
$\{\mathtt{x} = \mathtt{z}\}$

χ_1: $\mathtt{x} > \mathbf{0} \,\&\, \mathtt{x} > \mathbf{0} \,\&\, \gcd(\mathtt{x}, \mathtt{y}, \mathtt{z}) \to \mathtt{x} > \mathbf{0} \,\&\, \mathtt{y} > \mathbf{0} \,\&\, \gcd(\mathtt{x}, \mathtt{y}, \mathtt{z})$

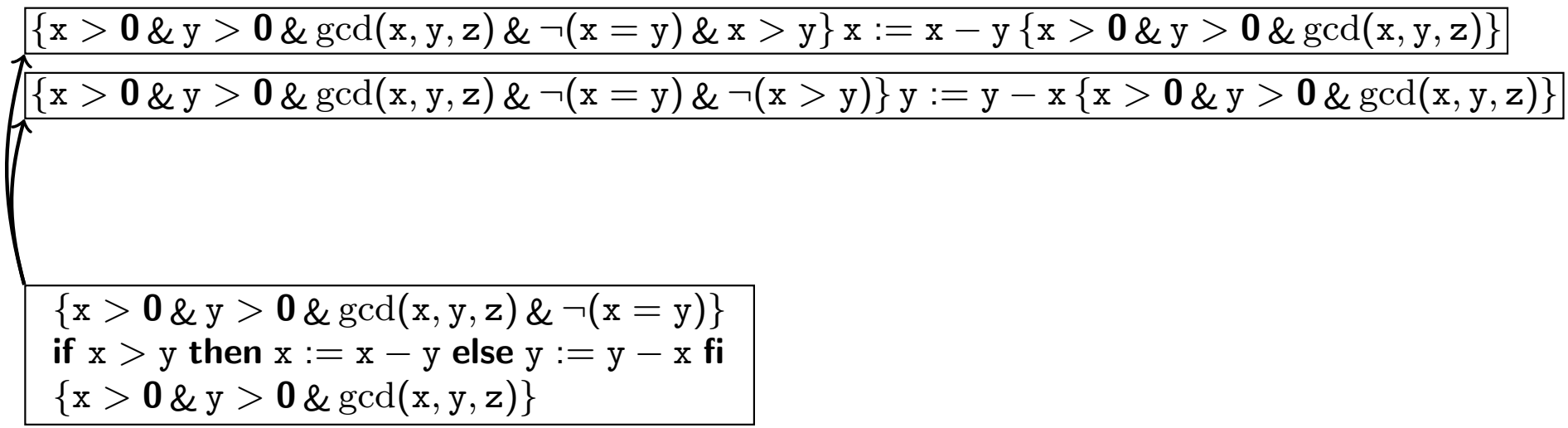
χ_2: $\mathtt{x} > \mathbf{0} \,\&\, \mathtt{y} > \mathbf{0} \,\&\, \gcd(\mathtt{x}, \mathtt{y}, \mathtt{z}) \,\&\, \neg\neg(\mathtt{x} = \mathtt{y}) \to \mathtt{x} = \mathtt{z}$

$\{\mathtt{x} > \mathbf{0} \,\&\, \mathtt{y} > \mathbf{0} \,\&\, \gcd(\mathtt{x}, \mathtt{y}, \mathtt{z})\}$
while $\neg(\mathtt{x} = \mathtt{y})$ **do if** $\mathtt{x} > \mathtt{y}$ **then** $\mathtt{x} := \mathtt{x} - \mathtt{y}$ **else** $\mathtt{y} := \mathtt{y} - \mathtt{x}$ **fi od**
$\{\mathtt{x} > \mathbf{0} \,\&\, \mathtt{y} > \mathbf{0} \,\&\, \gcd(\mathtt{x}, \mathtt{y}, \mathtt{z}) \,\&\, \neg\neg(\mathtt{x} = \mathtt{y})\}$

