Вводная про языки программирования

Косарев Дмитрий a.k.a. Kakadu

матмех СПбГУ

5 сентября 2019 г.

Первые языки программирования

- (?) Машинные коды
- Ассемблер

- Fortran (1957)
- LISP (1958)
- APL (1964)

Основное направление работы: человек пишет высокоуровненый (относительно ассемблера) код, а компилятор оптимизирует это под конкретное железо.

Язык Си (1972)

Смог сделать то, что не могли предоставить конкуренты в 1970х

- Оптимизации: у **программиста** появилась возможность оптимизировать руками
- Управление прерываниями
- Управление распределением ресурсов
- Управление распределением памяти

К 1960 году у нас было множество прекрасных языков: Лисп, APL, Фортран, Кобол, Алгол 60. Это языки более высокого уровня, чем Си. С момента появления Си мы серьезно деградировали. Си уничтожил нашу способность прогрессировать в автоматической оптимизации, автоматической параллелизации, автоматическом отображении языков высокого уровня в машинную архитектуру. Это одна из причин, по которым компиляторы, по сути, больше не изучаются в колледжах и университетах.

Из книги “Кодеры за работой”
(глава 13 “Фрэн Аллен”)

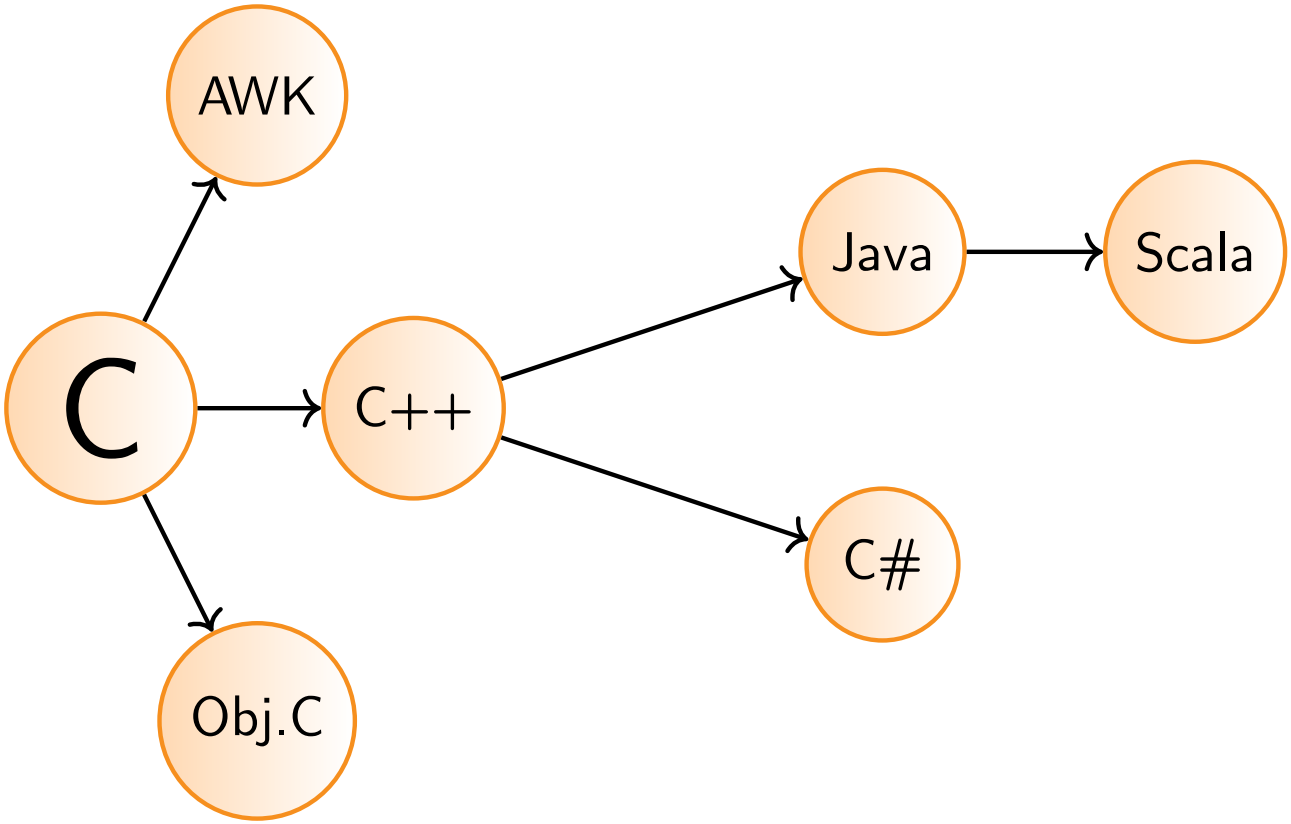
Язык Си (1972)

Изначально подразумевался для программирования в местах, где надо тесно общаться с железом
И всё было бы хорошо, если бы не одно но...

