Lecture 27 – Course Review

COSE212: Programming Languages

Jihyeok Park

2025 Fall

Goal of This Course

PLRG

To learn **essential concepts** of **programming languages**

- Why? After this course, you will be able to:
  - **learn new programming languages** quickly.
  - **evaluate** and pick the best language for a given task.
  - **design** your own **specialized languages** for specific tasks.
- How? You will learn how to:
  - **design** programming languages in a **mathematical** way.
  - **implement** their **interpreters** using **Scala**.

Summary

(Part 1) Untyped Languages	(Part 2) Typed Languages
Arithmetic Expressions **AE** (Lecture 2 & 3)	

Summary

(Part 1) Untyped Languages	(Part 2) Typed Languages

Arithmetic Expressions

AE

(Lecture 2 & 3)

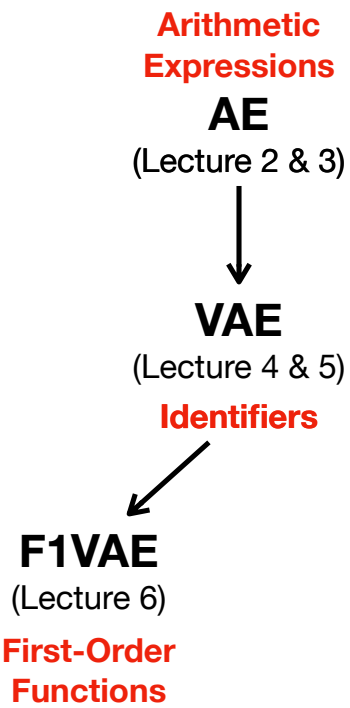
VAE

(Lecture 4 & 5)

