Формальные модели программ

Фундаментальные модели вычислений: конечные автоматы → машина Тьюринга; частично-рекурсивные функции; нормальные алгорифмы Маркова; λ-исчисление. Менее известные модели вычислений. Понятие эквивалентности фундаментальных моделей вычислений.

maxim.krivchikov@gmail.com

Материалы курса: https://maxxk.github.io/formal-models-2015/

Рубрика «По следам наших публикаций»

(дополнения и отступления по итогам предыдущих занятий и по заданным вопросам)

Free / Libre / Open Source Software

В терминологии Free Software Foundation — четыре «свободы» (права):

0. Свобода запускать программу как захочется пользователю, в любых целях.
1. Свобода изучать то, как работает программа, и изменять её в своих целях. Для реализации этой свободы подразумевается свободный доступ к исходному коду.
2. Свобода распространения копий программы, чтобы можно было помогать ближним.
3. Свобода распространения копий модифицированных версий программы другим. Таким образом всё общество получит пользу от ваших изменений. Для реализации этой свободы подразумевается свободный доступ к исходному коду.

FSF активно защищает эти четыре свободы, используя для этого т.н. «copyleft»-лицензии (например, GPL). Такие «вирусные» лицензии требуют, чтобы если вы используете фрагмент GPL-кода (или GPL-библиотеку) в вашей программе, вся программа должна распространяться под не противоречащими этой лицензии условиями.

Open Source Software

Для многих в индустрии «copyleft»-лицензии не подходят. Есть список OpenSource-лицензий, одобренных OSI: http://opensource.org/licenses/alphabetical

Определение OpenSource OSI — 10 критериев: (1) свободное распространение; (2) доступ к исходному коду и право на компиляцию; (3) право на модификацию и производные работы; (4) возможны незначительные ограничения на изменения (распространять модификацию только в виде патчей, изменять имя (Firefox/Iceweasel) или номер версии); (5) запрет на дискриминацию людей или групп лиц; (6) запрет на дискриминацию по области использования; (7) запрет на требование получения дополнительных разрешений для вступления OpenSource-лицензии в действие; (8) запрет на лицензии «только в составе дистрибутива»; (9) запрет на ограничение других совместно распространяемых программ; (10) технологическая нейтральность.

Примеры copyleft-лицензий: GPL, MPL
Примеры «хороших» не-copyleft-лицензий: MIT, Apache 2.0
Примеры «плохих» (не-OpenSource) лицензий: Ms-RSL («можно только смотреть»), Ms-LPL, Ms-LRL (только для разработки под Windows), «public domain» (всё сложно)

Open Source в нашей жизни

- 80-90% рынка движков web-браузеров:
  - WebKit (Safari, все iOS-браузеры)
  - Blink (Chrome — почти полностью OpenSource (Chromium), Opera, Yandex Browser)
  - Gecko (Firefox — OpenSource)
- 70% из 1 млн «наиболее загруженных сайтов», 55% по всем сайтам
- Linux, GNU, Android
- XNU, Darwin (низкоуровневая часть Mac OS X и iOS)
- VLC, FFmpeg, TeX, Zotero
- значительная часть интерпретаторов и компиляторов языков программирования

OpenSource и бизнес

- поддержка (как в обычном смысле, так и «допиливание» под нужды заказчика)
- двойное лицензирование (продажа исключений из copyleft-лицензии)
- «Community» и «Enterprise» версии
- SaaS

Для SaaS активно используется следующий принцип (на мой взгляд, наиболее правильный): открывать инфраструктурные компоненты и держать закрытым код основного продукта. Пример — GitHub.

Только на GitHub: Facebook и Instagram, Microsoft, Intel, Yandex, Google, Bloomberg, Twitter.

Yandex недавно писал о переходе с СУБД Oracle на PostgreSQL.

Верификация программ

- рецензирование, автоматизированное тестирование и статический анализ могут и должны использоваться в комплексе
- формальная верификация на практике применяется при разработке микросхем — в этой области ошибки дороги и неисправимы

Фундаментальные модели вычислений

Литература: Васенин В., Кривчиков М. Формальные модели программ и языков программирования. Часть 1. Библиографический обзор 1930—1989 гг // Программная инженерия. — 2015. — № 5. — С. 10–19. (и ссылки в ней)

Автомат

— «это пятёрка» (А.Б. Угольников) $A = (Q, I, O, \delta, \lambda)$

$\mathbf{Q}$
: множество состояний

$\mathbf{I, O}$
: входной и выходной алфавит

$\boldsymbol{\delta} : \mathbf{I} \times \mathbf{Q} \to \mathbf{Q}$
: функция состояний

$\boldsymbol{\lambda} : \mathbf{I} \times \mathbf{Q} \to \mathbf{O}$
: функция выходов

Автомат хранит текущее состояние q. На каждом шаге на вход автомата подаётся входной символ $i \in I$ и с помошью функций δ, λ определяется выходное значение и состояние для следующего шага.