$\{\mathtt{x} > \mathbf{0} \,\&\, \mathtt{y} > \mathbf{0} \,\&\, \gcd(\mathtt{x}, \mathtt{y}, \mathtt{z}) \,\&\, \neg(\mathtt{x} = \mathtt{y})\}$
if $\mathtt{x} > \mathtt{y}$ **then** $\mathtt{x} := \mathtt{x} - \mathtt{y}$ **else** $\mathtt{y} := \mathtt{y} - \mathtt{x}$ **fi**
$\{\mathtt{x} > \mathbf{0} \,\&\, \mathtt{y} > \mathbf{0} \,\&\, \gcd(\mathtt{x}, \mathtt{y}, \mathtt{z})\}$

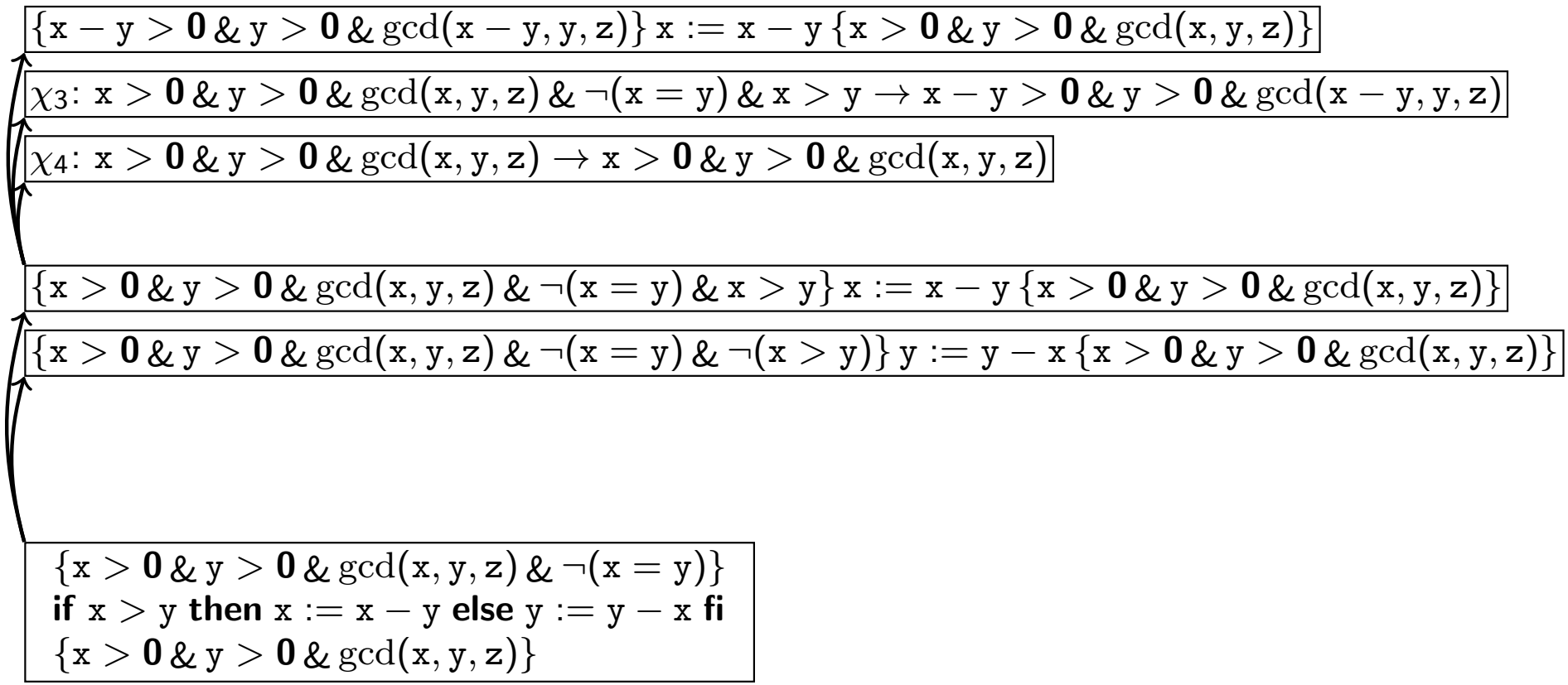
$$R_{\textbf{while}}\colon \frac{\{\varphi\}\ \textbf{while}\ C\ \textbf{do}\ \pi\ \textbf{od}\ \{\varphi \,\&\, \neg C\}}{\{\varphi \,\&\, C\}\ \pi\ \{\varphi\}}$$

