

COMP 430 Lecture 1

Kyle Dewey

About Me

- My research:
 - Novel programming language development, in collaboration with JPL
 - Automated test case generation, particularly on testing compilers
- Fourth time teaching this course

About this Class

- Revamped: more flexibility in project features, value of project components negotiated by group, partially flipped classroom for lab time
- See something wrong? Want something improved? Email me about it!
(kyle.dewey@csun.edu)
- I generally operate based on feedback

Bad Feedback

- This guy sucks.
- This class is boring.
- This material is useless.

Good Feedback

- This guy sucks, *I can't read his writing.*
- This class is boring, *it's way too slow.*
- This material is useless, *I don't see how it relates to anything in reality.*
- I can't fix anything if I don't know what's wrong

Motivation

*When will I implement a
compiler?*

*When will I implement a
compiler?*

Probably never.

- *When will I need to reuse my own code?*
- *When will I need to understand how a language works?*
- *When will I need to work on a team?*
- *When will I need to understand why a language was designed a certain way?*

- *When will I need to reuse my own code?*
- *When will I need to understand how a language works?*
- *When will I need to work on a team?*
- *When will I need to understand why a language was designed a certain way?*

Basically always.

Understanding Language Behavior

Understanding Language Behavior

```
int i = 0;  
i = i++ + i++;  
      0   +   1 = 1  
// what is i? (Java)
```

Understanding Language Behavior

```
int i = 0;  
i = i++ + i++;  
// what is i? (Java)  
// what is i? (C)
```

Understanding Language Behavior

```
int i = 0;  
i = i++ + i++;  
// what is i? (Java)  
// what is i? (C)
```

The point: understanding compilers can aid language understanding.

Course Design

- Emphasis on modern compilers
 - Minimal parsing
 - Minimal ultra low-level stuff

Course Design

- Emphasis on modern compilers
 - Minimal parsing
 - Minimal ultra low-level stuff
- It's about writing code

Course Design

- Emphasis on modern compilers
 - Minimal parsing
 - Minimal ultra low-level stuff
- It's about writing code
- It's about teamwork

Choose Your Own Adventure

- Design your own language with certain kinds of features
- Incrementally implement those features
- By the end, you'll have a compiler

Fair Warning

- This is a **lot** of work
- I will try to give you effectively lab time in class, when possible
- As we progress, lectures may get more specialized (depends on you)