Детерминированный конечный автомат

— ограничиваемся рассмотрением конечных множеств состояний Q

Инициальный автомат

— выделяем исходное состояние q_0

Распознающий автомат

— выделяем подмножество состояний $F \subset Q$, подаём на вход последовательность символов α; если после подачи последнего символа автомат находится в состоянии F, то говорим, что автомат распознал слово (последовательность) α

Недетерминированный автомат

— может находиться сразу в множестве состояний $Q' \subseteq Q$

Кризис оснований математики

http://personal.us.es/josef/pcmCrisis.pdf
https://en.wikipedia.org/wiki/Foundations_of_mathematics#Foundational_crisis

В начале XX века был представлен ряд парадоксов наивной теории множеств, связанный как раз с широко обсуждаемыми с конца XIX века вещами — бесконечностью, предикативностью и т.п.

Парадокс Б. Рассела («парадокс брадобрея»): $R = \{x | x \in x\} \Longrightarrow (R \in R \Leftrightarrow R \notin R)$ (ошибка — возможность определения «множества всех множеств»).

Парадокс Бурали-Форти: пусть x — ординал, а $\cup x$ — верхняя грань x. Для Q — множества всех ординалов, тогда $\cup Q + 1$ — ординал, который не лежит в Q (ошибка — возможность определения множества элементов, обладающих произвольным свойством).

(ординал — транзитивное множество, которое является полным порядком по отношению включения)

$\Rightarrow$ математика в опасности

Формализм

— давайте все математические утверждения записывать в виде формул (строк символов), которые переписываются по конечному набору правил, про которые известно, что они не дадут парадоксов.

План

1. Сформулировать строгие синтаксические аксиомы и правила вывода (переписывания).
2. Доказать 3 простых свойства, которые показывают, что аксиомы «хорошие» (программа Гильберта):
   1. Полнота (мы можем переписыванием получить из аксиом все истинные утверждения математики)
   2. Непротиворечивость (мы не можем получить переписыванием из аксиом ложное утверждение)
   3. Разрешимость (руководствуясь простым набором правил можно для любой формулы получить, выводима она или нет).
3. Представить основные теоремы анализа в терминах этой системы

Entscheidungsproblem

(decision problem, проблема разрешения)

Найти алгоритм, который может для любой формулы исчисления предикатов (возможно, расширенного конечным набором аксиом) дать ответ, является ли эта формула общезначимой («Да» или «Нет»).

Задача 2.0 ***** ↑ ;-)

Что такое алгоритм?

Машина Тьюринга (1936)

— бесконечная лента с записанными на ней символами (конечного) алфавита Σ и «пустыми ячейками» ε. Над лентой установлен (детерминированный) конечный автомат, множество входа и выхода которого — Σ. На каждом шаге автомат считывает значение ячейки, над которой он установлен, записывает выходное значение и сдвигается влево, вправо или останавливается. Если автомат остановился или сдвинулся на пустую ячейку, то вычисление закончено, следующий шаг не выполняется.

В начальный момент времени на ленте непусто конечное число ячеек.

Тезис Чёрча–Тьюринга: «Если функция может быть вычислена физическим устройством, то существует вычисляющая её машина Тьюринга».

Пространства сложности

Пусть дана некоторая задача, размер которой оценивается натуральным числом n, и известно решение этой задачи в виде машины Тьюринга. За сколько шагов (за какое время), в зависимости от n, остановится (даст ответ) машина Тьюринга?

P — время выражается как полином
NP — время выражается как полином для *недетерминированной* машины Тьюринга (неформальное определение — «ответ можно проверить за полиномиальное время»)
NPC — любая задача из **NP** за полиномиальное время может быть сведена к такой задаче

Универсальная машина Тьюринга

Мы можем закодировать автомат, входные и выходные данные в виде нулей и единиц и расположить на ленте в виде:

| автомат # входные данные *

и написать автомат, который, используя ленту для хранения состояния может эмулировать работу произвольной машины Тьюринга.

Проблема останова

Не все машины Тьюринга останавливаются для любого набора входных данных. Пример: машина с одним состоянием; на входе 0 — записать 0 и сдвинуться вправо, на входе 1 — записать 1 и сдвинуться влево.

Проблема останова: написать машину Тьюринга, которая по данной кодировке универсальной машины Тьюринга для данной машины и входных данных определяет (за конечное время), остановится ли машина для входных данных («Да» или «Нет», должна работать на всех входных данных).