Логика Хоара



$$R_{\textbf{if}}\colon\ \frac{\{\varphi\}\ \textbf{if}\ C\ \textbf{then}\ \pi_1\ \textbf{else}\ \pi_2\ \textbf{fi}\ \{\psi\}}{\{\varphi \,\&\, C\}\,\pi_1\,\{\psi\}\,,\{\varphi \,\&\, \neg C\}\,\pi_2\,\{\psi\}}$$

Логика Хоара

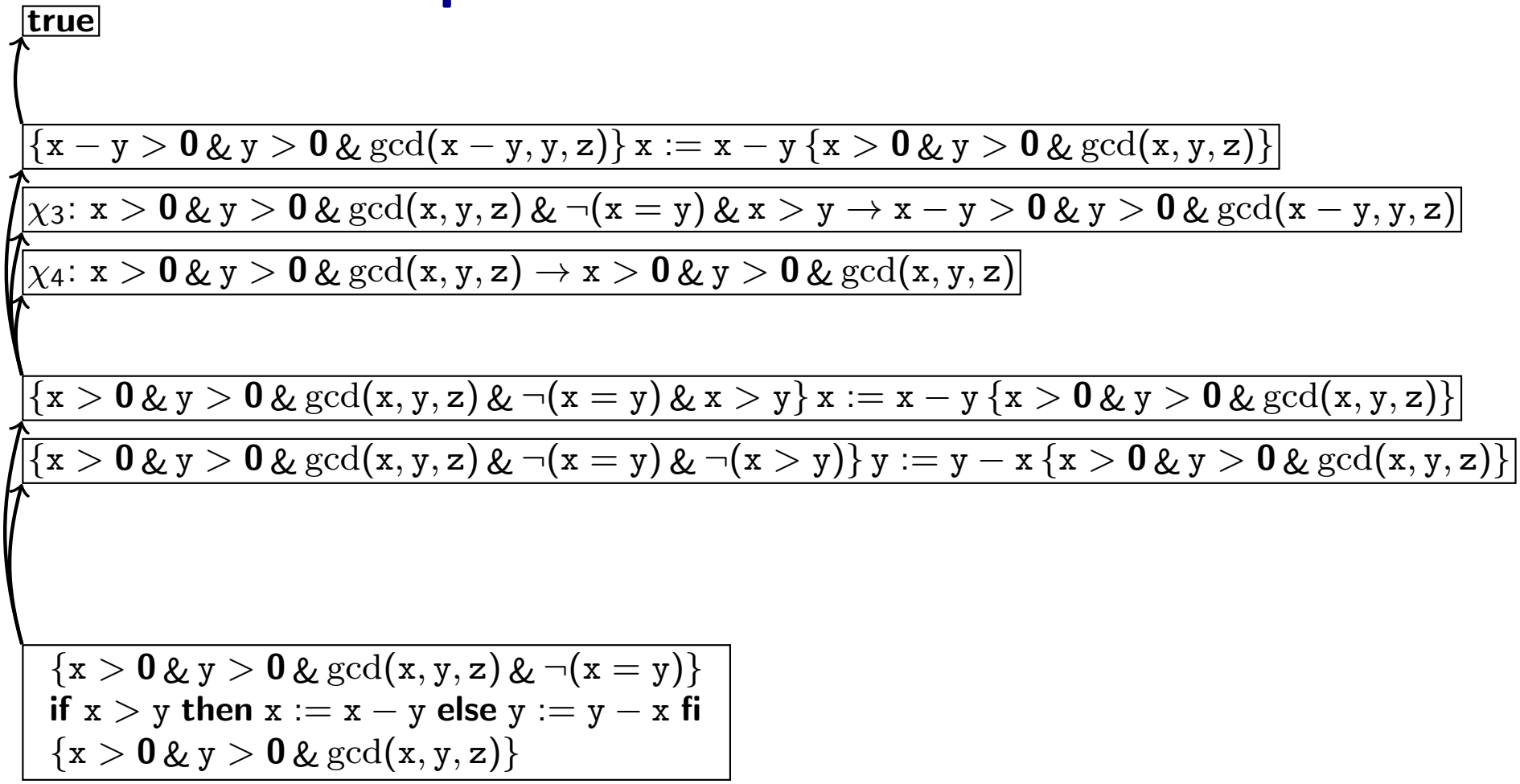


R_{inf}: $\dfrac{\{\varphi\}\,\pi\,\{\psi\}}{\varphi \to \varphi',\ \{\varphi'\}\,\pi\,\{\psi'\},\ \psi' \to \psi}$

χ_3 истинна в $\mathcal{I}$

χ_4 истинна в $\mathcal{I}$

Логика Хоара



$$R_{:=}\colon \frac{\{\varphi\{x/t\}\}\, x := t\, \{\varphi\}}{\textbf{true}}$$

Логика Хоара

$\boxed{\mathbf{true}}$

$\boxed{\{\mathtt{x}-\mathtt{y}>\mathbf{0}\,\&\,\mathtt{y}>\mathbf{0}\,\&\,\gcd(\mathtt{x}-\mathtt{y},\mathtt{y},\mathtt{z})\}\,\mathtt{x}:=\mathtt{x}-\mathtt{y}\,\{\mathtt{x}>\mathbf{0}\,\&\,\mathtt{y}>\mathbf{0}\,\&\,\gcd(\mathtt{x},\mathtt{y},\mathtt{z})\}}$