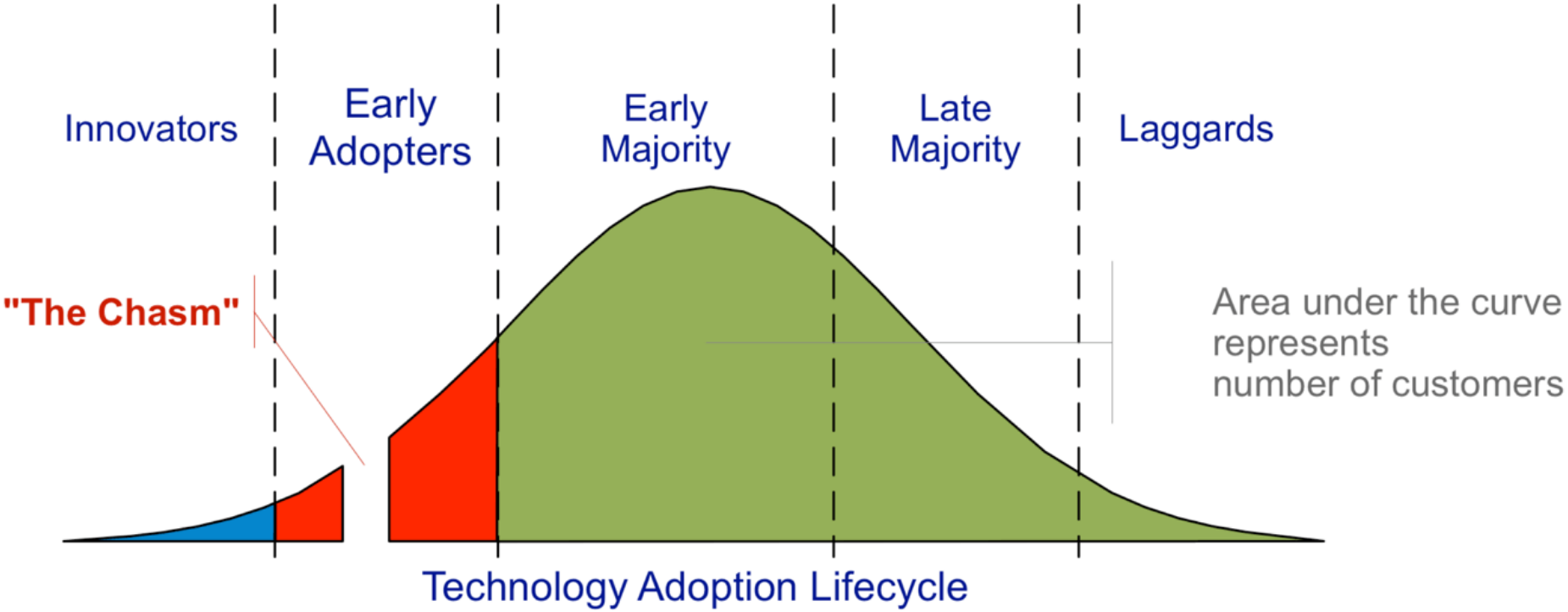
Язык Си (1972)

Изначально подразумевался для программирования в местах, где надо тесно общаться с железом

И всё было бы хорошо, если бы не одно но...



Про введение новых технологий (инженерный взгляд)



"Early adopters" "Pragmatists"

Из книги Geoffrey Moore "Crossing the Chasm"

Early adopters
(ранние приемники)

- Ведóмы радостью проведения исследования
- **Не полагаются** на известные авторитеты

“Прагматики”

- Ведóмы желанием “to get things done”
- **Полагаются** на известные авторитеты

Прагматики всякий раз предпочитают “предсказуемо плохие“ “блестящим, но непроверенным временем“ вариантам.

