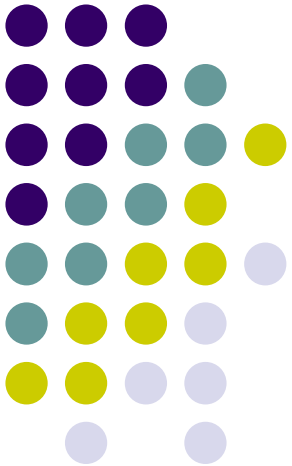
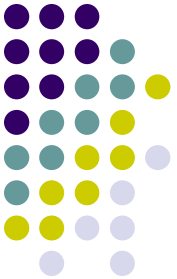
Современные динамические языки

Денис С. Мигинский

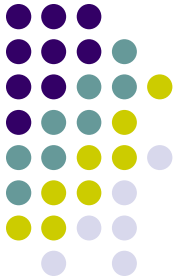


Вопросы к размышлению



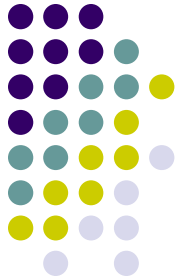
- Чем принципиально различаются языки?
- Сколько нужно языков?
- Как выбирать языки?

Классификации языков



- Поддерживаемые парадигмы
- Назначение
- Модель исполнения
- Выразительность
- Сложность
- Производительность
- …

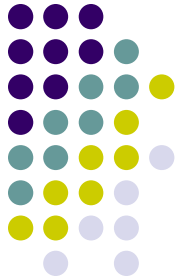
Сложность языка



Синтаксическая сложность – количество правил, определяющих синтаксис языка

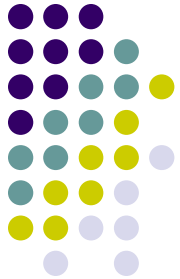
Семантическая сложность – количество элементов языка и/или стандартной библиотеки, без которых язык не может быть успешно использован

Синтаксическая сложность языков



- Forth (1 правило)
- Lisp (2 правила)
- Tcl (11 правил)
- C++ (~100 правил)
- COBOL (см. фразеологический словарь английского языка)

Выразительность языка



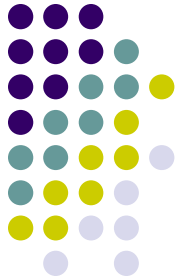
Выразительность языка – среднее количество программного кода на единицу функциональности программы

При этом код должен быть легко создаваем и читаем (в качестве антипримера см. Malbolge)

“Hello, world!” (Malbolge):

```
(=<`:9876Z4321UT.-
Q+*)M'&%$H"!~}|Bzy?=|{z]KwZY44Eq0/{mlk**hKs_dG5[m_BA{?-
Y;;Vb'rR5431M}/.zHGwEDCBA@98\6543W10/.R,+O<
```

Связь выразительности, простоты и всего остального

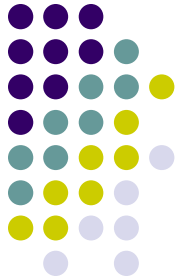


Чем ниже синтаксическая сложность языка, тем он выразительнее (эмпирическое наблюдение)

Чем более специализирован язык, тем он более выразителен

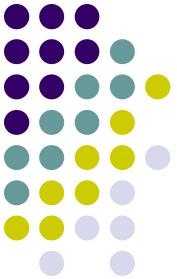
НО, чем выразительнее язык, тем больше возможностей написать запутанный и трудноотлаживаемый код

Классификация языков по модели исполнения



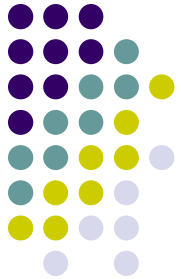
- Интерпретируемые
- Компилируемые
- С гибридной моделью исполнения

Вопросы



- Чем компилируемый язык отличается от интерпретируемого?
- Примеры того и другого?

http://en.wikipedia.org/wiki/Interpreter_(computing)