$\boxed{\chi_3\colon\ \mathtt{x}>\mathbf{0}\,\&\,\mathtt{y}>\mathbf{0}\,\&\,\gcd(\mathtt{x},\mathtt{y},\mathtt{z})\,\&\,\neg(\mathtt{x}=\mathtt{y})\,\&\,\mathtt{x}>\mathtt{y}\to\mathtt{x}-\mathtt{y}>\mathbf{0}\,\&\,\mathtt{y}>\mathbf{0}\,\&\,\gcd(\mathtt{x}-\mathtt{y},\mathtt{y},\mathtt{z})}$

$\boxed{\chi_4\colon\ \mathtt{x}>\mathbf{0}\,\&\,\mathtt{y}>\mathbf{0}\,\&\,\gcd(\mathtt{x},\mathtt{y},\mathtt{z})\to\mathtt{x}>\mathbf{0}\,\&\,\mathtt{y}>\mathbf{0}\,\&\,\gcd(\mathtt{x},\mathtt{y},\mathtt{z})}$

$\boxed{\{\mathtt{x}>\mathbf{0}\,\&\,\mathtt{y}>\mathbf{0}\,\&\,\gcd(\mathtt{x},\mathtt{y},\mathtt{z})\,\&\,\neg(\mathtt{x}=\mathtt{y})\,\&\,\mathtt{x}>\mathtt{y}\}\,\mathtt{x}:=\mathtt{x}-\mathtt{y}\,\{\mathtt{x}>\mathbf{0}\,\&\,\mathtt{y}>\mathbf{0}\,\&\,\gcd(\mathtt{x},\mathtt{y},\mathtt{z})\}}$

$\boxed{\{\mathtt{x}>\mathbf{0}\,\&\,\mathtt{y}>\mathbf{0}\,\&\,\gcd(\mathtt{x},\mathtt{y},\mathtt{z})\,\&\,\neg(\mathtt{x}=\mathtt{y})\,\&\,\neg(\mathtt{x}>\mathtt{y})\}\,\mathtt{y}:=\mathtt{y}-\mathtt{x}\,\{\mathtt{x}>\mathbf{0}\,\&\,\mathtt{y}>\mathbf{0}\,\&\,\gcd(\mathtt{x},\mathtt{y},\mathtt{z})\}}$

