Domain-Specific Language

- Parsing, grammars, code generation and optimization
- principle to allow you self-study classical compilers.

DSL: offers constructs tailored to a domain.

* SQL: relations and query expressions

Flavor of DSL:

→ May look like a compiled language: SQL

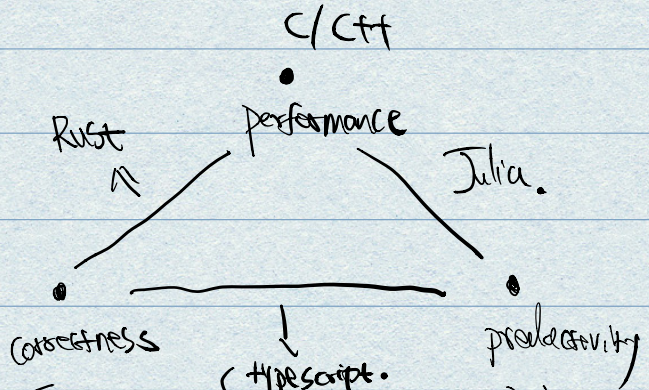
→ May look like a library: RxJS

→ May look like a hybrid: Hadoop.

Problem addressed by recent language

1. Memory management (GC instead of malloc/free)
2. Speed → maintainability
3. availability of JVM/LLVM.
4. New users
5. hardwares
6. type safety.

⇒ Before: Can get one of 3

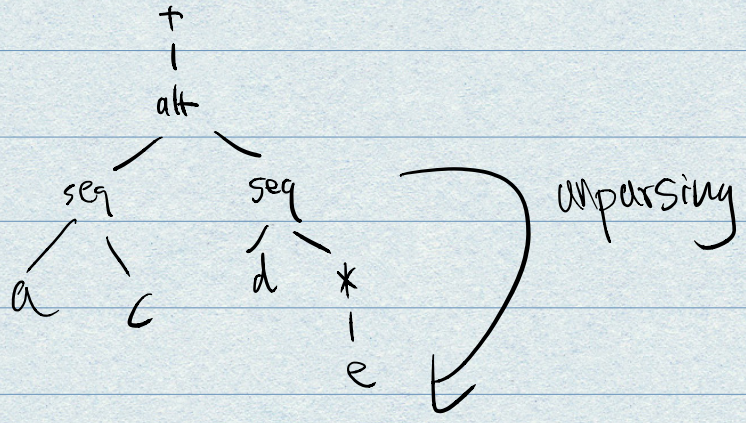


Java } Kotlin Hack Python

→ How to implement a PL.

① Abstract Syntax Tree. (AST)

regular expression:



⇒ Regex: $(ac \mid dex)+$

Why a tree: defines order of evaluation.

Why no parens in the AST. - - ↑

② The frontend.

Converts the syntax to AST by Parsing or embedding.

- Parsing: (ab*)*
- Embedding: RE.match('a').then('b')...

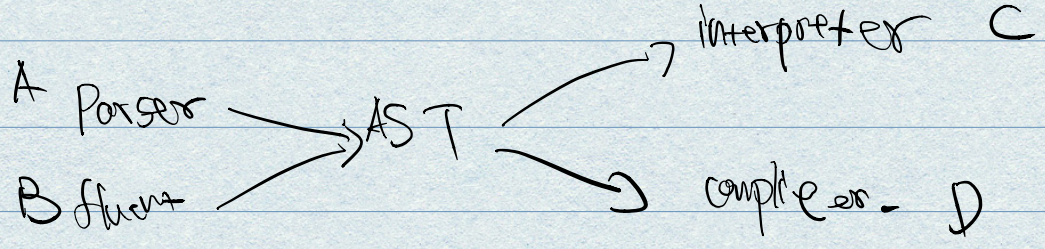
③ Backend: Evaluates the AST (interpreter / gen)

Homeworks:

① Regex. HW1 ② Data Engine HW2 ③ Templating Engine. HW3

④ Reactive Language HW4 ⑤. Final Project.

HW5



- HW1: B → C
- HW2: B → D
- HW3: A → D
- HW4: B → C