”Не устраивают текущие решения и ищут новые пути“ vs ”Думают, что всё знают о программировании“

Потребности трех сторон

Потребности трех сторон

Программистам

- Побольше дробь $\frac{\text{деньги}}{\text{трата нервов}}$ ✔
- Интересные задачи
- Удовлетворенность результатом

Заказчику

- Быстро
- Хорошо
- Поменьше стоимость

Менеджеру (outsourcing конторе)

- Тяп-ляп, и в продакшн
- Долгие отношения с заказчиком
- Поменьше платить программистам
- Легкая заменяемость программистов

Потребности трех сторон

Программистам

- Побольше дробь $\frac{\text{деньги}}{\text{трата нервов}}$ ✔
- Интересные задачи ✔
- Удовлетворенность результатом

Заказчику

- Быстро ?
- Хорошо ✔
- Поменьше стоимость

Менеджеру (outsourcing конторе)

- Тяп-ляп, и в продакшн ?
- Долгие отношения с заказчиком ✖
- Поменьше платить программистам
- Легкая заменяемость программистов ✖

Что из этого предоставляет функциональное программирование?

Не очень определение функционального программирования

Определение

Функциональные языки программирования позволяют обращаться с функциями как с объектами первого класса (first class citizens).

Не очень определение функционального программирования

Определение

Функциональные языки программирования позволяют обращаться с функциями как с объектами первого класса (first class citizens).

Не очень определение
Не очень понятно, что такое first class citizens.
Если это про передачу функций как значений, то это умеет даже Си

Вопрос к залу: что такое first class citizens?

Два семейства функциональных языков

ML

- Строгая система типов
- Часто используется некоторый математический аппарат при проектировании
- Пишем типы, чтобы не допускать ошибок
- Представители: SML, OCaml, ReasonML, F#, Haskell, Scala, Agda, Idris, Gallina

Scheme

- Традиционно бестиповые
- Математичность не так очевидна
- Проектируем мини-языки (=API) с помощью макросов, чтобы не допускать ошибок
- Представители: Scheme, Common LISP, Emacs LISP, Racket, Clojure

Этот курс про типизированные языки, а именно Haskell, OCaml/ReasonML, Scala 3 (sic!)

Языки по количеству фич

Некоторые больше любят минималистичные языки, в которых меньше "фич", а поэтому:

- их проще изучать
- компилятор проще и компактнее

Lua $<$ OCaml $<$ Haskell $\simeq$ Scala 2

Большие языки сложнее изучать

Миф про чистые функции

“В функциональном программировании все функции чистые”

Следствие: “программы на ФП тормозят”

Миф про математику

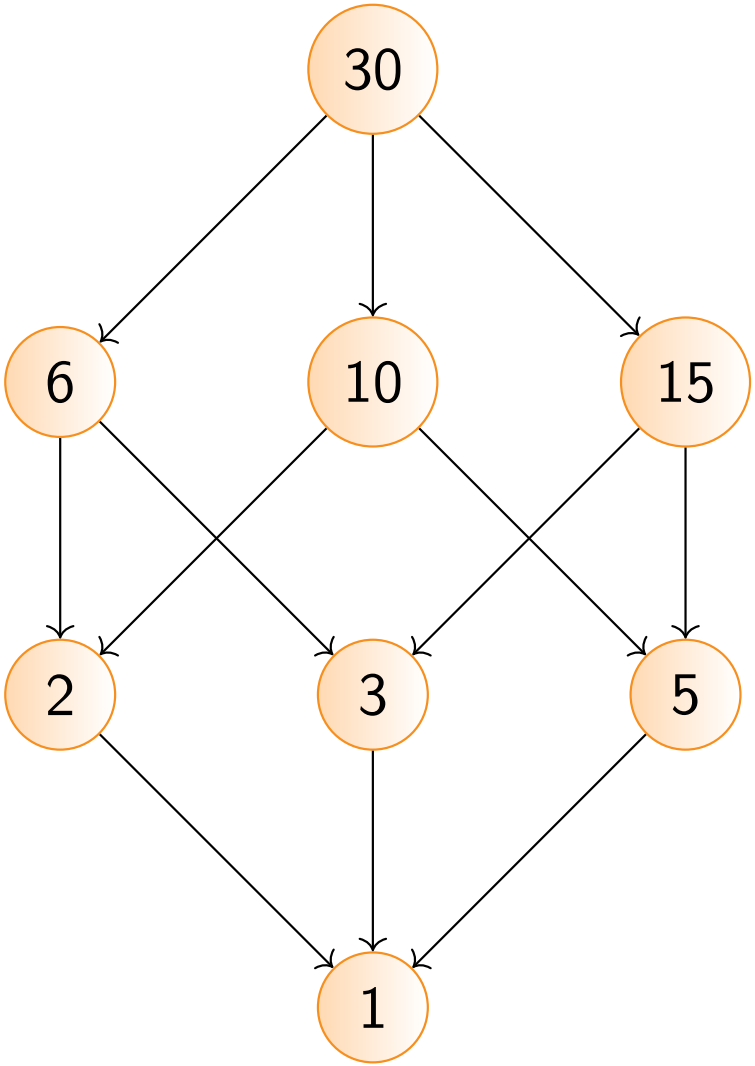
“Для осознания функционального программировании нужно быть кандидатом наук”