Identifiers

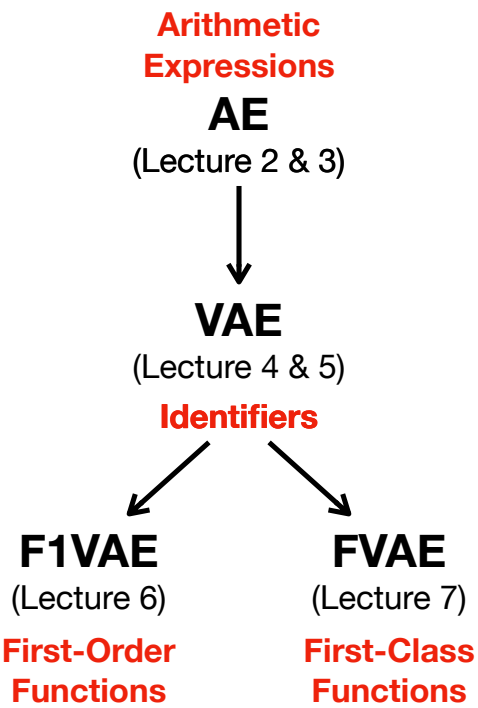
Summary

(Part 1) Untyped Languages	(Part 2) Typed Languages



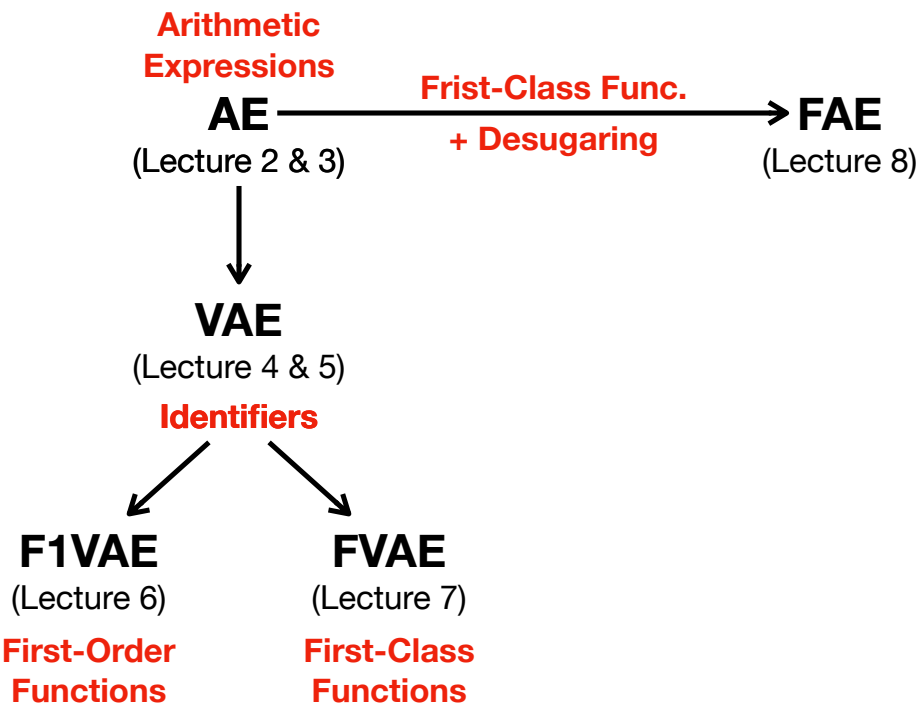
Summary

(Part 1) Untyped Languages	(Part 2) Typed Languages



Summary

(Part 1)
Untyped Languages

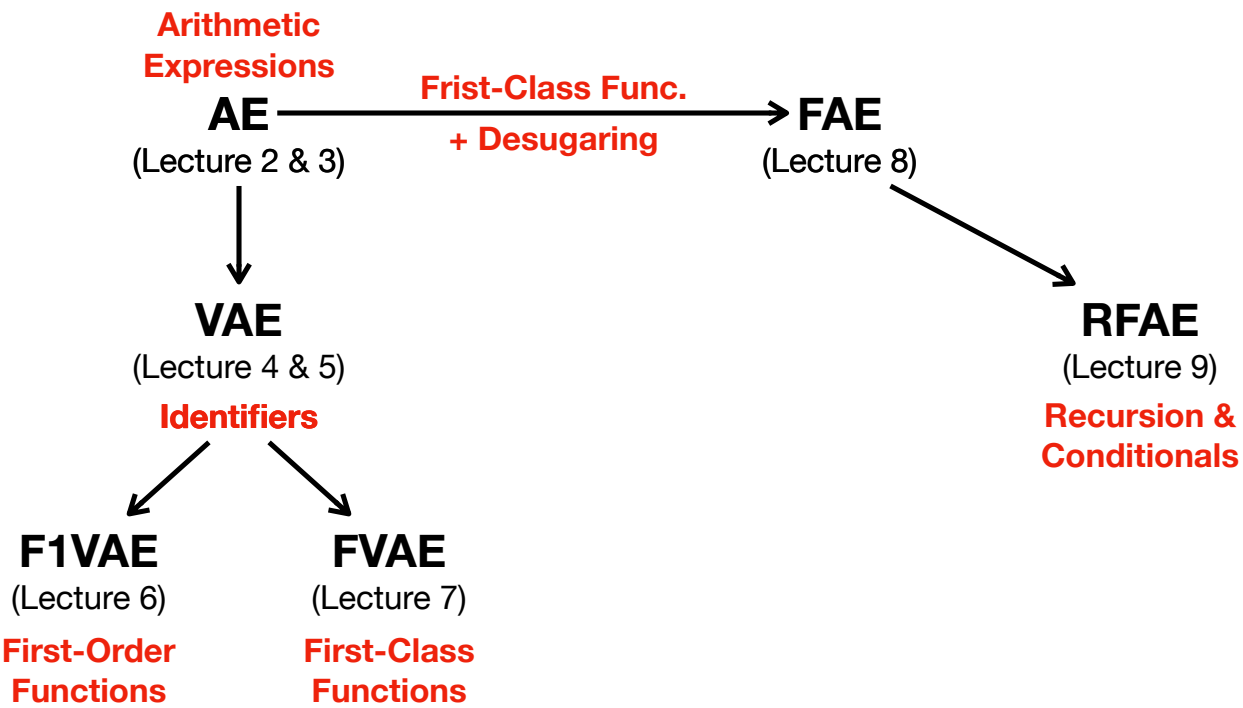


(Part 2)
Typed Languages

Summary

(Part 1)
Untyped Languages

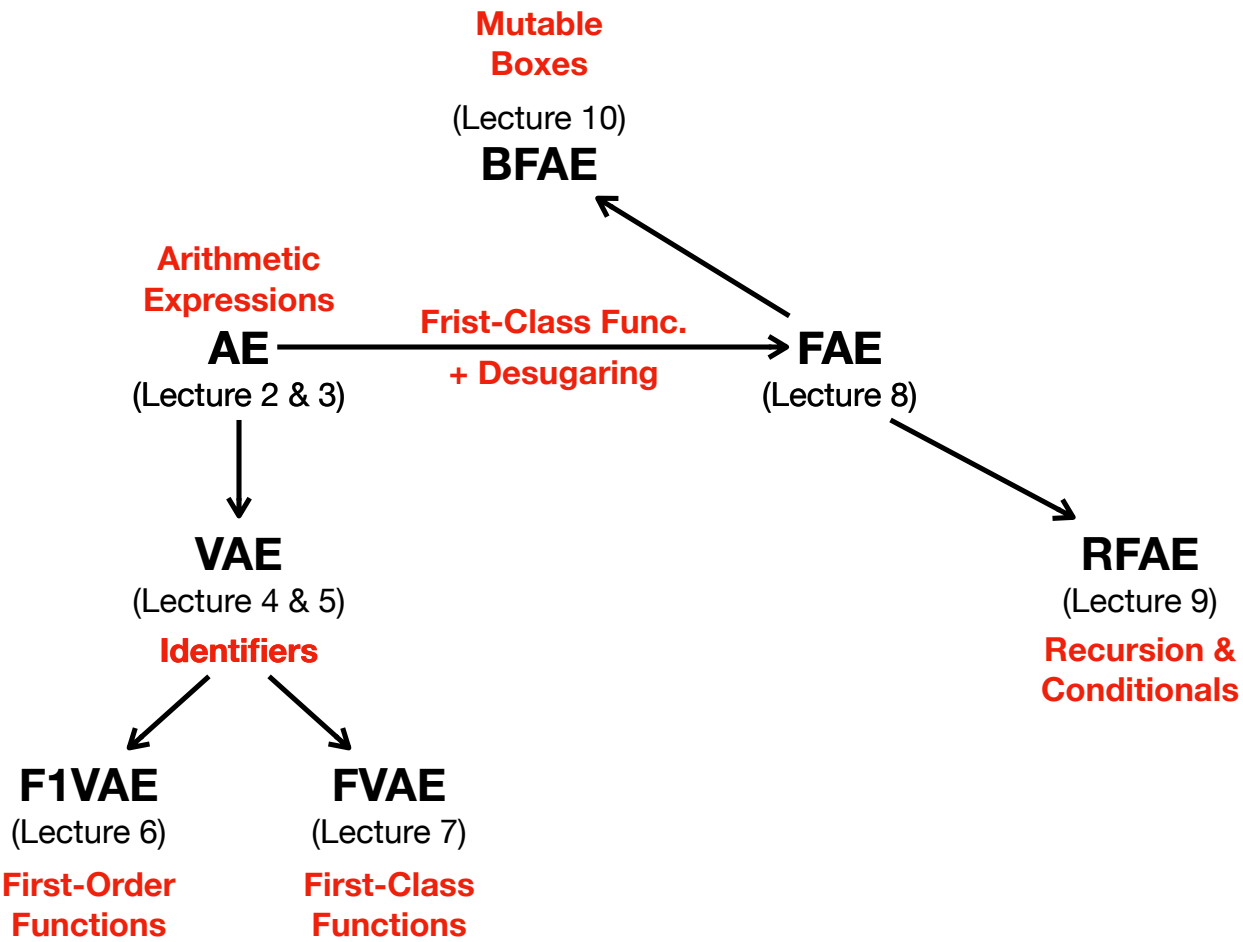
(Part 2)
Typed Languages



Summary

(Part 1)
Untyped Languages

(Part 2)
Typed Languages

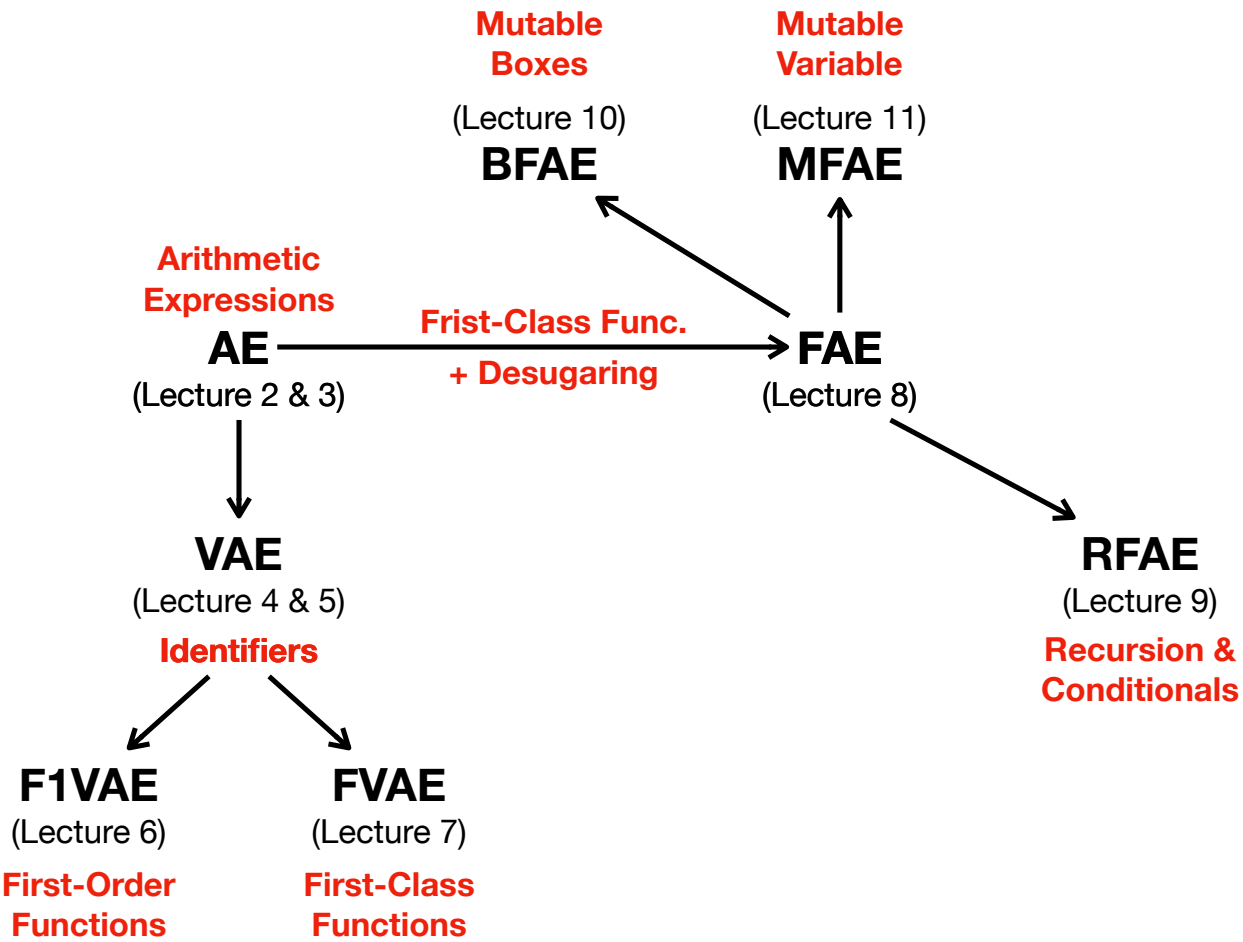


Summary

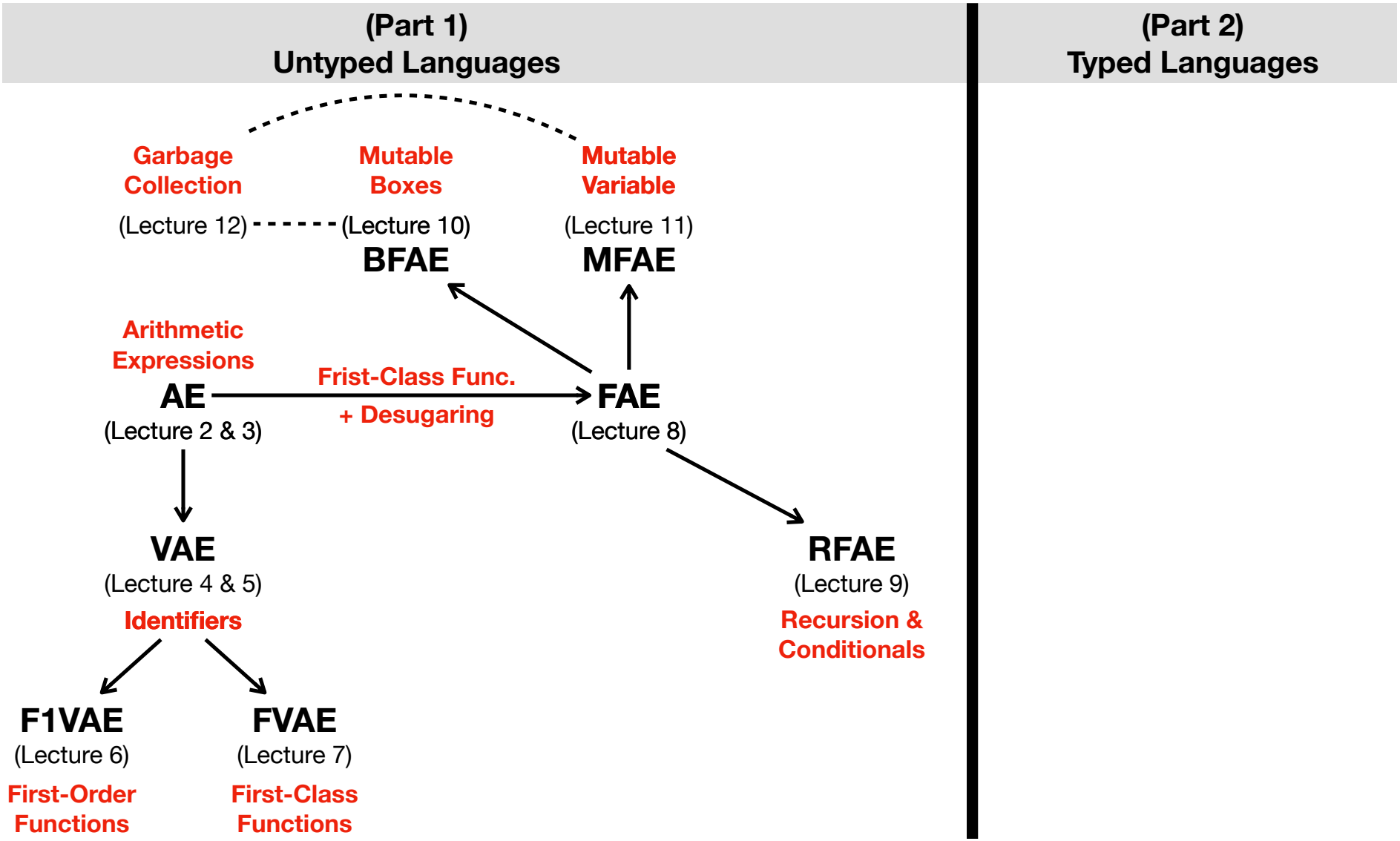
PLRG

(Part 1)
Untyped Languages

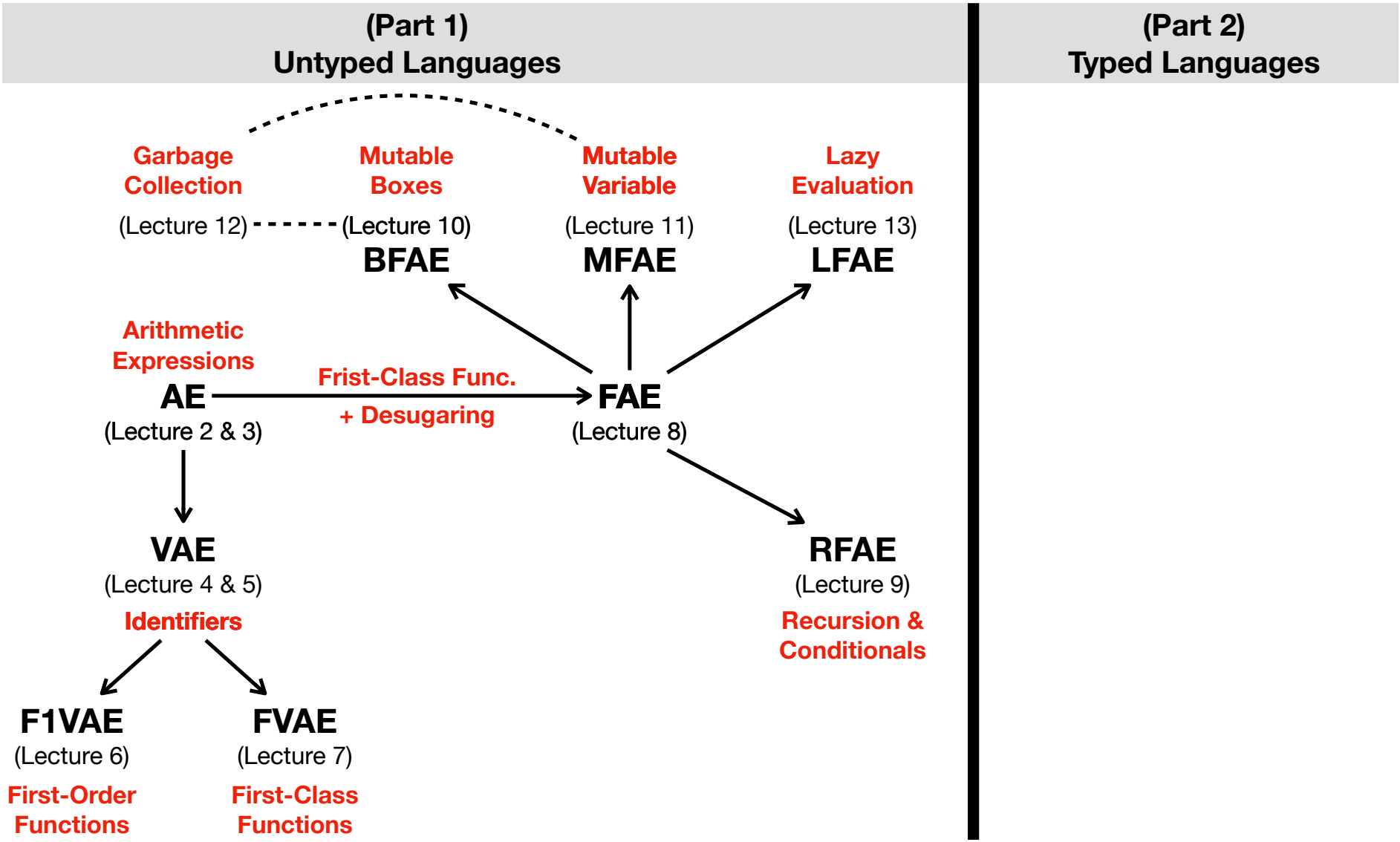
(Part 2)
Typed Languages



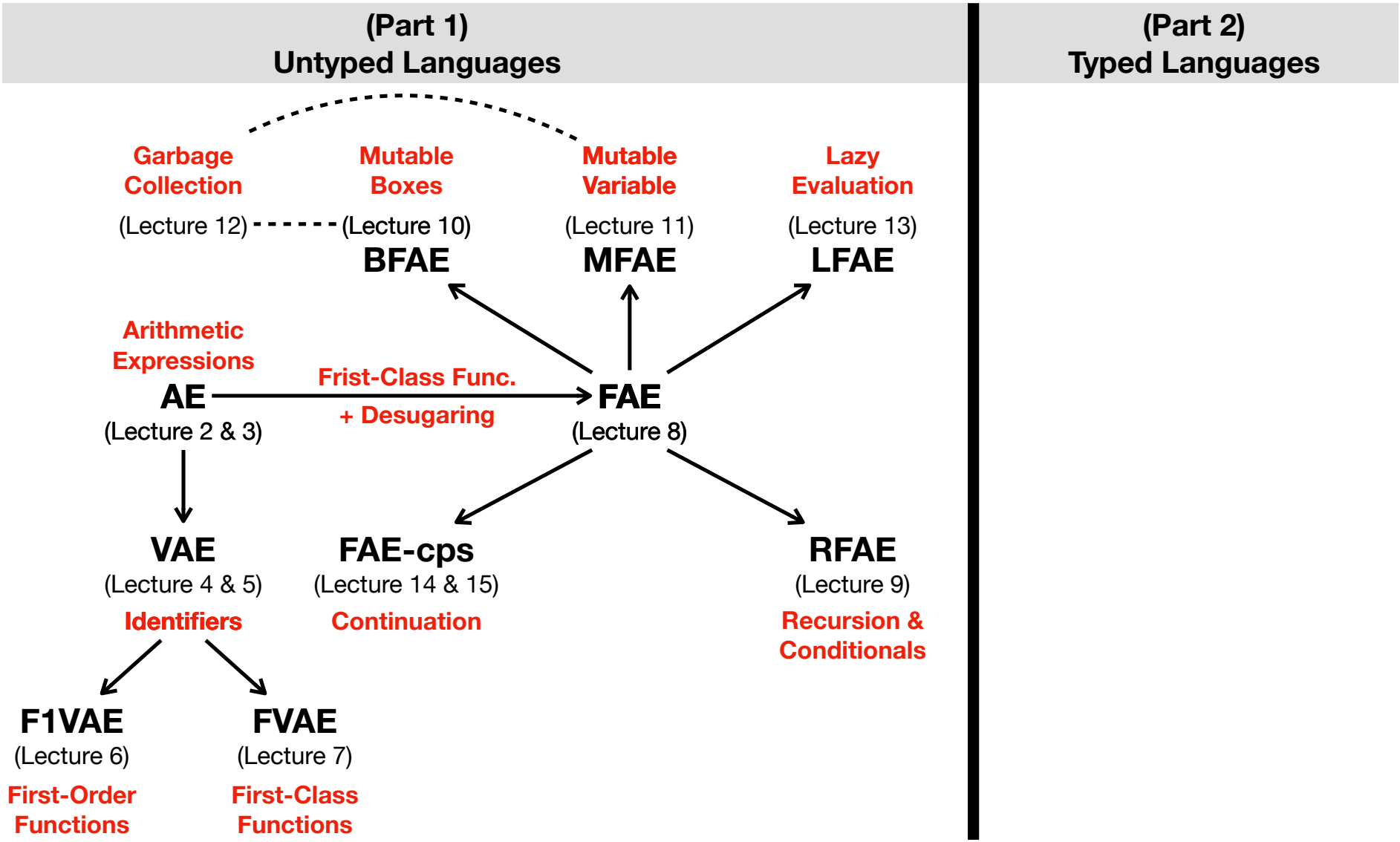
Summary


(Part 1)
Untyped Languages
(Part 2)
Typed Languages
Garbage Collection
(Lecture 12)
Mutable Boxes
(Lecture 10)
Mutable Variable
(Lecture 11)
BFAE
MFAE
Arithmetic Expressions
AE
(Lecture 2 & 3)
Frist-Class Func.
+ Desugaring
FAE
(Lecture 8)
VAE
(Lecture 4 & 5)
Identifiers
RFAE
(Lecture 9)
Recursion & Conditionals