$\boxed{\chi_5\colon\ \begin{array}{r}\mathtt{x}>\mathbf{0}\,\&\,\mathtt{y}>\mathbf{0}\,\&\,\gcd(\mathtt{x},\mathtt{y},\mathtt{z})\,\&\,\neg(\mathtt{x}=\mathtt{y})\,\&\,\neg(\mathtt{x}>\mathtt{y})\to\\ \mathtt{x}>\mathbf{0}\,\&\,\mathtt{y}-\mathtt{x}>\mathbf{0}\,\&\,\gcd(\mathtt{x},\mathtt{y}-\mathtt{x},\mathtt{z})\end{array}}$

$\boxed{\chi_6\colon\ \mathtt{x}>\mathbf{0}\,\&\,\mathtt{y}>\mathbf{0}\,\&\,\gcd(\mathtt{x},\mathtt{y},\mathtt{z})\to\mathtt{x}>\mathbf{0}\,\&\,\mathtt{y}>\mathbf{0}\,\&\,\gcd(\mathtt{x},\mathtt{y},\mathtt{z})}$

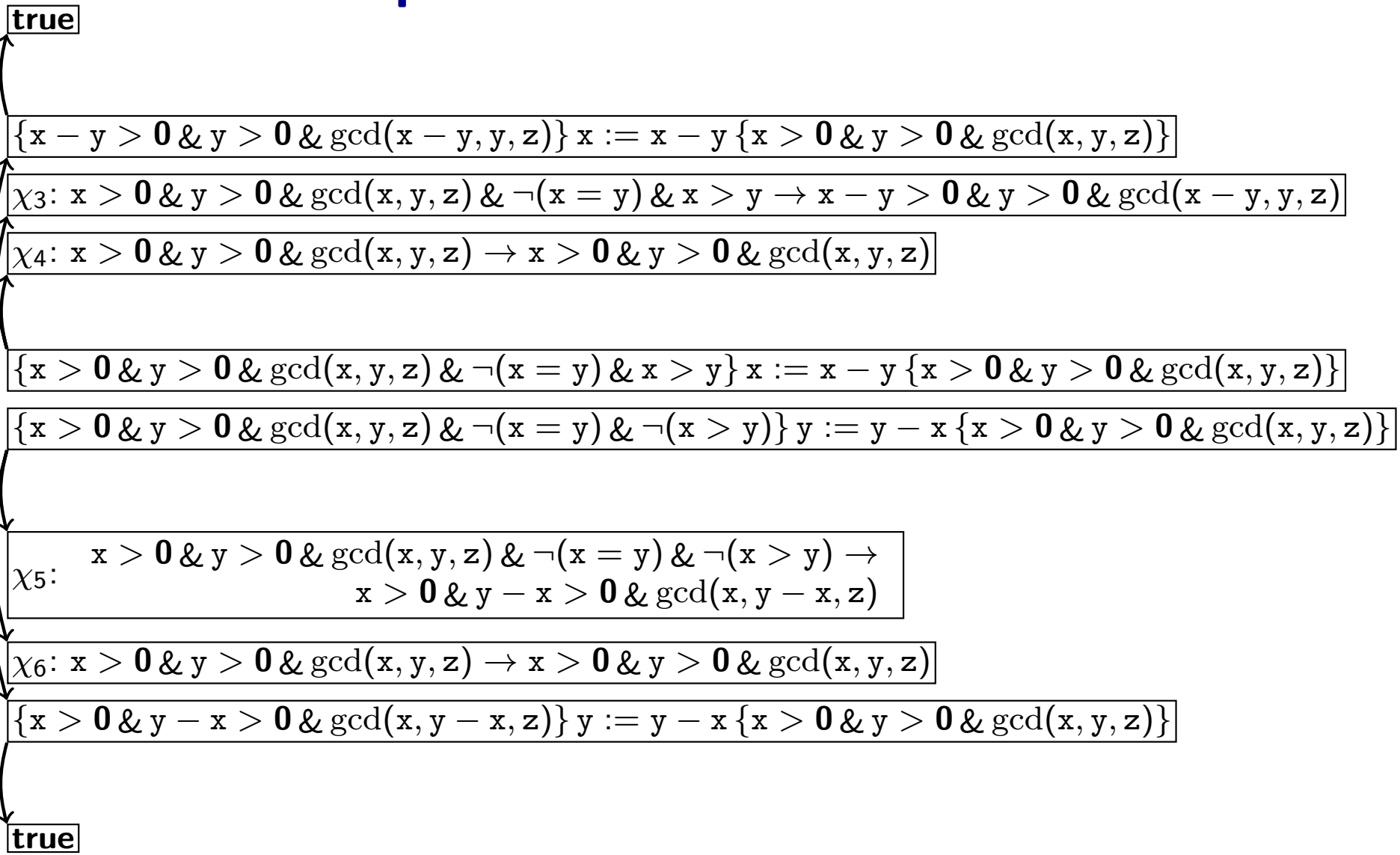
$\boxed{\{\mathtt{x}>\mathbf{0}\,\&\,\mathtt{y}-\mathtt{x}>\mathbf{0}\,\&\,\gcd(\mathtt{x},\mathtt{y}-\mathtt{x},\mathtt{z})\}\,\mathtt{y}:=\mathtt{y}-\mathtt{x}\,\{\mathtt{x}>\mathbf{0}\,\&\,\mathtt{y}>\mathbf{0}\,\&\,\gcd(\mathtt{x},\mathtt{y},\mathtt{z})\}}$