Вообще-то нет.

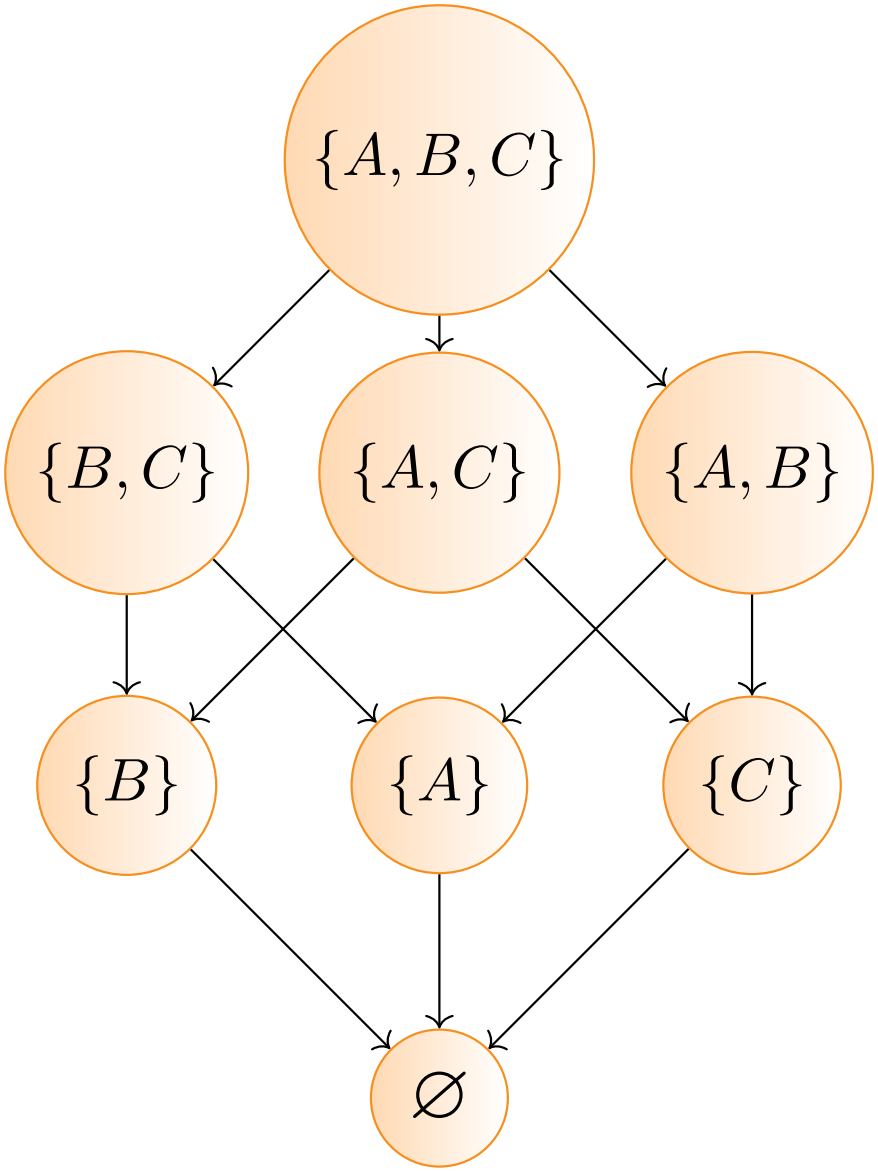
Математика бывает разная

- Простая
  - Для человека
  - Для компьютера
- Сложная
  - Для человека
  - Для компьютера

Дважды простая математика



Отчасти простая математика



A, B, C – это переменные формулы высказываний.