F1VAE
(Lecture 6)
First-Order Functions
FVAE
(Lecture 7)
First-Class Functions

Summary


(Part 1)
Untyped Languages
(Part 2)
Typed Languages
Garbage Collection
(Lecture 12)
Mutable Boxes
(Lecture 10)
BFAE
Mutable Variable
(Lecture 11)
MFAE
Lazy Evaluation
(Lecture 13)
LFAE
Arithmetic Expressions
AE
(Lecture 2 & 3)
Frist-Class Func.
+ Desugaring
FAE
(Lecture 8)
VAE
(Lecture 4 & 5)
Identifiers
RFAE
(Lecture 9)
Recursion & Conditionals
F1VAE
(Lecture 6)
First-Order Functions
FVAE
(Lecture 7)
First-Class Functions

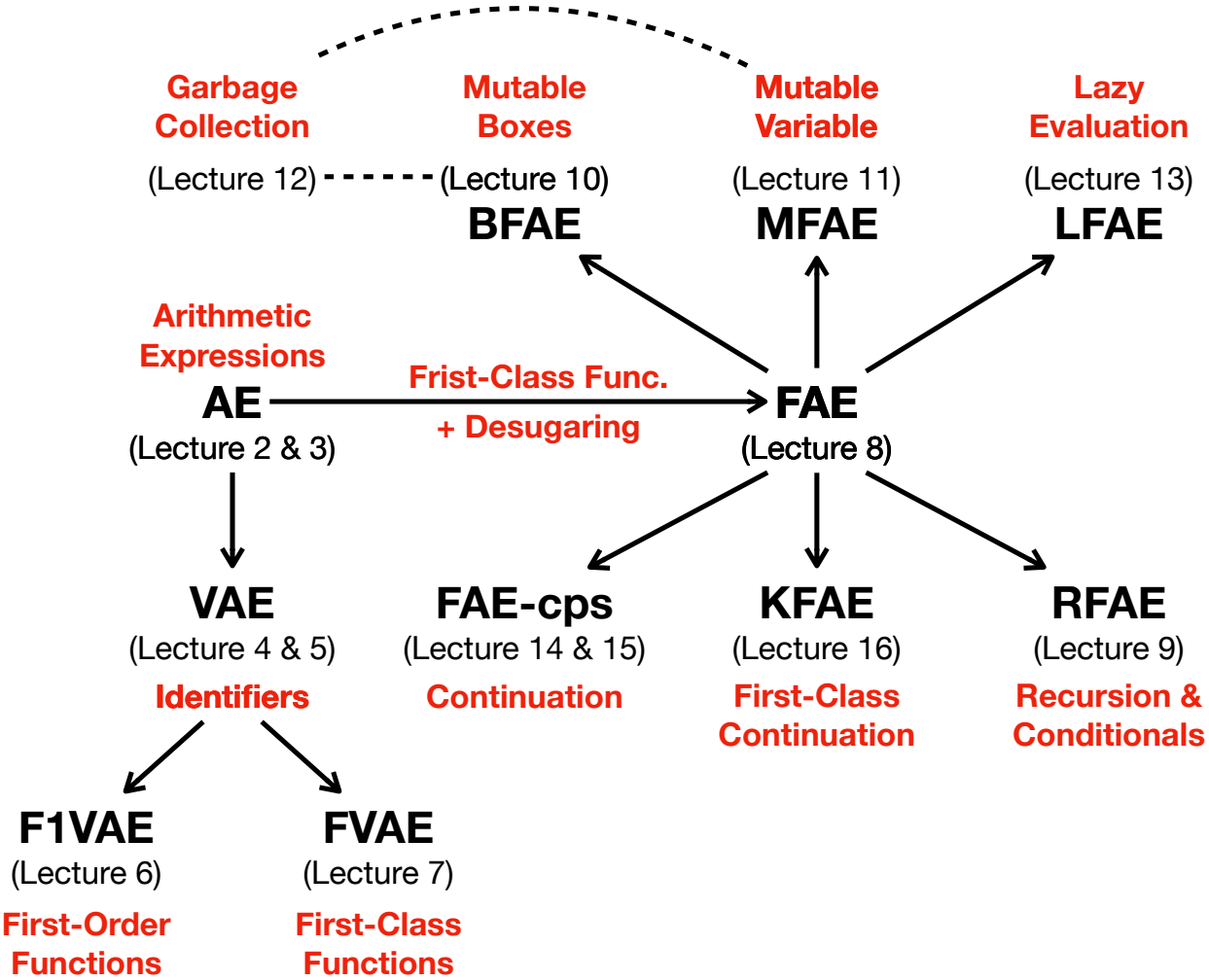
Summary


(Part 1)
Untyped Languages
(Part 2)
Typed Languages
Garbage Collection
(Lecture 12)
Mutable Boxes
(Lecture 10)
BFAE
Mutable Variable
(Lecture 11)
MFAE
Lazy Evaluation
(Lecture 13)
LFAE
Arithmetic Expressions
AE
(Lecture 2 & 3)
Frist-Class Func.
+ Desugaring
FAE
(Lecture 8)
VAE
(Lecture 4 & 5)
Identifiers
FAE-cps
(Lecture 14 & 15)
Continuation
RFAE
(Lecture 9)
Recursion & Conditionals
F1VAE
(Lecture 6)
First-Order Functions
FVAE
(Lecture 7)
First-Class Functions