Разрешимость Entscheidungsproblem ⇒ решение проблемы останова.

Проблема останова неразрешима (1936)

План доказательства:

Пусть дана (полная вычислимая) функция $h(i, x)$, которая решает проблему остаова для машины i и входа x.
Для **любой** (полной вычислимой) функции $f(a, b)$ можно построить *частичную* (вычислимую) функцию $g(i)$, такую, что она равна 0, если $f(i, i) = 0$ и не определена в противном случае (т.е. её машина-вычислитель не останавливается для такого входа).

Пусть e — это код машины-вычислителя g. Тогда ни для какой f значение $h(e, e)$ не будет совпадать со значением $f(e, e)$:

1. $f(e, e) = 0 \Longrightarrow g(e) = 0 \Longrightarrow h(e, e) = 1$
2. $f(e, e) \neq 0 \Longrightarrow g(e) \perp \Longrightarrow h(e, e) = 0$

Формальные языки

— подмножества строк конечного алфавита Σ.
Иерархия Хомского (Noam Chomsky, 1956):

- регулярные языки: $\emptyset, a \in \Sigma, A \cup B, A \cap B, A^*$ (любой конечный язык и некоторые бесконечные; распознаются детерминированным конечным автоматом)
- контекстно-свободные языки (произвольное количество парных скобок $(^n)^n$ — контекстно-свободный язык, распознаются автоматом с магазинной памятью (стеком))
- контекстно-зависимые языки (произвольное количество троек $(^n|^n)^n$ — контекстно-зависимый язык, распознаются линейно-ограниченным автоматом — машиной Тьюринга с лентой, длина которой линейно зависит от длины входа)
- неограниченные языки (распознаются машиной Тьюринга)

Леммы о накачке (разрастании)

pumping lemma

— используются когда нужно опровергнуть принадлежность языка к некоторому классу.

Для регулярных языков: для любого регулярного языка есть такое число α, такое, что, если взять слово ω длины $> \alpha$, то можно так представить его в виде $\omega = xyz$ ($|xy| \leqslant \alpha$, $|y| \geqslant 1$), причём для любого натурального n слово $xy^n z$ тоже входит в язык.

Для контекстно-свободных: аналогичная формулировка, но: $\omega = uvwxy$, $|vwx| < \alpha$, $|vx| \geqslant 1$ и языку принадлежит $uv^n wx^n y$.

(Нетипизированное) λ-исчисление Чёрча (1932)

— изначально было предложено в качестве формальной системы, аналогичной требуемой в программе Гильберта.

Рассмотрим формулы, составленные из выражений вида:

1. $\lambda x.\, F$ — абстракция (x — имя переменной, которая может встречаться в формуле F)
2. $x, y, z, \ldots$ — переменная
3. $A \cdot B$ — приложение (аппликация), A, B — формулы.

Введём правило переписывания (β-редукция):

$$(\lambda x.\, F) \cdot G \longrightarrow_{\beta} F[G/x]$$

Буква α зарезервирована для понятия α-эквивалентности — формулы, эквивалентные с точностью до переименования переменных.

Вычисление считается завершённым когда нет β-редексов (подформул, к которым применима β-редукция)

Индексы де Брёйна (de Bruijn, де Браун)

Чтобы избавиться от α-эквивалентности, вместо имён используем цифры — глубину терма

$$\lambda\ (\lambda\ 1\ (\lambda\ 1))\ (\lambda\ 2\ 1)$$

Примеры

1. $\Omega \equiv (\lambda x.\, x \cdot x) \cdot (\lambda x.\, x \cdot x)$ — редукция не останавливается

2. Нумералы Чёрча — представляем натуральное число n как n-кратное применение функции:
   $0 \equiv \lambda f.\, \lambda x.\, x$
   $1 \equiv \lambda f.\, \lambda x. f \cdot x$
   $2 \equiv \lambda f.\, \lambda x. f \cdot (f \cdot x)$
   $\ldots$

μ-рекурсивные функции Клини

Нормальные алгоритмы Маркова

Понятие об эквивалентности фундаментальных моделей вычислений

— все четыре перечисленные (и более экзотические, которые рассмотрим в следующий раз) модели вычислений (кроме автоматов) эквивалентны, т.е. в одной из моделей можно записать интерпретатор другой.

Примеры и интерактивные демонстрации

Задачи «со звёздочкой»

Задача 2.1а * Привести пример неограниченного языка.
Задача 2.1б ** … и доказать, что язык неограниченный.

Задача 2.2а ** Построить интерпретатор одной модели вычислений в другой.
Задача 2.2б *** … в обе стороны.

Задача 2.3 ** Сделать интерактивный эмулятор одной из моделей вычислений (JavaScript).