Каждая переменная может быть либо true, либо false.

Подмножества обозначают те переменные, которые истинны.

Всего 2^3 возможных значений переменных (и всего подмножеств).

Сложная для человека, простая для компьютера

Парадокс Монти-Холла

Даны три двери, но только за одной из них приз

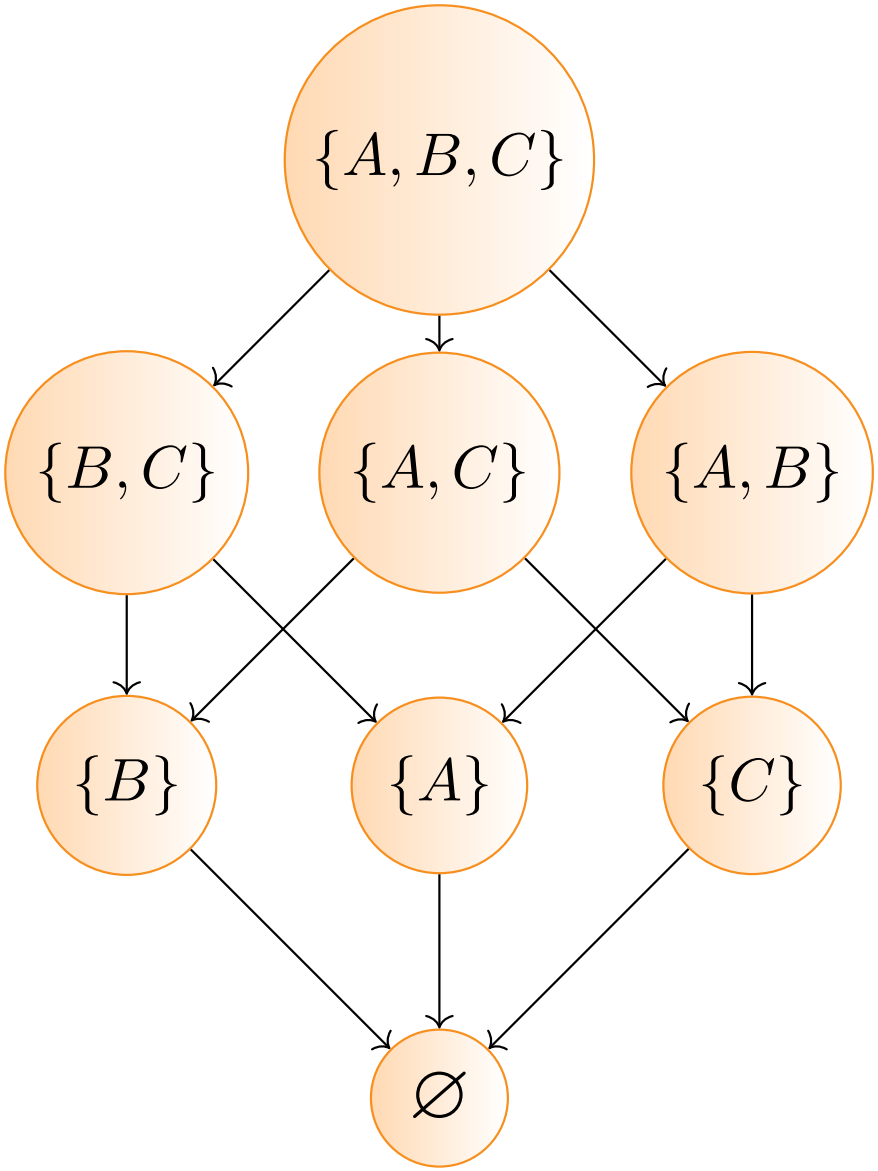
Вы указываете пальцем на какую-то дверь

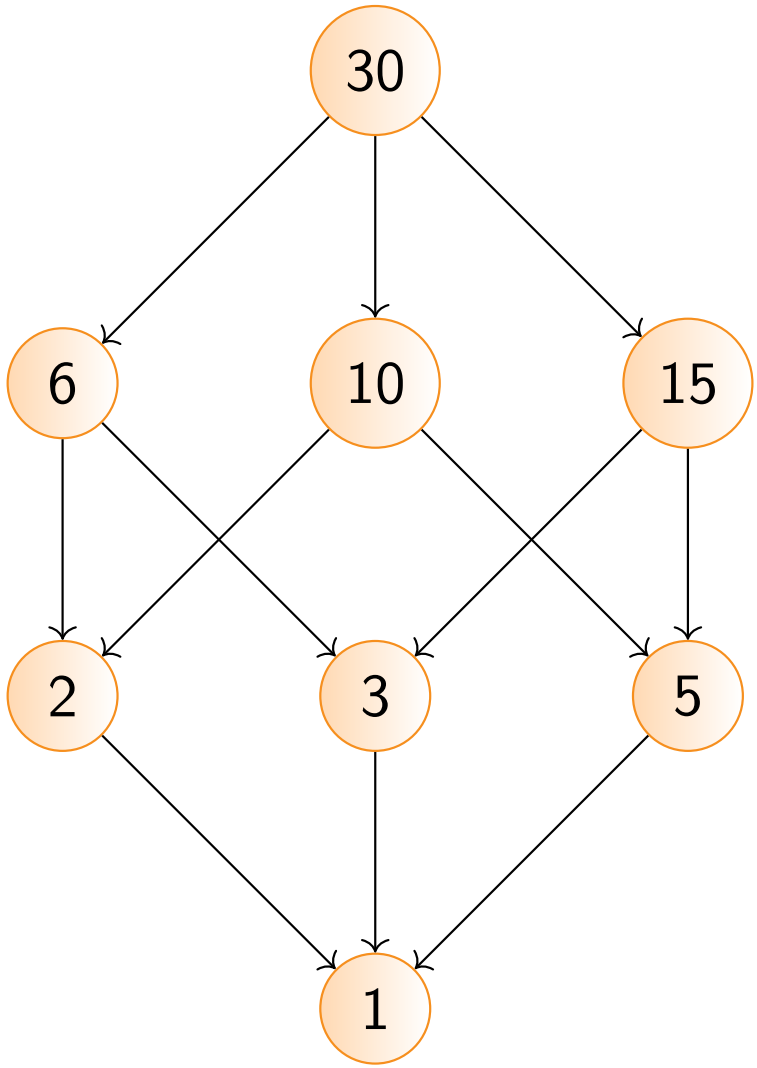
Ведущий открывают одну из двух других дверей, ту, которая без приза...