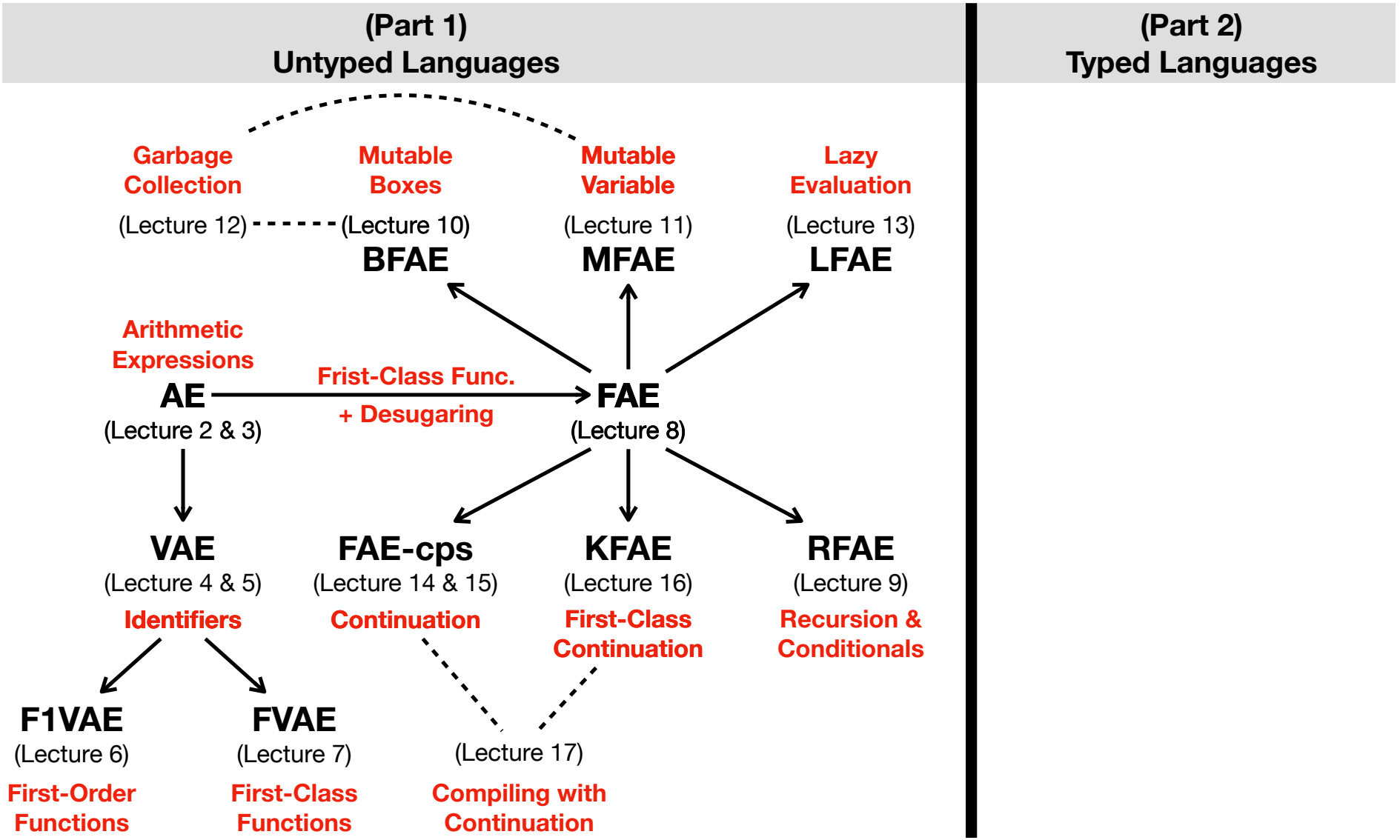
Summary

(Part 1)
Untyped Languages

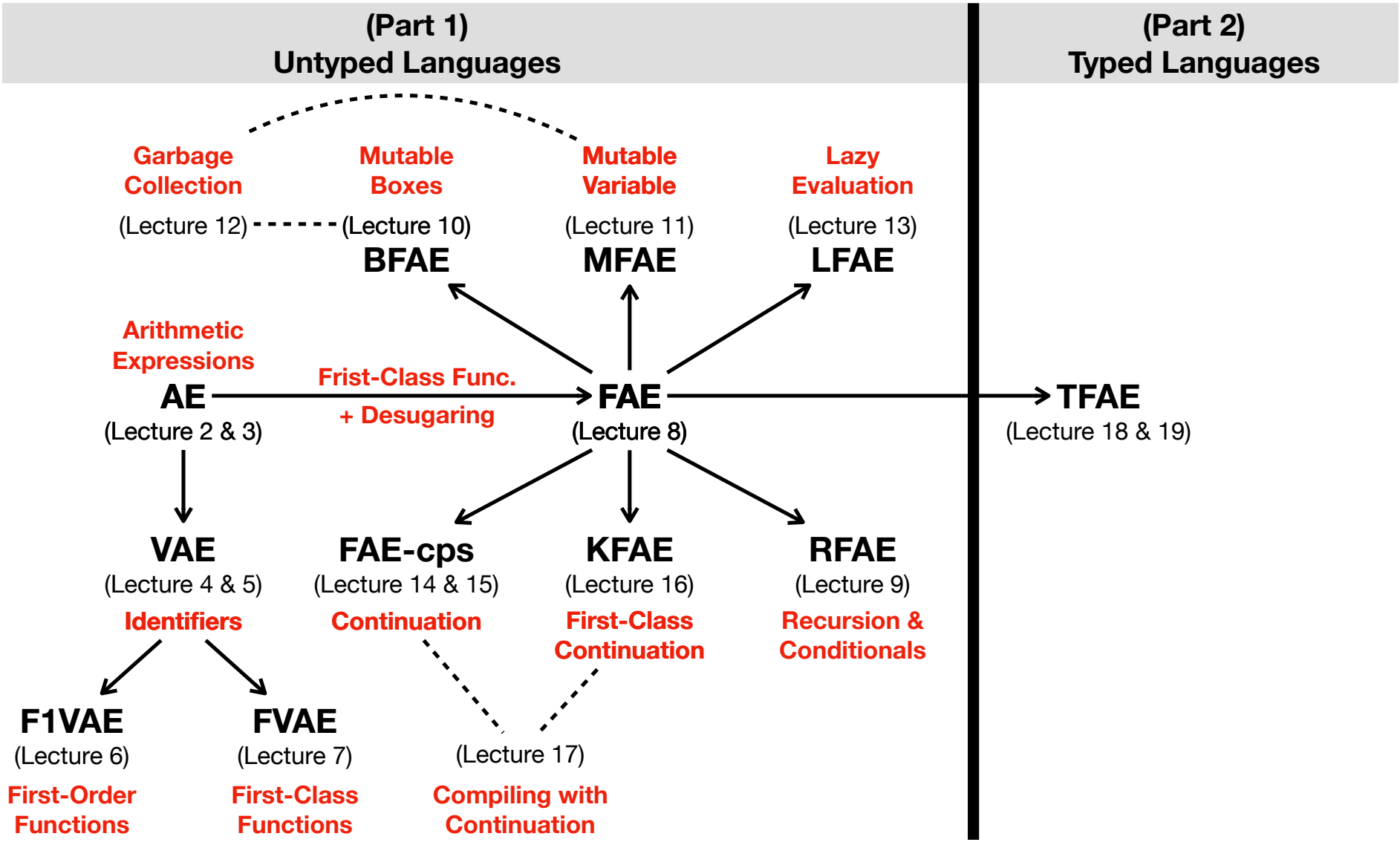
(Part 2)
Typed Languages



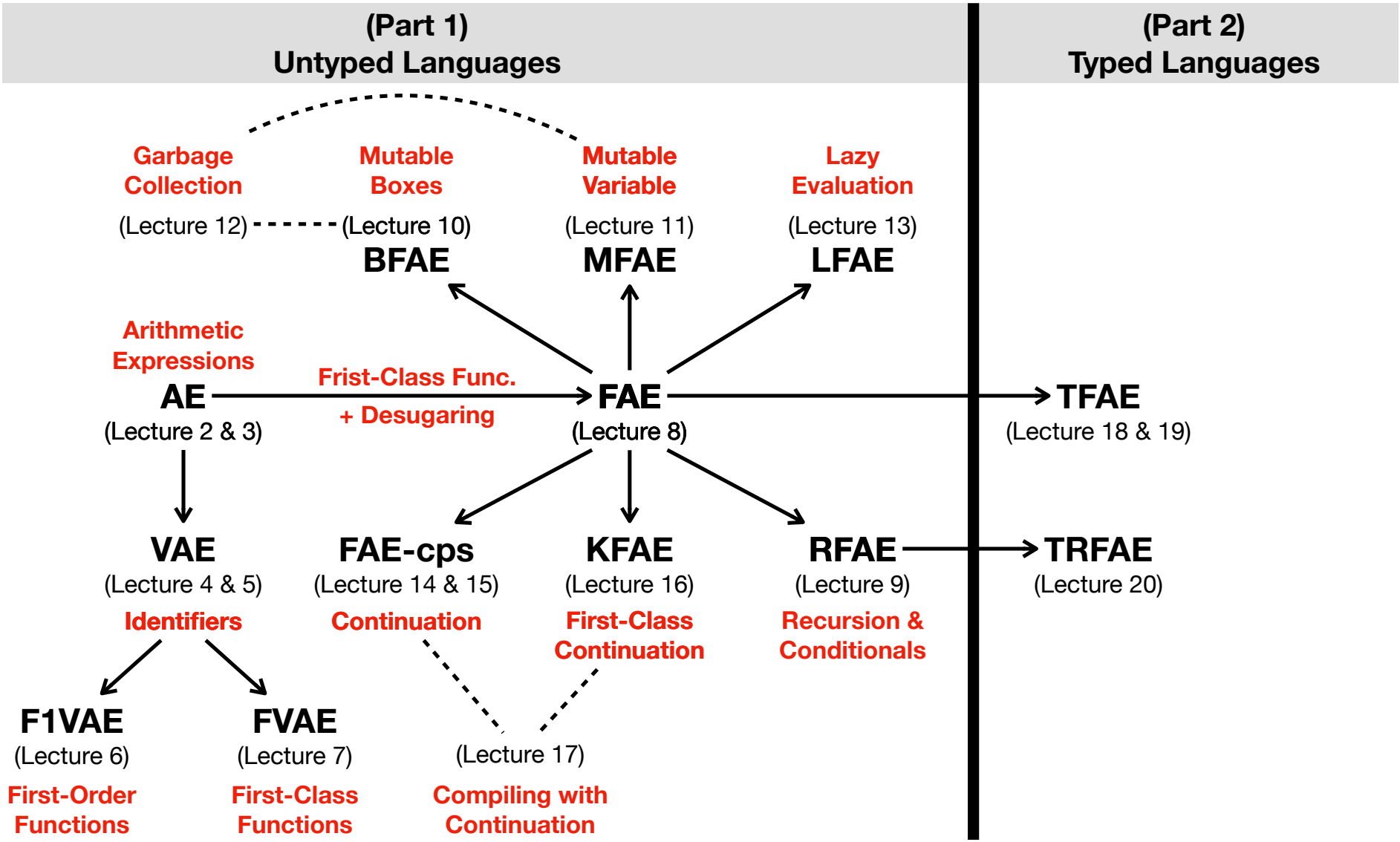
Summary


(Part 1)
Untyped Languages
(Part 2)
Typed Languages
Garbage Collection
(Lecture 12)
Mutable Boxes
(Lecture 10)
BFAE
Mutable Variable
(Lecture 11)
MFAE
Lazy Evaluation
(Lecture 13)
LFAE
Arithmetic Expressions
AE
(Lecture 2 & 3)
Frist-Class Func.
+ Desugaring
FAE
(Lecture 8)
VAE
(Lecture 4 & 5)
Identifiers
FAE-cps
(Lecture 14 & 15)
Continuation
KFAE
(Lecture 16)
First-Class Continuation
RFAE
(Lecture 9)
Recursion & Conditionals
F1VAE
(Lecture 6)
First-Order Functions
FVAE
(Lecture 7)
First-Class Functions
(Lecture 17)
Compiling with Continuation