$R_{inf}\colon\ \dfrac{\{\varphi\}\,\pi\,\{\psi\}}{\varphi\to\varphi',\{\varphi'\}\,\pi\,\{\psi'\},\psi'\to\psi}$

χ_5 истинна в $\mathcal{I}$

χ_6 истинна в $\mathcal{I}$

Логика Хоара



$$R_{:=}\colon \frac{\{\varphi\{x/t\}\}\, x := t\, \{\varphi\}}{\text{true}}$$

Полнота логики Хоара

Согласно *теореме о корректности логики Хоара*, правильность программы можно обосновать, построив успешный табличный вывод подходящего триплета Хоара

А правда ли, что корректность **любой** правильной программы может быть обоснована построением успешного табличного вывода подходящего триплета?

На самом деле это не один вопрос, а два принципиально разных:

1. Все ли свойства правильности программ могут быть записаны в виде формул логики предикатов?
2. Для любого ли триплета существует успешный вывод?

Полнота логики Хоара

Все ли свойства правильности программ могут быть записаны в виде формул логики предикатов?

Ответ на этот вопрос — в общем случае **нет**: в виде формул можно записать только отношения, *выразимые* в интерпретации

Сигнатуру интерпретации можно расширить, но чем шире класс выразимых отношений, тем труднее анализировать формулы соответствующей сигнатуры — например:

- формулами сигнатуры *формальной арифметики* можно выразить все графики вычислимых функций, но анализировать произвольные формулы этой сигнатуры практически невозможно
- формулы сигнатуры *арифметики Пресбургера* поддаются анализу, но выразить сколько-нибудь полезные свойства программ в этой арифметике практически невозможно

Полнота логики Хоара

Для любого ли триплета существует успешный вывод?

Ответ на этот вопрос — **иногда да, иногда нет**:

- **да**, если при применении правила

$$R_{inf} : \frac{\{\varphi\}\,\pi\,\{\psi\}}{\varphi \to \varphi', \{\varphi'\}\,\pi\,\{\psi'\}, \psi' \to \psi}$$

  всегда есть возможность записать в виде формул φ', ψ' свойства данных, "подходящие" для доказательства
  - строгая формулировка и, тем более, обоснование этого факта в курсе не приводятся
- **нет**, если язык формул недостаточно выразителен для записи таких свойств
  - например, используя циклы, можно реализовать **умножение** в программе сигнатуры $\langle\{0\}, \{+, \mathbf{s}\}, \{=\}\rangle$, работающей с целыми неотрицательными числами: в доказательстве корректности такой программы потребуется *выразить* умножение формулой, чего сделать нельзя

Автоматизация проверки правильности программ

А можно ли реализовать программу, автоматически доказывающую корректность программ?