... и предлагает по желанию поменять выбор.

Стоит ли его менять?

Простая для человека, сложная для компьютера





Минизаключение про математику

Математика может быть 1) в задаче, которая решается, или 2) в инструментах решения.

Минизаключение про математику

Математика может быть 1) в задаче, которая решается, или 2) в инструментах решения.

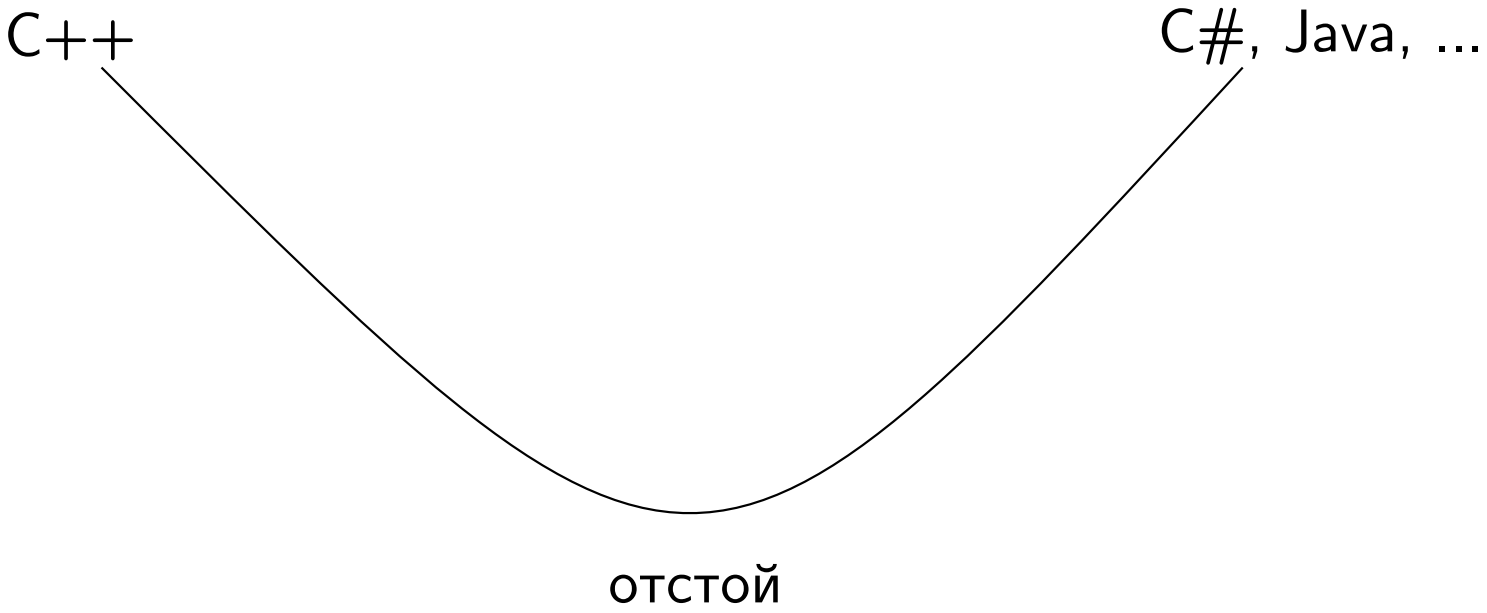
Исторически так сложилось, что функциональные языки проектируются с большей оглядкой на математику, чем мейнстримные императивные.