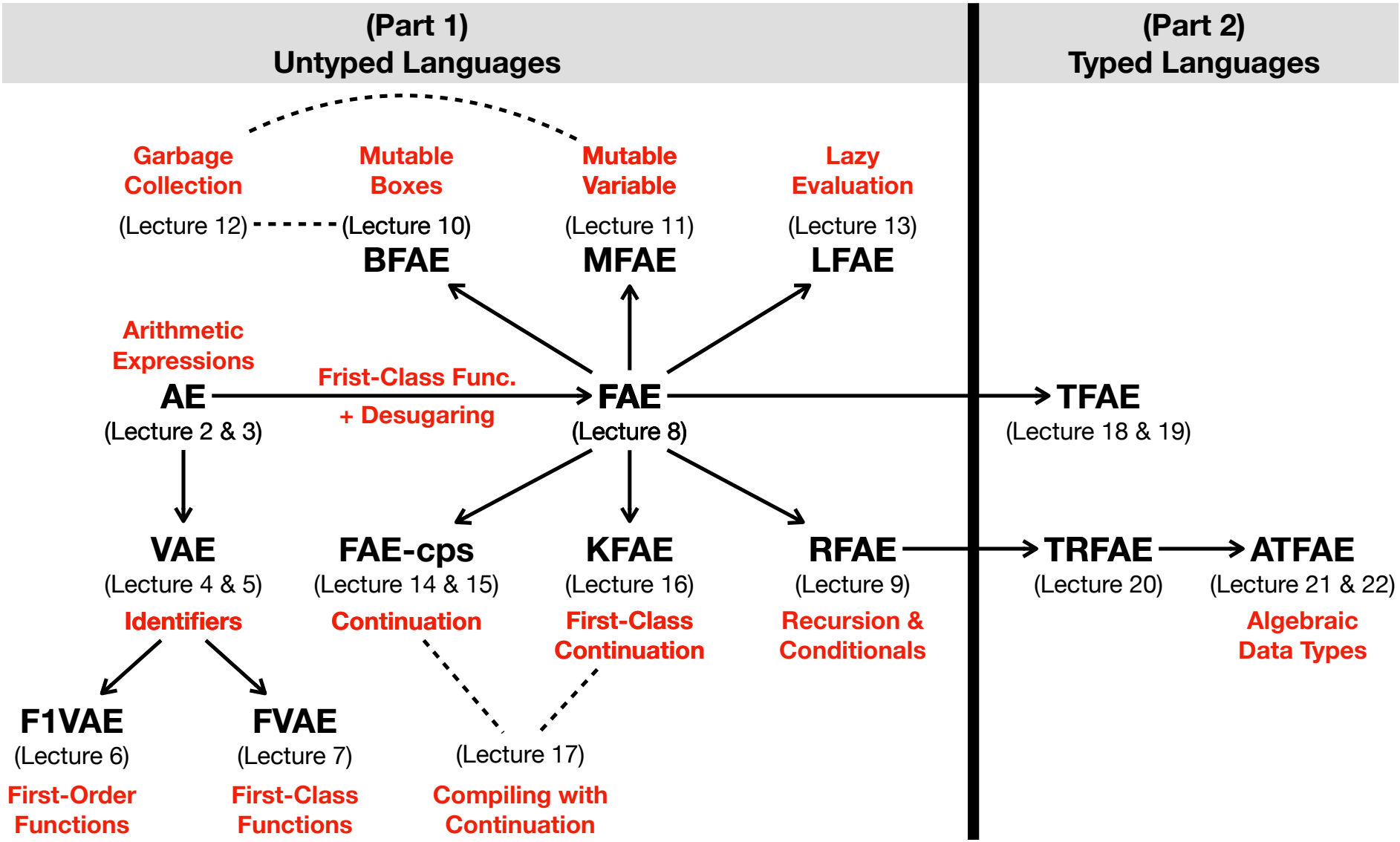
Summary


(Part 1)
Untyped Languages
(Part 2)
Typed Languages
Garbage Collection
(Lecture 12)
Mutable Boxes
(Lecture 10)
Mutable Variable
(Lecture 11)
Lazy Evaluation
(Lecture 13)
BFAE
MFAE
LFAE
Arithmetic Expressions
AE
(Lecture 2 & 3)
Frist-Class Func.
+ Desugaring
FAE
(Lecture 8)
TFAE
(Lecture 18 & 19)
VAE
(Lecture 4 & 5)
Identifiers
FAE-cps
(Lecture 14 & 15)
Continuation
KFAE
(Lecture 16)
First-Class Continuation
RFAE
(Lecture 9)
Recursion & Conditionals
F1VAE
(Lecture 6)
First-Order Functions
FVAE
(Lecture 7)
First-Class Functions
(Lecture 17)
Compiling with Continuation

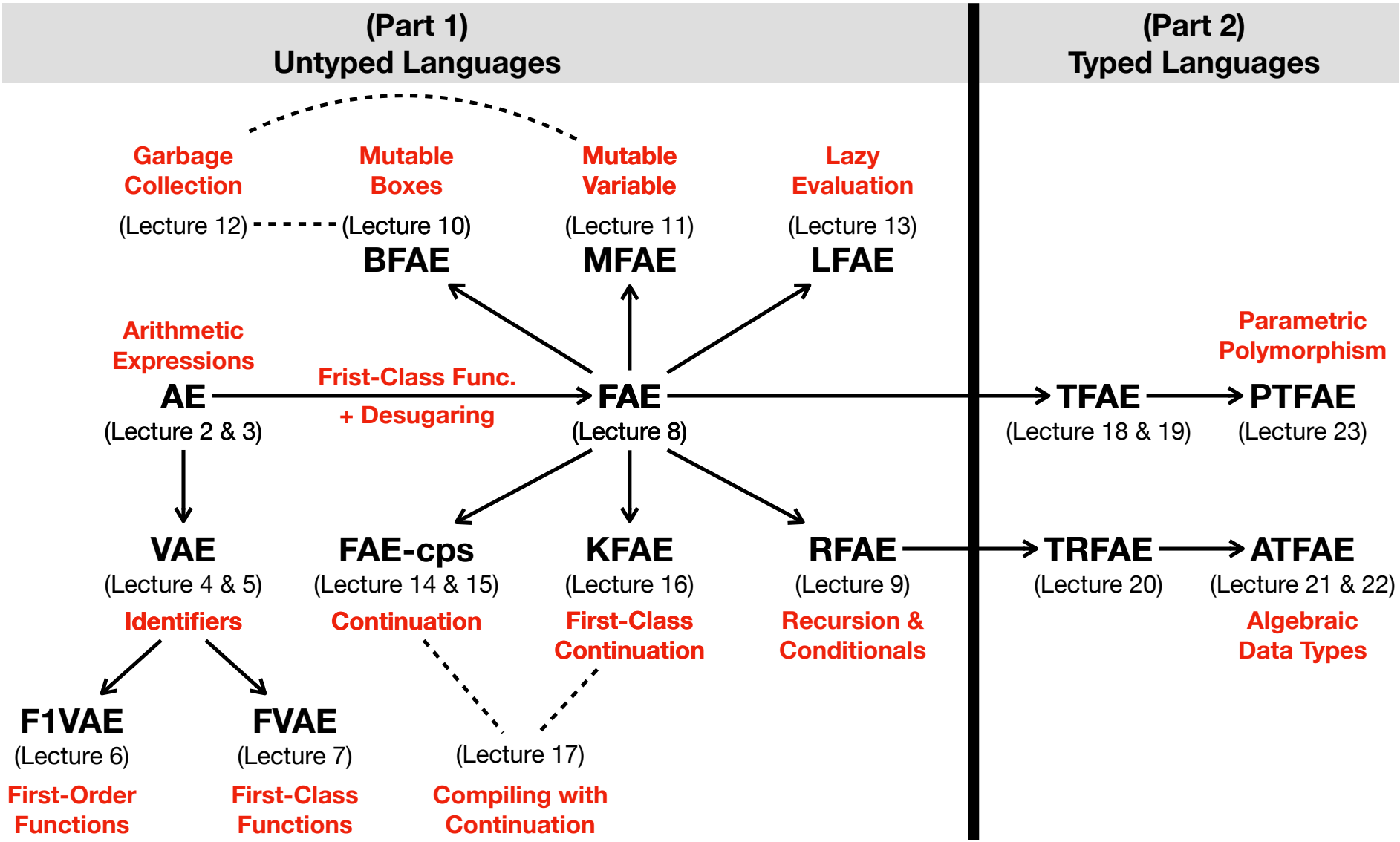
Summary


(Part 1)
Untyped Languages
(Part 2)
Typed Languages
Garbage Collection
(Lecture 12)
Mutable Boxes
(Lecture 10)
Mutable Variable
(Lecture 11)
Lazy Evaluation
(Lecture 13)
BFAE
MFAE
LFAE
Arithmetic Expressions
AE
(Lecture 2 & 3)
Frist-Class Func.
+ Desugaring
FAE
(Lecture 8)
TFAE
(Lecture 18 & 19)
VAE
(Lecture 4 & 5)
Identifiers
FAE-cps
(Lecture 14 & 15)
Continuation
KFAE
(Lecture 16)
First-Class Continuation
RFAE
(Lecture 9)
Recursion & Conditionals
TRFAE
(Lecture 20)
F1VAE
(Lecture 6)
First-Order Functions
FVAE
(Lecture 7)
First-Class Functions
(Lecture 17)
Compiling with Continuation