Как и в вопросе про полноту, это на самом деле не один вопрос, а два:

1. Можно ли автоматизировать построение триплетов Хоара, описывающих свойства правильности программ?
2. Можно ли автоматизировать построение успешного вывода для заданных триплетов Хоара?

Ответ на первый вопрос однозначен — **нет**: один из главных недостатков формальной верификации состоит в том, что формальная спецификация программы, как правило, создаётся вручную специально обученным экспертом

Автоматизация проверки правильности программ

Можно ли автоматизировать построение успешного вывода для заданных триплетов Хоара?

С этим вопросом дела обстоят чуть лучше

Слабейшим предусловием для программы π и постусловия ψ в интерпретации $\mathcal{I}$ называется формула $wpr(\pi, \psi, \mathcal{I})$, такая что

- $\mathcal{I} \models \{wpr(\pi, \psi, \mathcal{I})\}\,\pi\,\{\psi\}$ и
- для любой формулы φ, такой что $\mathcal{I} \models \{\varphi\}\,\pi\,\{\psi\}$, верно соотношение $\mathcal{I} \models \varphi \to wpr(\pi, \psi, \mathcal{I})$

Замечание: слабейших предусловий на самом деле бывает много, но все они *равносильны* в интерпретации, так что для простоты иногда будем считать, что оно единственно

Автоматизация проверки правильности программ

Теорема. $\mathcal{I} \models \{\varphi\}\,\pi\,\{\psi\} \Leftrightarrow$
$\mathcal{I} \models \{\mathit{wpr}(\pi, \psi, \mathcal{I})\}\,\pi\,\{\psi\}$ **и** $\mathcal{I} \models \varphi \to \mathit{wpr}(\pi, \psi, \mathcal{I})$

Доказательство. Следует из определения слабейшего предусловия ▼

Таким образом, чтобы автоматизировать проверку выполнимости триплетов Хоара, достаточно научиться автоматически

- проверять истинность формул в интерпретации
  - (*это одна из "тяжёлых" проблем, возникающих при автоматизации проверки правильности программ*) и
- вычислять слабейшее предусловие для программы, интерпретации и постусловия

Автоматизация проверки правильности программ

Теорема(*о слабейшем предусловии*).

- $wpr(\emptyset, \psi, \mathcal{I}) = \psi$
- $wpr(\mathtt{x} := t, \psi, \mathcal{I}) = \psi\{\mathtt{x}/t\}$,
  если переменная $\mathtt{x}$ **свободна для** t **в** ψ
- $wpr(\textbf{if } C \textbf{ then } \pi_1 \textbf{ else } \pi_2 \textbf{ fi}, \psi, \mathcal{I}) =$
  $C \,\&\, wpr(\pi_1, \psi, \mathcal{I}) \vee \neg C \,\&\, wpr(\pi_2, \psi, \mathcal{I})$
- $wpr(\pi_1; \pi_2, \psi, \mathcal{I}) = wpr(\pi_1, wpr(\pi_2, \psi, \mathcal{I}), \mathcal{I})$

Доказательство. Попробуйте самостоятельно

Таким образом, для всех программ, устройство которых подходит под условие теоремы, слабейшее предусловие вычисляется автоматически и довольно просто и не зависит от выбора интерпретации

Автоматизация проверки правильности программ

А как вычислить слабейшее предусловие для цикла?

Вид слабейшего предусловия тесно связан с устройством правил доказательства корректности программ: вычисление слабейшего предусловия — настолько же (не)простая задача, насколько и применение правила для построения успешного вывода

Чтобы применить правило

$$R_{\textbf{while}} : \frac{\{\varphi\}\,\textbf{while}\ C\ \textbf{do}\ \pi\ \textbf{od}\,\{\varphi\,\&\,\neg C\}}{\{\varphi\,\&\,C\}\,\pi\,\{\varphi\}},$$

требуется предварительно найти формулу φ, в которой записано свойство состояний данных, сохраняющееся при выполнении каждого витка цикла

Эта формула φ называется инвариантом цикла и явно или неявно используется практически во всех попытках анализа циклов

Автоматическая генерация инвариантов циклов — это **ключевая проблема** автоматизации проверки правильности программ