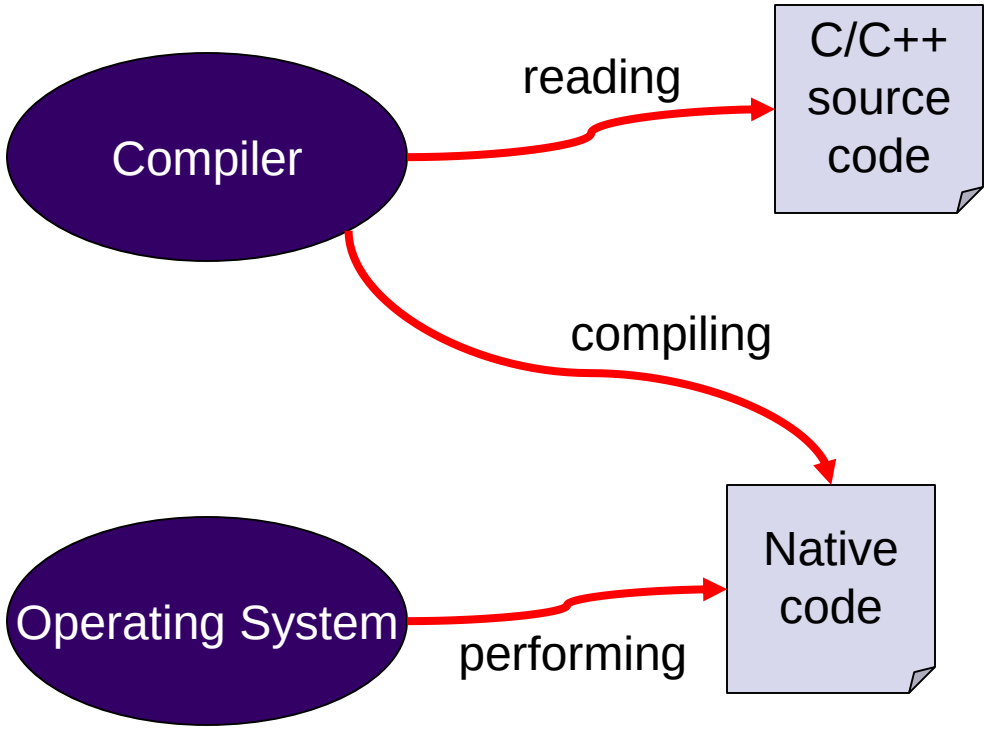
An interpreter normally means a computer program that executes, i.e. *performs*, instructions written in a programming language.

An interpreter may be a program that either:

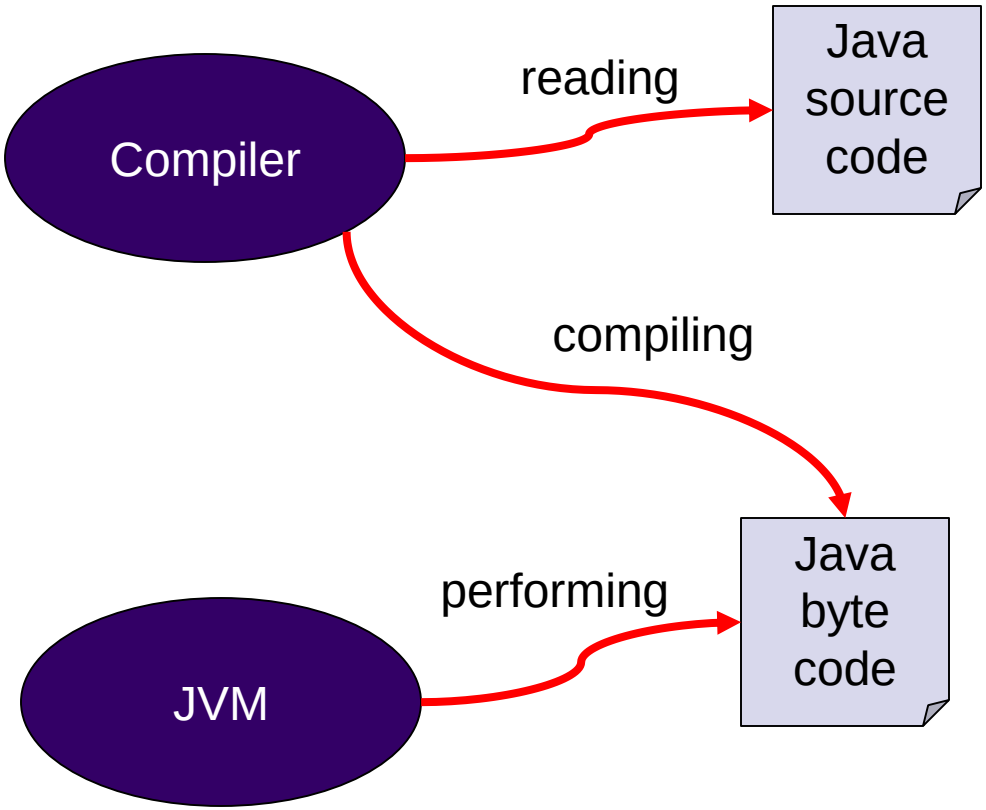
1. executes the source code directly
2. translates source code into some efficient intermediate representation (code) and immediately executes this
3. explicitly executes stored precompiled code[1] made by a compiler which is part of the interpreter system