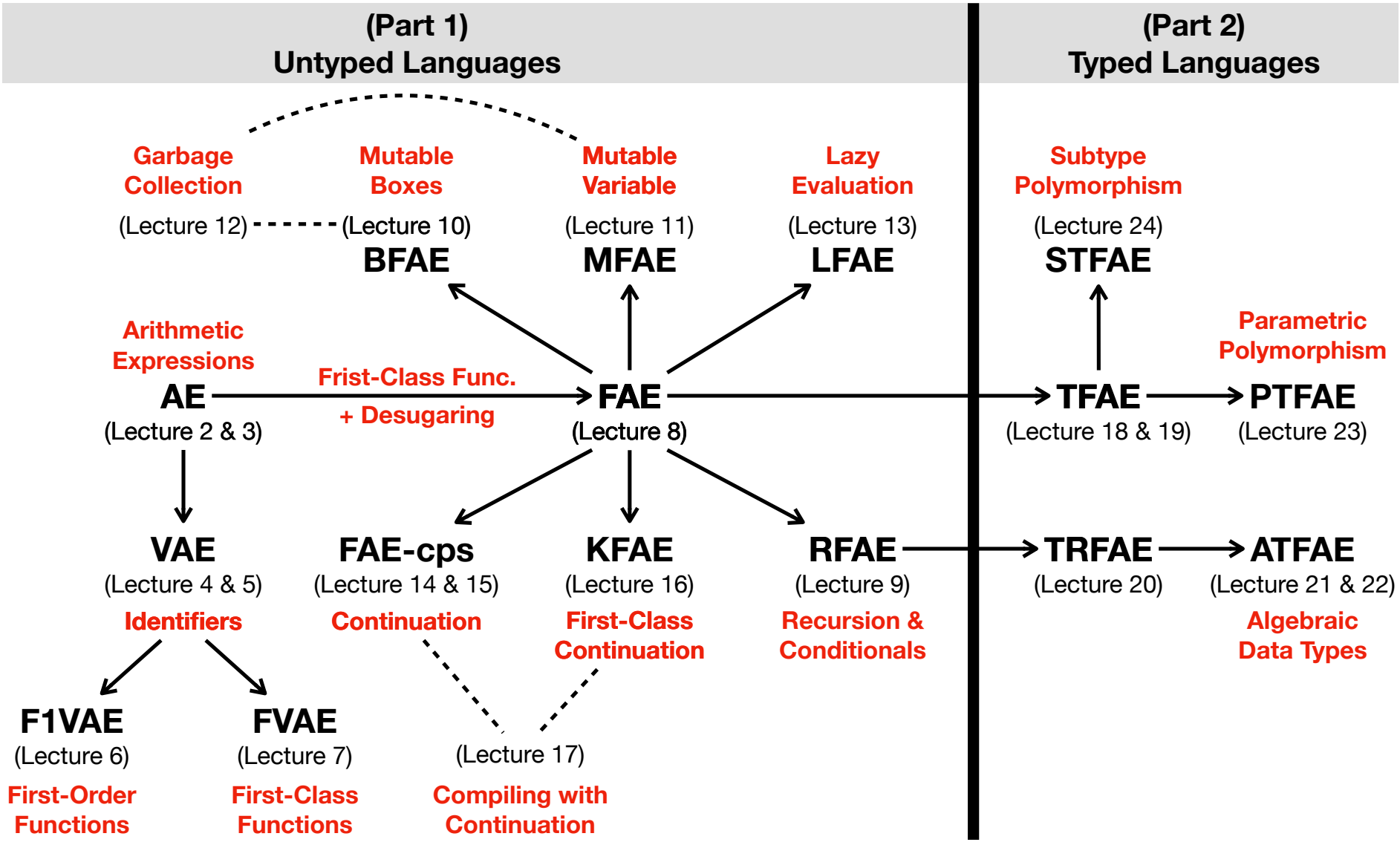
Summary


(Part 1)
Untyped Languages
(Part 2)
Typed Languages
Garbage Collection
(Lecture 12)
Mutable Boxes
(Lecture 10)
BFAE
Mutable Variable
(Lecture 11)
MFAE
Lazy Evaluation
(Lecture 13)
LFAE
Arithmetic Expressions
AE
(Lecture 2 & 3)
Frist-Class Func.
+ Desugaring
FAE
(Lecture 8)
TFAE
(Lecture 18 & 19)
VAE
(Lecture 4 & 5)
Identifiers
FAE-cps
(Lecture 14 & 15)
Continuation
KFAE
(Lecture 16)
First-Class Continuation
RFAE
(Lecture 9)
Recursion & Conditionals
TRFAE
(Lecture 20)
ATFAE
(Lecture 21 & 22)
Algebraic Data Types
F1VAE
(Lecture 6)
First-Order Functions
FVAE
(Lecture 7)
First-Class Functions
(Lecture 17)
Compiling with Continuation

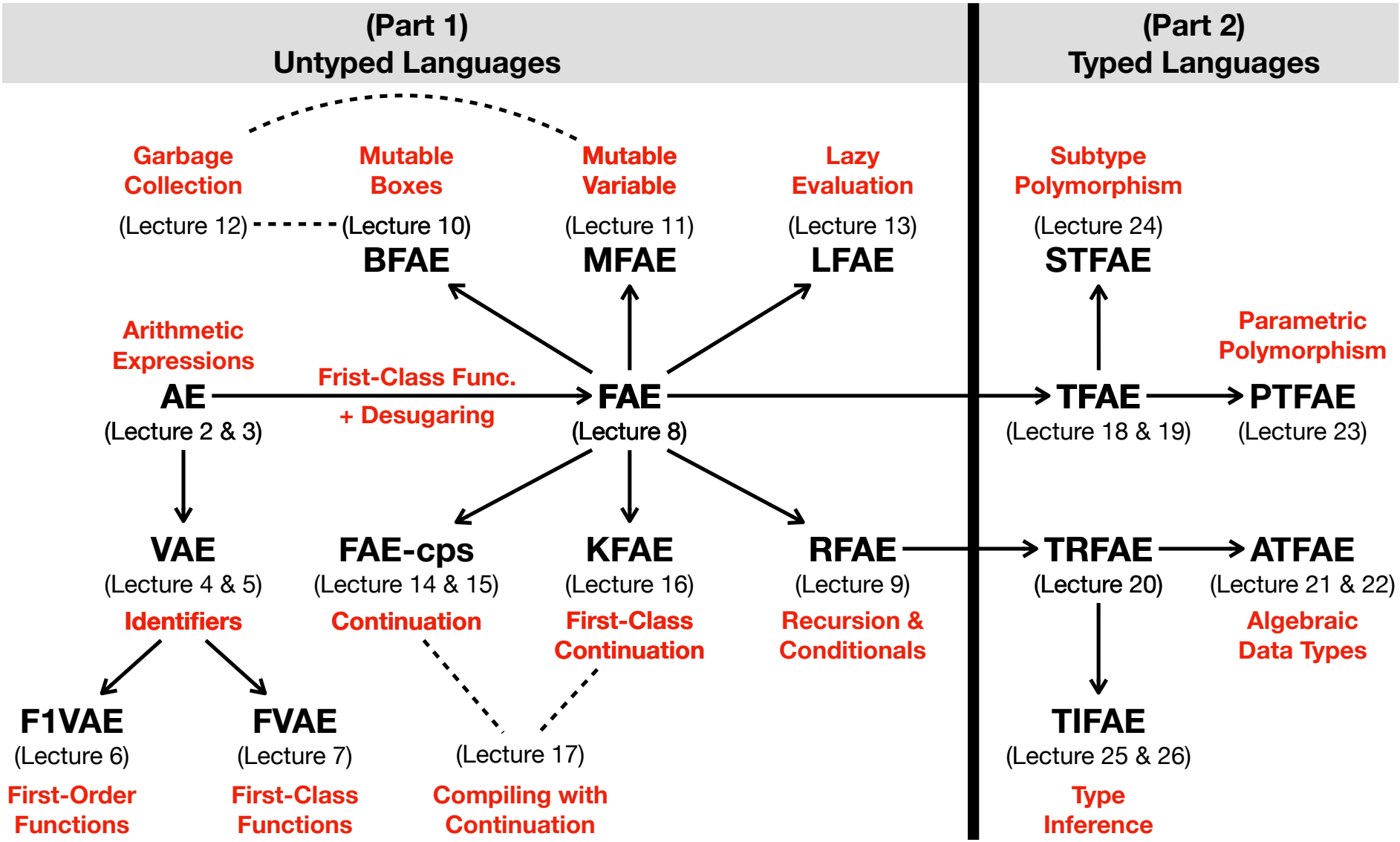
Summary


(Part 1)
Untyped Languages
(Part 2)
Typed Languages
Garbage Collection
(Lecture 12)
Mutable Boxes
(Lecture 10)
Mutable Variable
(Lecture 11)
Lazy Evaluation
(Lecture 13)
BFAE
MFAE
LFAE
Arithmetic Expressions
AE
(Lecture 2 & 3)
Frist-Class Func.
+ Desugaring
FAE
(Lecture 8)
Parametric Polymorphism
TFAE
(Lecture 18 & 19)
PTFAE
(Lecture 23)
VAE
(Lecture 4 & 5)
Identifiers
FAE-cps
(Lecture 14 & 15)
Continuation
KFAE
(Lecture 16)
First-Class Continuation
RFAE
(Lecture 9)
Recursion & Conditionals
TRFAE
(Lecture 20)
ATFAE
(Lecture 21 & 22)
Algebraic Data Types
F1VAE
(Lecture 6)
First-Order Functions
FVAE
(Lecture 7)
First-Class Functions
(Lecture 17)
Compiling with Continuation

Summary


(Part 1)
Untyped Languages
(Part 2)
Typed Languages
Garbage Collection
(Lecture 12)
Mutable Boxes
(Lecture 10)
Mutable Variable
(Lecture 11)
Lazy Evaluation
(Lecture 13)
Subtype Polymorphism
(Lecture 24)
BFAE
MFAE
LFAE
STFAE
Arithmetic Expressions
AE
(Lecture 2 & 3)
Frist-Class Func.
+ Desugaring
FAE
(Lecture 8)
Parametric Polymorphism
TFAE
(Lecture 18 & 19)
PTFAE
(Lecture 23)
VAE
(Lecture 4 & 5)
Identifiers
FAE-cps
(Lecture 14 & 15)
Continuation
KFAE
(Lecture 16)
First-Class Continuation
RFAE
(Lecture 9)
Recursion & Conditionals
TRFAE
(Lecture 20)
ATFAE
(Lecture 21 & 22)
Algebraic Data Types
F1VAE
(Lecture 6)
First-Order Functions
FVAE
(Lecture 7)
First-Class Functions
(Lecture 17)
Compiling with Continuation

Summary


(Part 1)
Untyped Languages
(Part 2)
Typed Languages
Garbage Collection
(Lecture 12)
Mutable Boxes
(Lecture 10)
Mutable Variable
(Lecture 11)
Lazy Evaluation
(Lecture 13)
Subtype Polymorphism
(Lecture 24)
BFAE
MFAE
LFAE
STFAE
Arithmetic Expressions
AE
(Lecture 2 & 3)
Frist-Class Func.
+ Desugaring
FAE
(Lecture 8)
TFAE
(Lecture 18 & 19)
Parametric Polymorphism
PTFAE
(Lecture 23)
VAE
(Lecture 4 & 5)
Identifiers
FAE-cps
(Lecture 14 & 15)
Continuation
KFAE
(Lecture 16)
First-Class Continuation
RFAE
(Lecture 9)
Recursion & Conditionals
TRFAE
(Lecture 20)
ATFAE
(Lecture 21 & 22)
Algebraic Data Types
F1VAE
(Lecture 6)
First-Order Functions
FVAE
(Lecture 7)
First-Class Functions
(Lecture 17)
Compiling with Continuation
TIFAE
(Lecture 25 & 26)
Type Inference

Applications of PL Foundations

PLRG

A deeper understanding of programming languages can help you in:

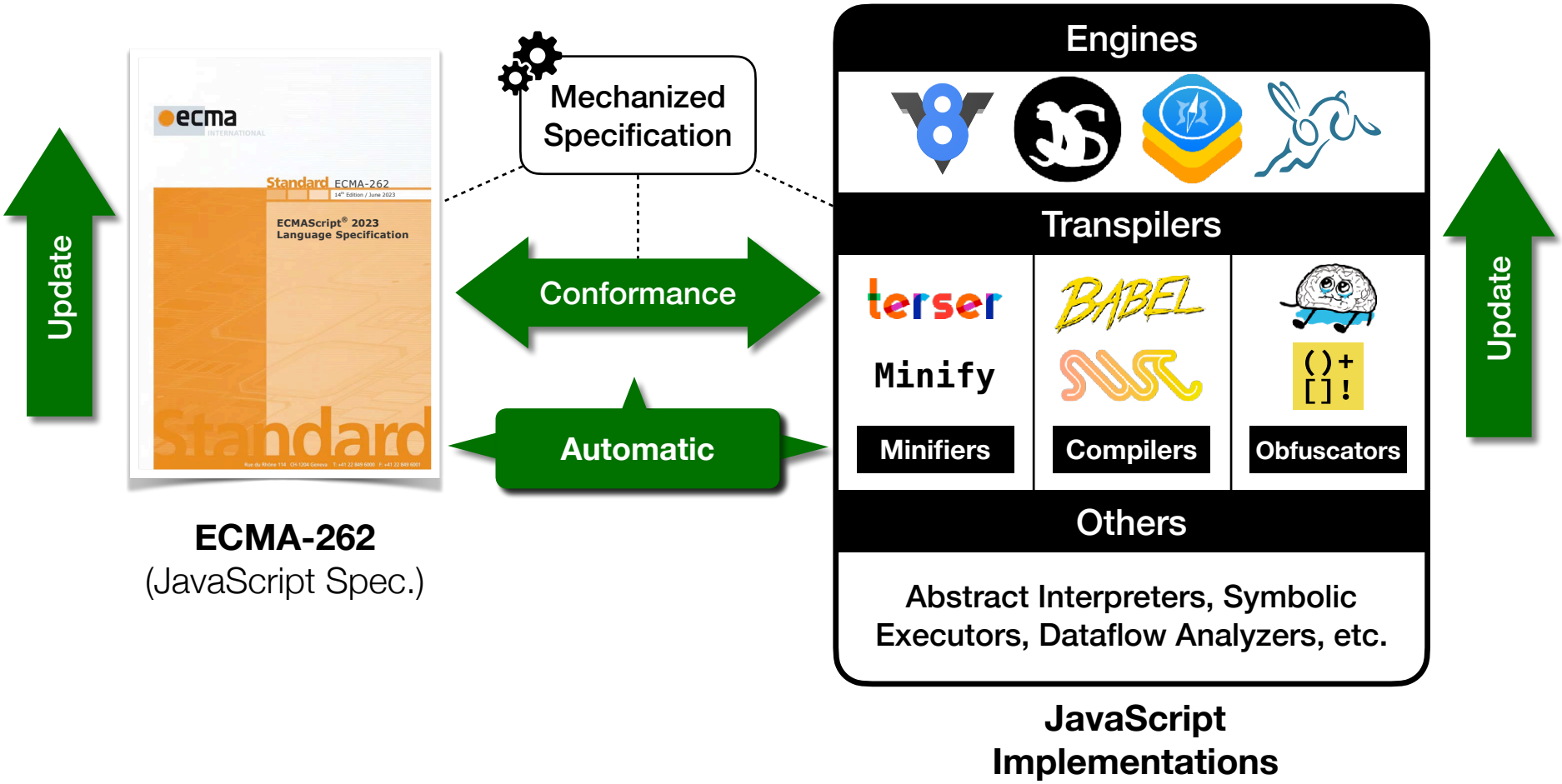
- Static/Dynamic Analysis
- Vulnerability Detection
- Automated Testing
- Program Synthesis
- Explainable AI
- etc.

In the rest of this lecture, I will introduce some of the applications developed in our **programming languages research group (PLRG)**:

https://plrg.korea.ac.kr

Application 1 – Mechanized Specification

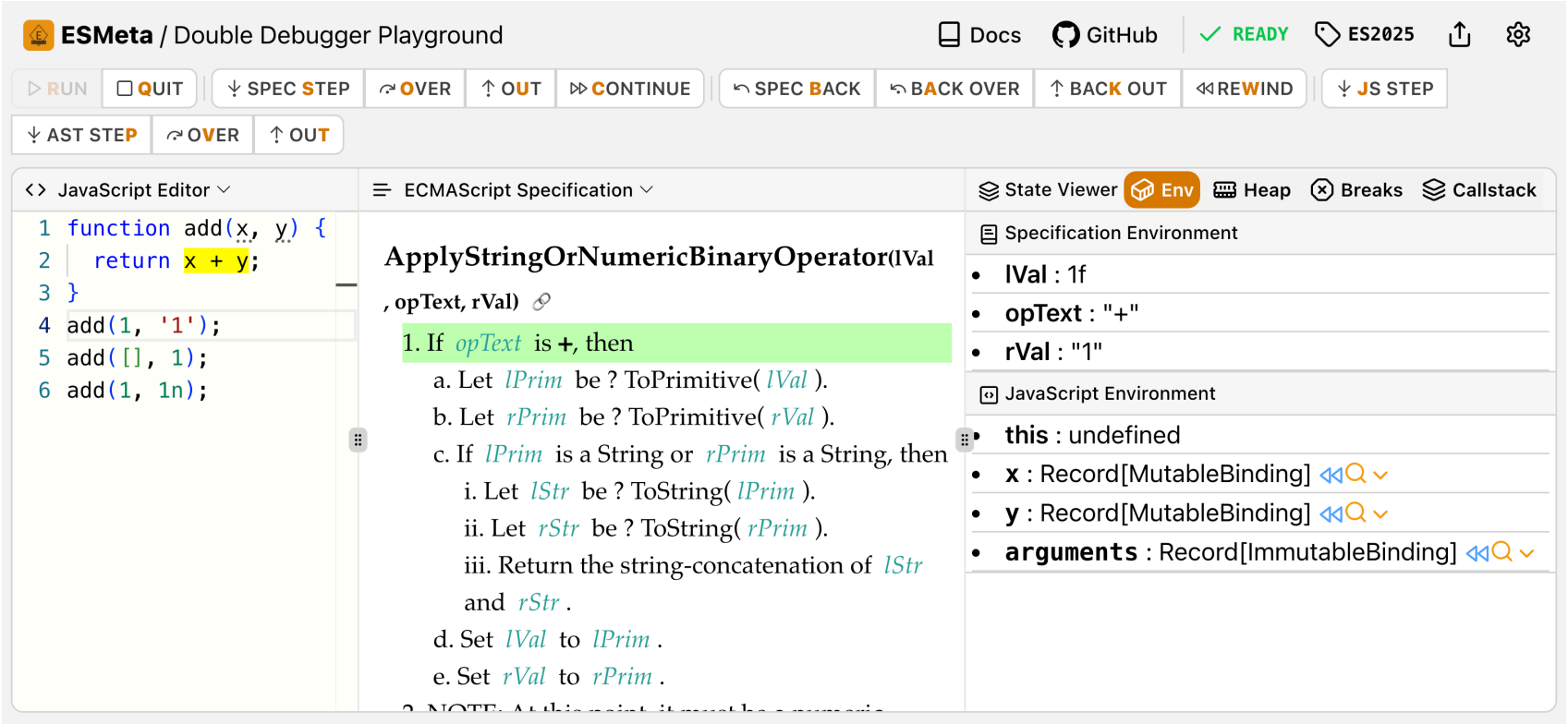
ECMA-262 is the official specification of JavaScript written in English.[1]
ESMeta is a **mechanized** version of ECMA-262 developed by our lab.[2]



[1] https://tc39.es/ecma262/2025/multipage/
[2] https://github.com/es-meta/esmeta

Application 2 – Double Debugger

We can **visualize** the execution of JavaScript programs based on the mechanized specification to help **understand/debug** complex features.[3]

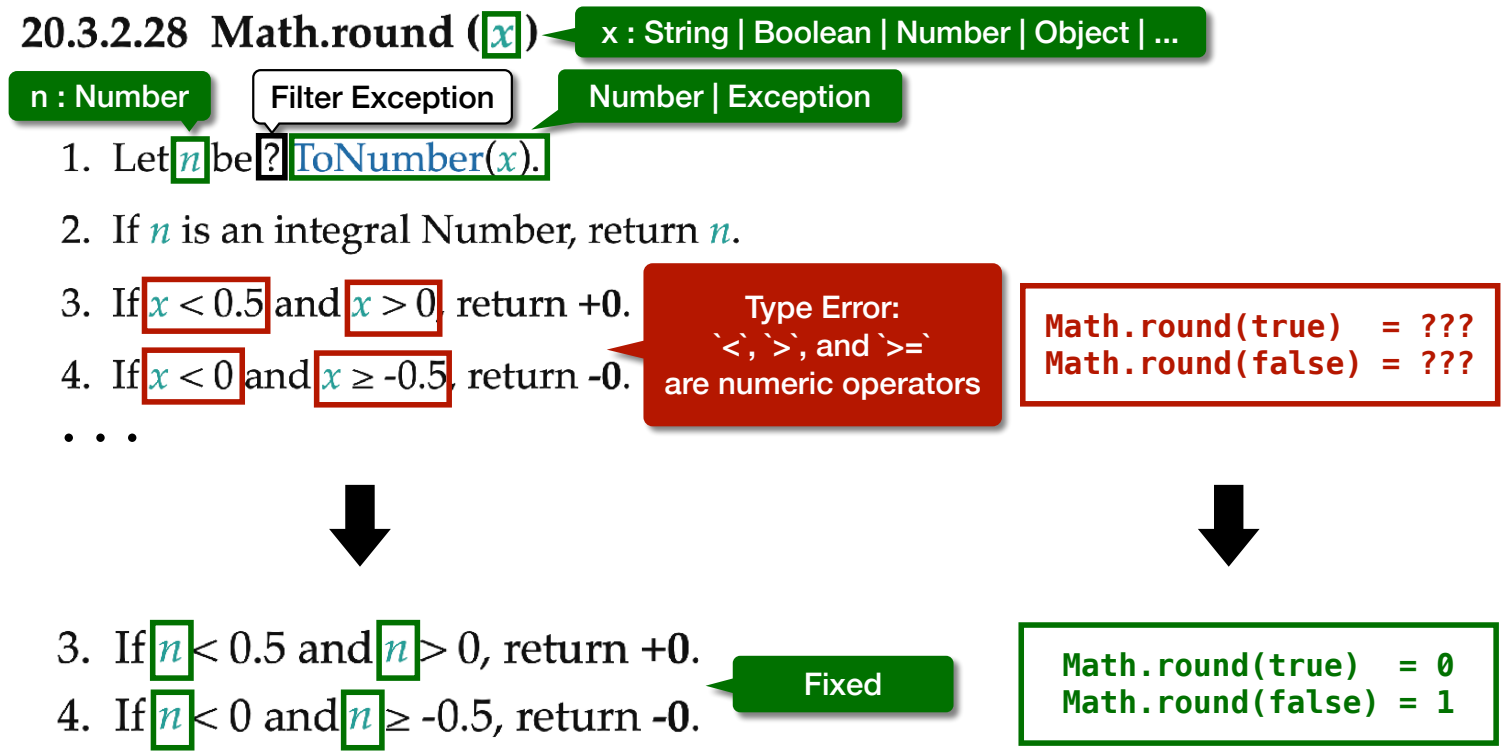


[FSE'25 Demo] M. Choe*, K.Song*, H. Kim, and **J. Park**, "JSSpecVis: A JavaScript Language Specification Visualization Tool"

[3] https://es-meta.github.io/playground/

Application 3 – Type Analysis

We can detect **specification errors** using **type analysis** based on the mechanized specification. It is officially used in language design process.[4]

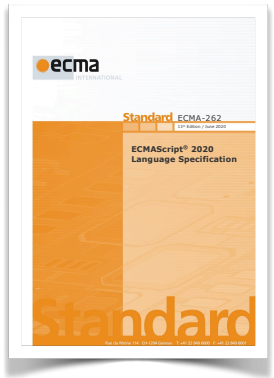


[ASE'21] **J. Park**, S. An, W. Shin, Y. Sim, and S. Ryu, "JSTAR: JavaScript Specification Type Analyzer using Refinement"