Поэтому, если надо запускать математичные исследования функциональных программ (например, верификацию), то результат бывает более математически красив и менее костылен.

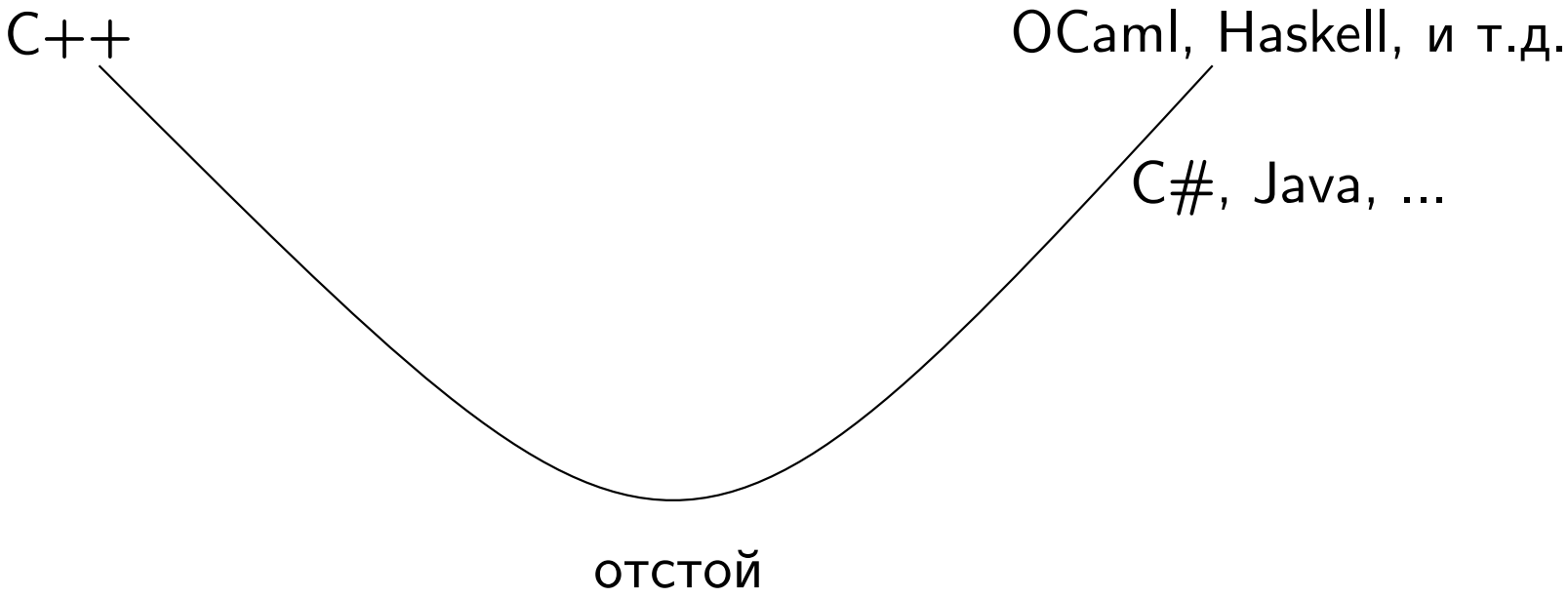
Языки для умных и обычные языки

Доклад к просмотру [4].