[1] *In this sense, the CPU is also an interpreter, of machine instructions*

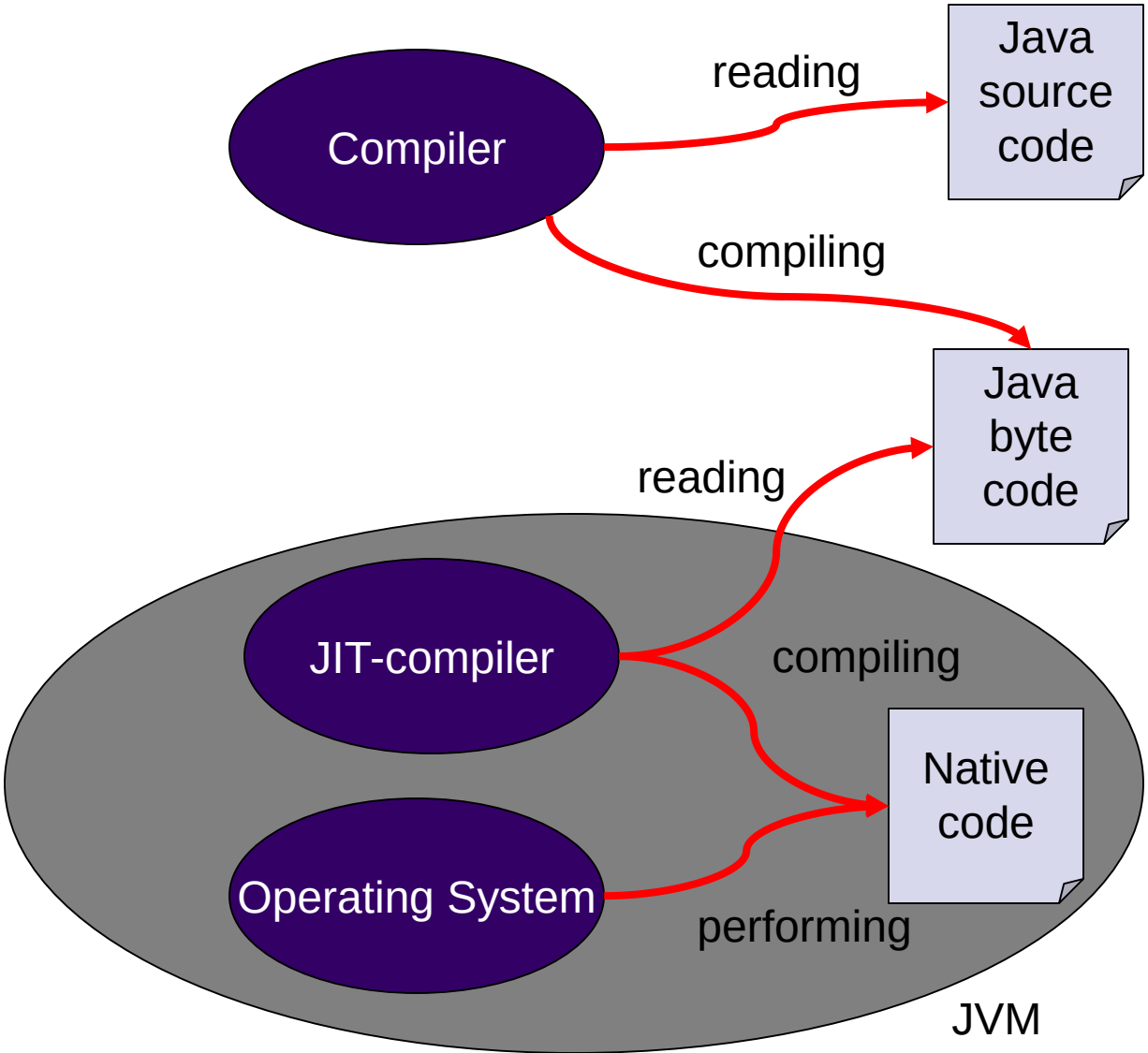
Модель исполнения C/C++



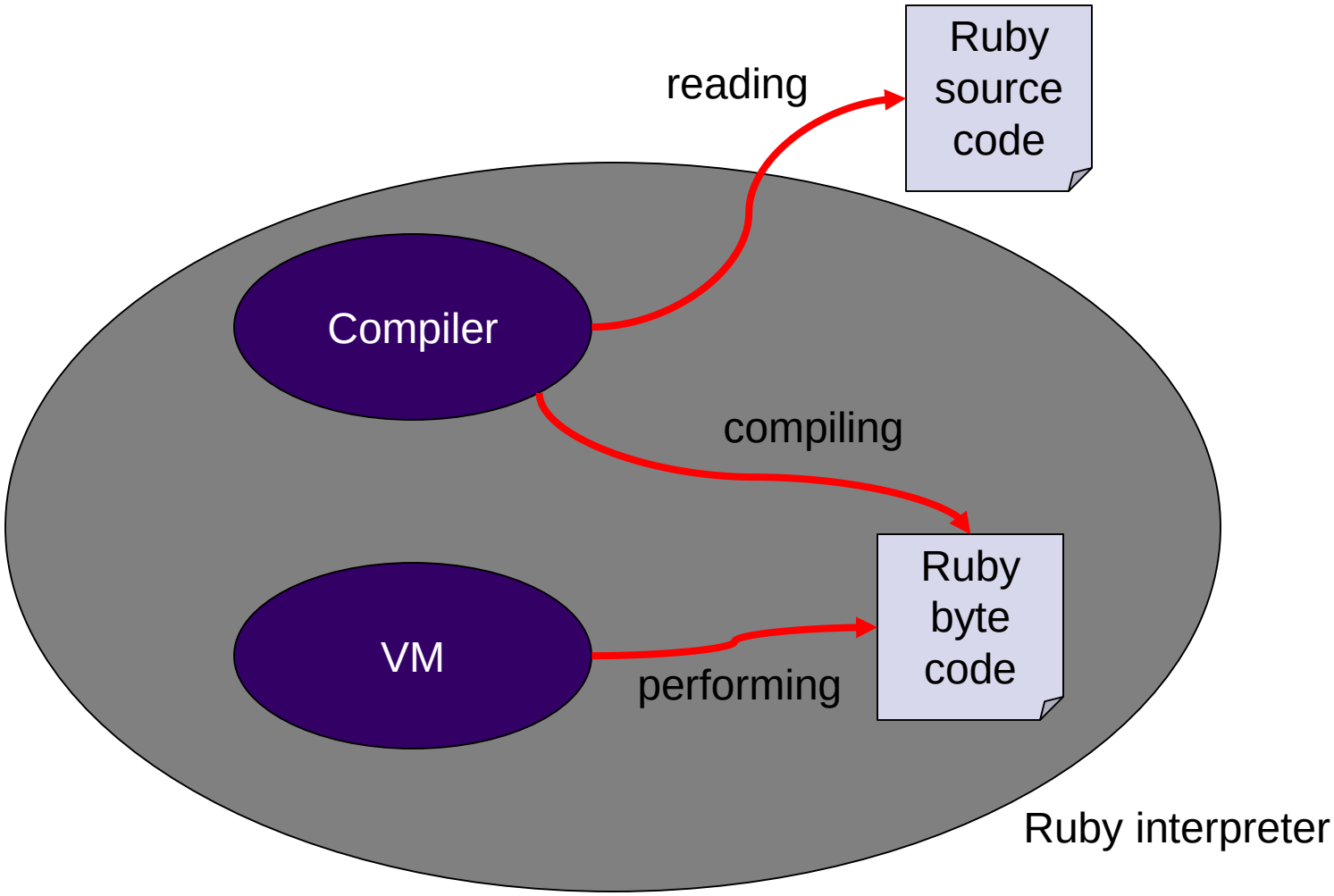
Модель исполнения Java



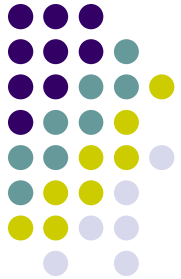
Модель исполнения Java: подробнее



Модель исполнения Ruby

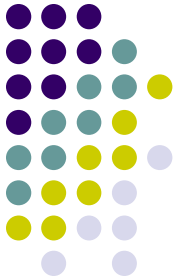


Классификация языков по назначению



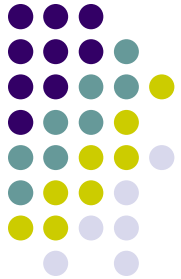
- Языки «общего» назначения (C++, Java, C#)
- Языки сценариев
- Языки представления данных (XML, JSON)
- Языки запросов (SQL, XPath)
- …

Вопросы



- Что такое языки общего назначения?
- Что такое языки сценариев?