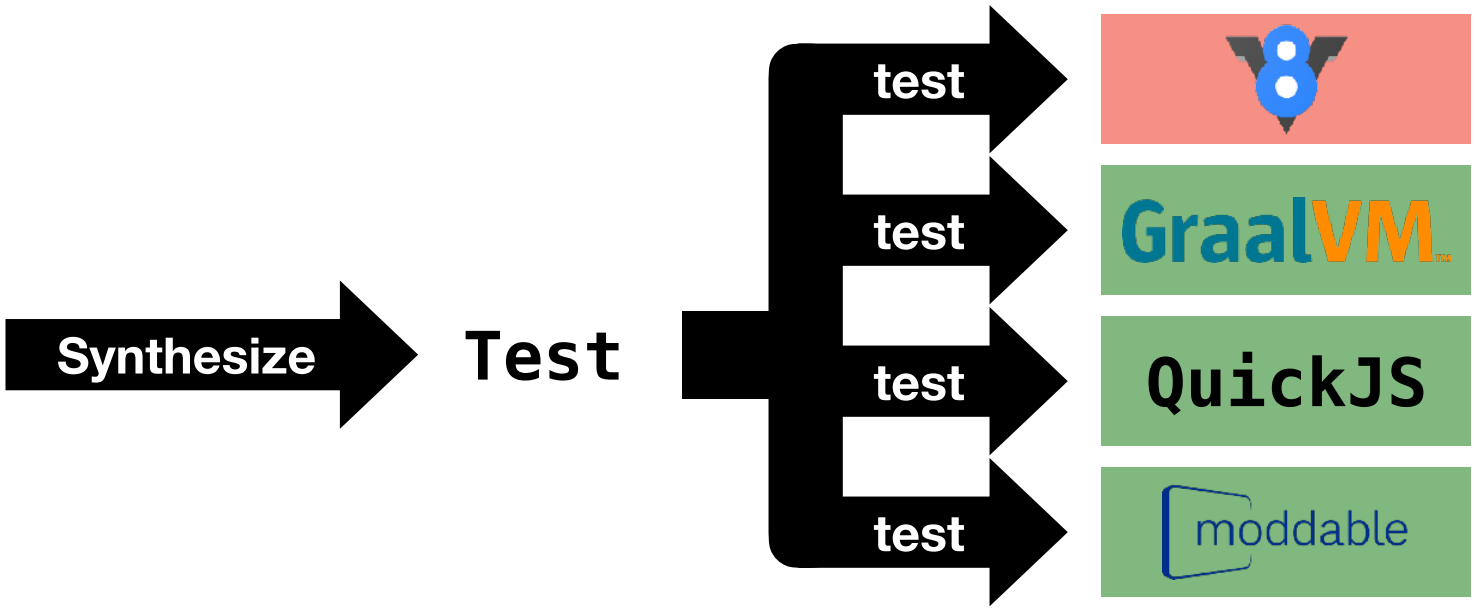
[4] https://github.com/tc39/ecma262/actions/workflows/esmeta-typecheck.yml

Application 4 – Program Synthesis

We can **synthesize JS programs** with assertions from the mechanized specification to **test real-world JS interpreters** (e.g., V8 for Chrome).



ECMA-262
(JavaScript Spec.)



JavaScript Engines

[PLDI'23] **J. Park**, D. Youn, K. Lee, and S. Ryu, "Feature-Sensitive Coverage for Conformance Testing of Programming Language Implementations"

Application 4 – Program Synthesis

PLRG

For example, we found a bug in the SpiderMonkey JavaScript engine (v107.0b4) used in Firefox.[5]

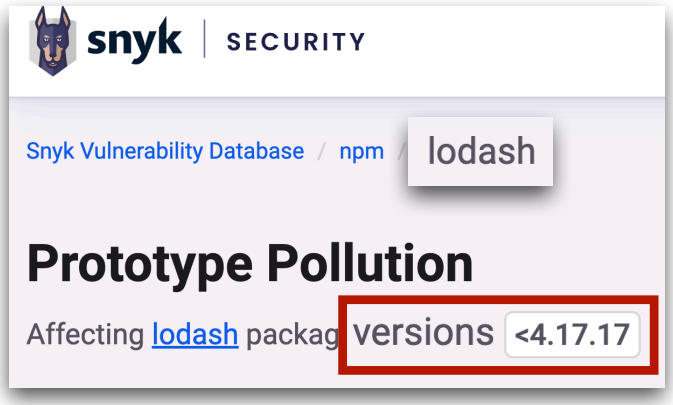
```
var x = (async function ([]) {})();
// Assertions
...
$assert.sameValue(Object.getPrototypeOf(x), Promise.prototype);
$assert.sameValue(Object.isExtensible(x), true);
$assert.notCallable(x);
$assert.notConstructable(x);
...
```

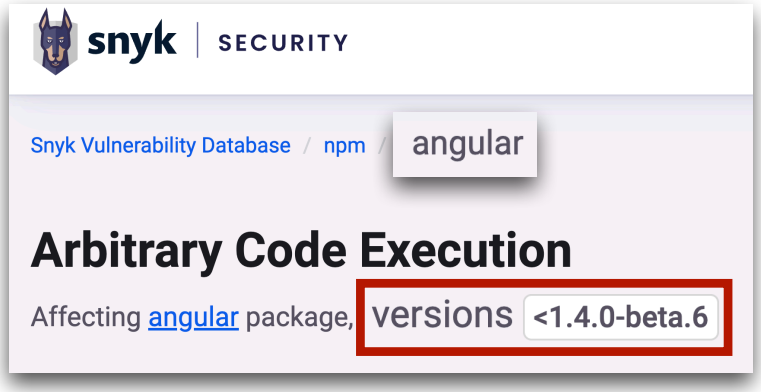
While it should be terminated normally, SpiderMonkey engine throws a run-time `TypeError` when executing this program.

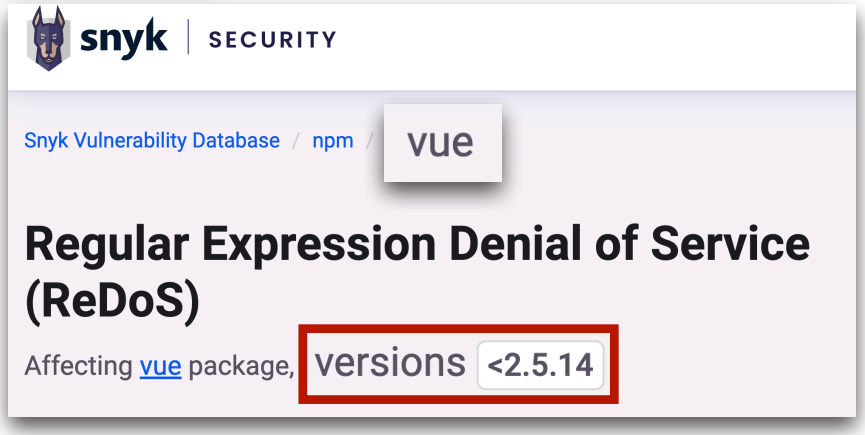
[5] https://bugzilla.mozilla.org/show_bug.cgi?id=1799288

Application 5 – Vulnerability Detection

Detecting vulnerable JS libraries on the web is important for web security.







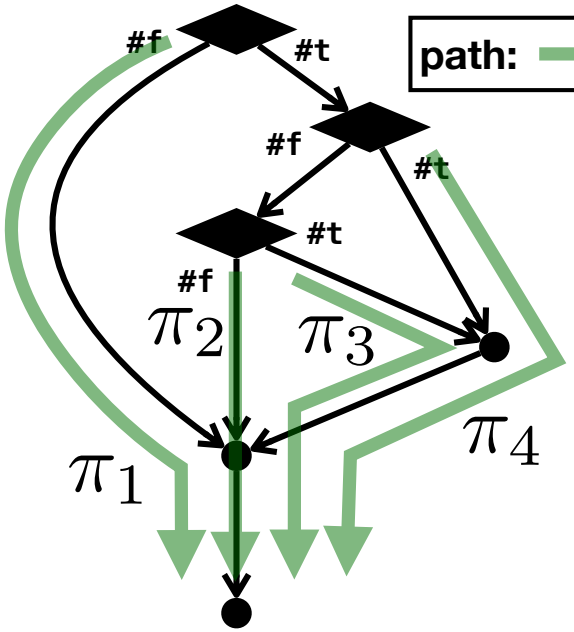
Application 5 – Vulnerability Detection

However, it is challenging to detect JS libraries accurately because:

- Bundlers ***modify*** and ***compress*** **user code** and **libraries** together using diverse **transpilers**
- It makes **difficult to detect libraries** in transpiled code in web applications

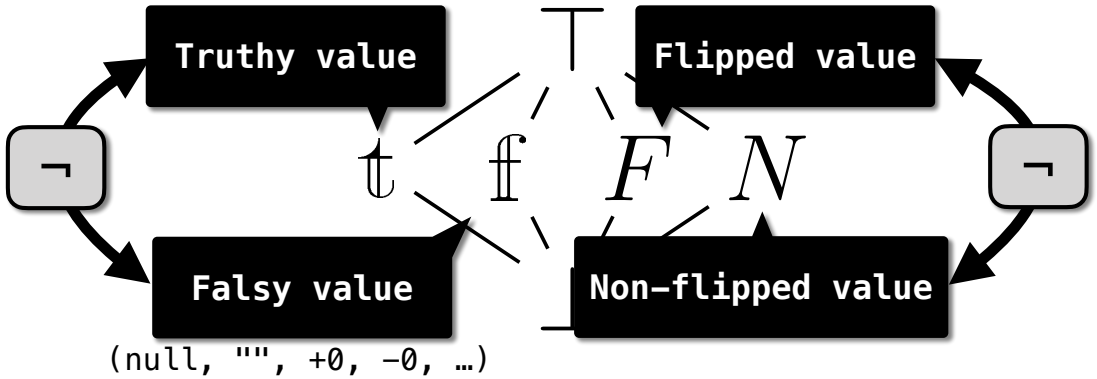
Application 5 – Vulnerability Detection

We can accurately detect vulnerable JS libraries on web using property-order graphs extracted through **static analysis** techniques.



path:

Track truthiness of variables along execution paths!
(Path = Control flow from each branch)



Abstract domain

We are recruiting motivated students who are interested in this topic!

[ASE'25] S. Kim*, S. Park*, and **J. Park**, "Debun: Detecting Bundled JavaScript Libraries on Web using Property-Order Graphs"

Final Exam

- **Date:** 18:30 – 21:00 (150 min.), December 17 (Wed.).
- **Location:** B102, IT & General Education Center (정운오IT교양관)
- **Coverage:** Lectures 14 – 26
- **Format:** closed book and closed notes
  - Fill-in-the-blank questions about the PL concepts.
  - Write the evaluation results of given expressions.
  - Draw derivation trees of given expressions.
  - Define the syntax or semantics of extended language features.
  - Define typing rules for the given language features.
  - etc.
- Note that there is **no class** on **December 15 (Mon.)**.

- I hope you enjoyed the class!

Jihyeok Park
jihyeok_park@korea.ac.kr
https://plrg.korea.ac.kr