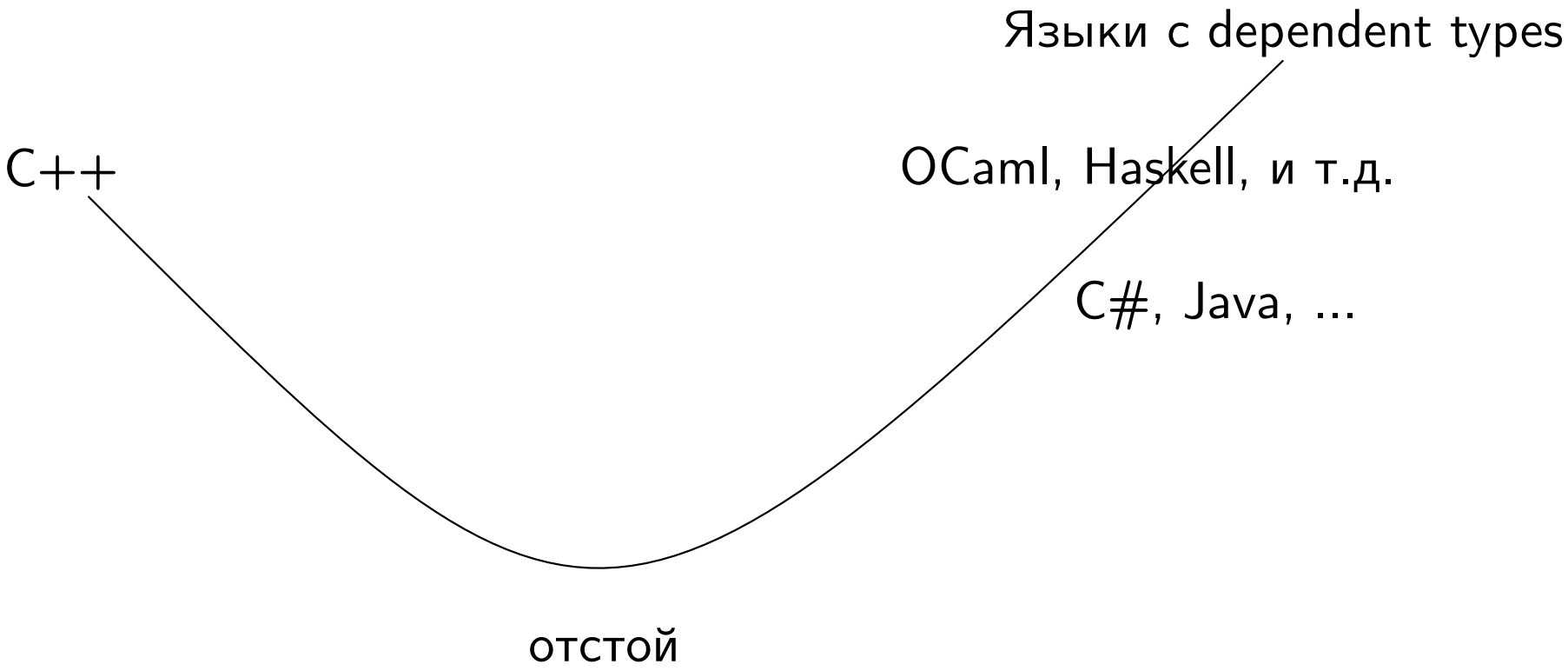
Языки для “умных” программистов vs. языки для “простых” программистов



Языки для умных и функциональные языки



Языки для умных и... для умных



Конец

Дальше только список литературы

Ссылки I

Why your ReasonML Evangelism isn't working
Gage Peterson
YouTube

Визуализация вляния языков друг на друга
ссылка

Category Theory in Life
Eugenia Cheng
YouTube

Preventing the Collapse of Civilization
Jonathan Blow
YouTube

Ссылки II

Advice for Haskell beginners

Gabriel Gonzalez

ссылка

Detailed walkthrough for a beginner Haskell program

Gabriel Gonzalez

ссылка