Языки общего назначения

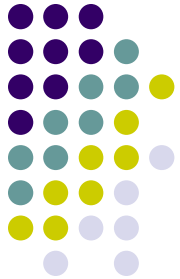


В теории: языки применимые для решения любых/широкого круга задач

На практике: языки, которые применяют для решения всех подряд задач независимо от эффективности

Вывод: следует для каждой задачи выбирать наиболее подходящий язык(-и), для чего следует ориентироваться в современных языках

Языки сценариев



Изначально: языки, встраиваемые в программные системы для записи сценариев

Сейчас:

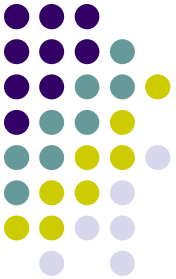
•встраиваемые языки (shell для ОС, JS для Web-browser)

•DSL (domain-specific languages)

•динамические языки/языки сценариев «общего» назначения (Perl, Ruby, Python, Tcl, Forth, Lua)

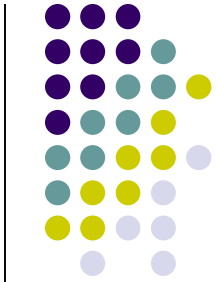
•…

Характерные особенности динамических языков



- Возможности динамической генерации и интерпретации кода (evaluate)
- Динамическая типизация
- Поддержка элементов метапрограммирования
- Возможности функционального программирования
- Генерация байт-кода «на лету», иногда – JIT-компиляция

Виды типизации



- сильная – для любой операции контролируется тип аргументов
- слабая – тип аргументов не контролируется, или контролируется не всегда
- статическая – проверка типов осуществляется на стадии компиляции
- динамическая – проверка типов осуществляется на стадии исполнения