Syllabus

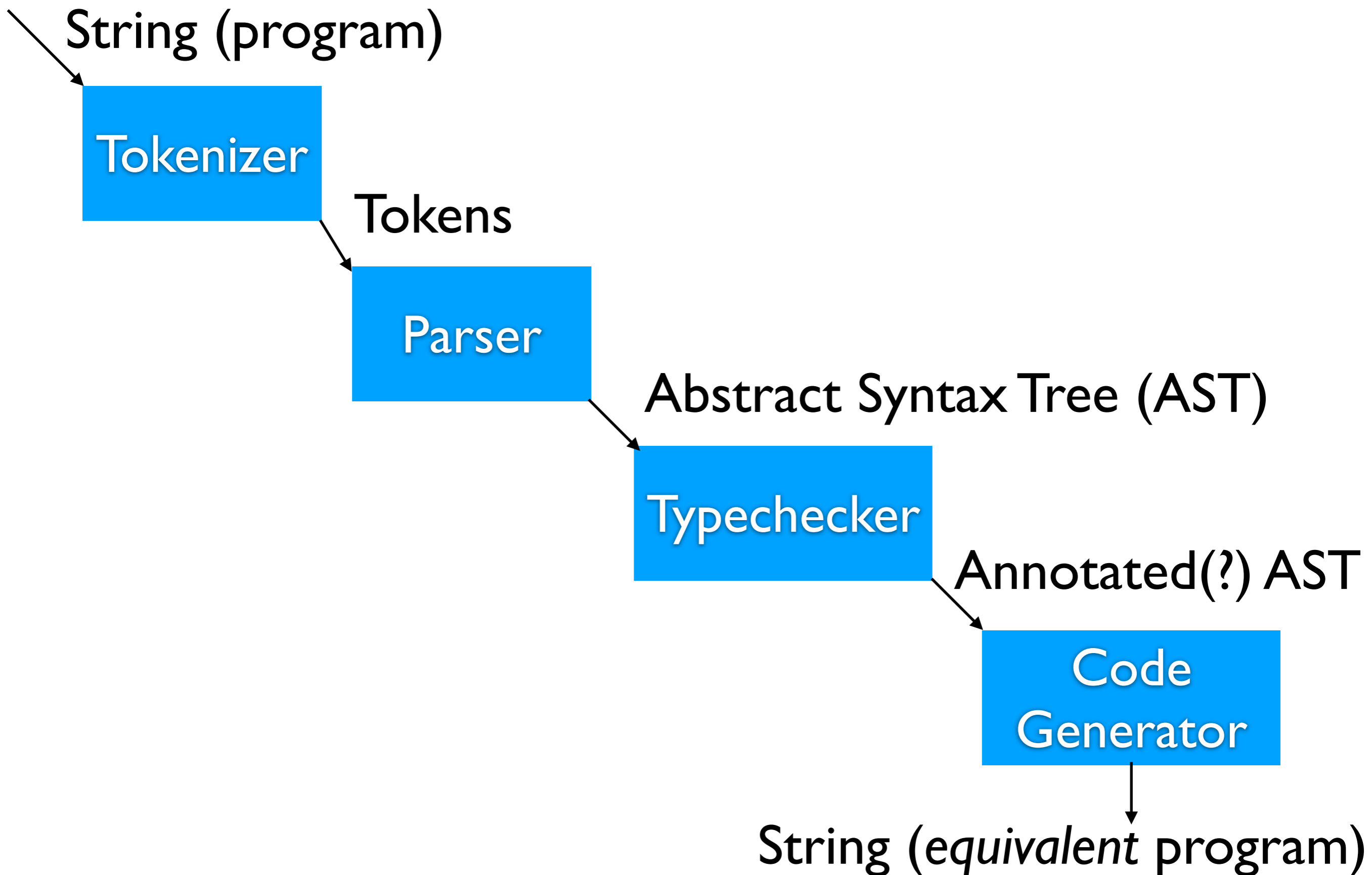
Project Information

Birds-eye View

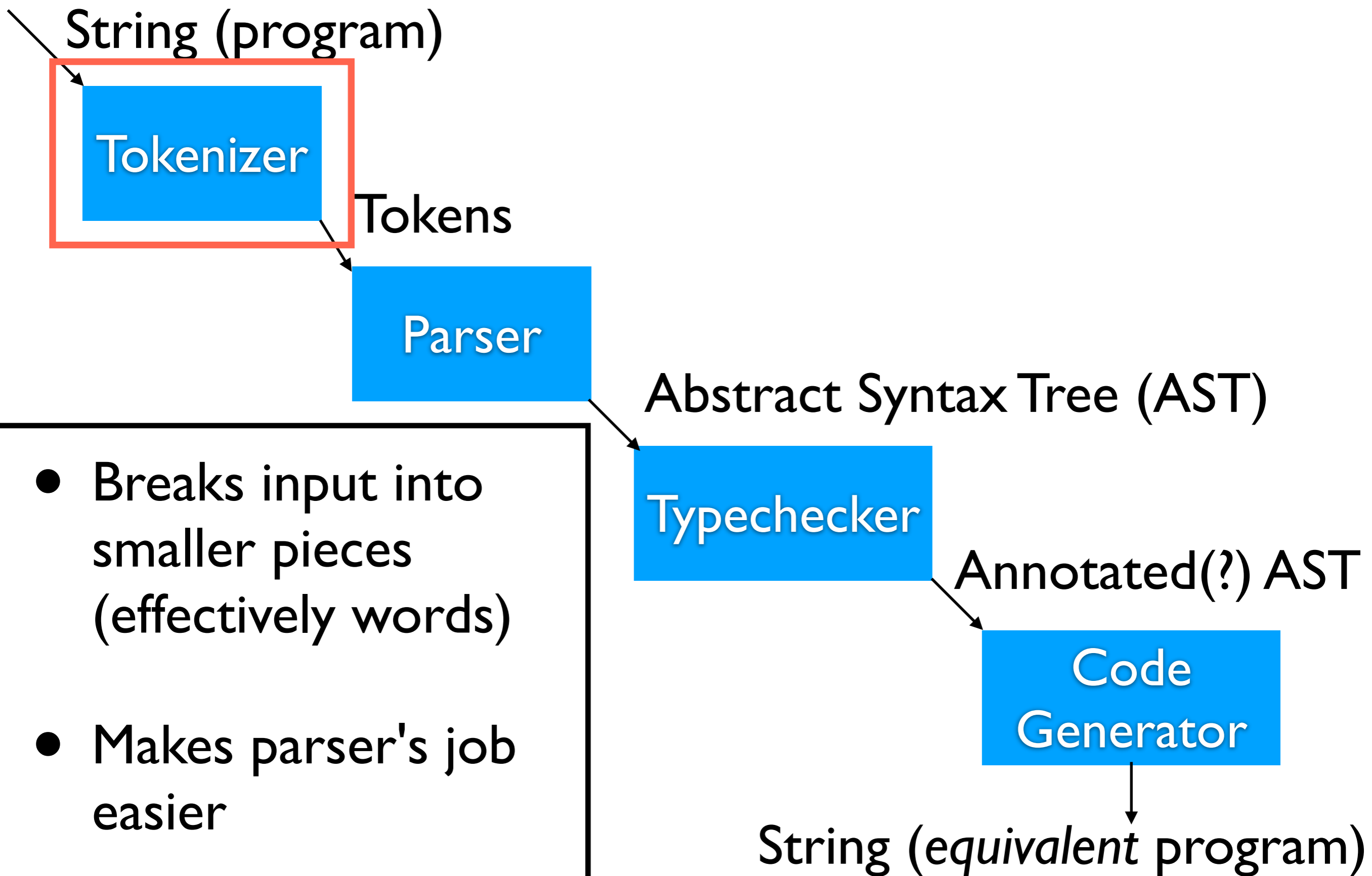
Compiler



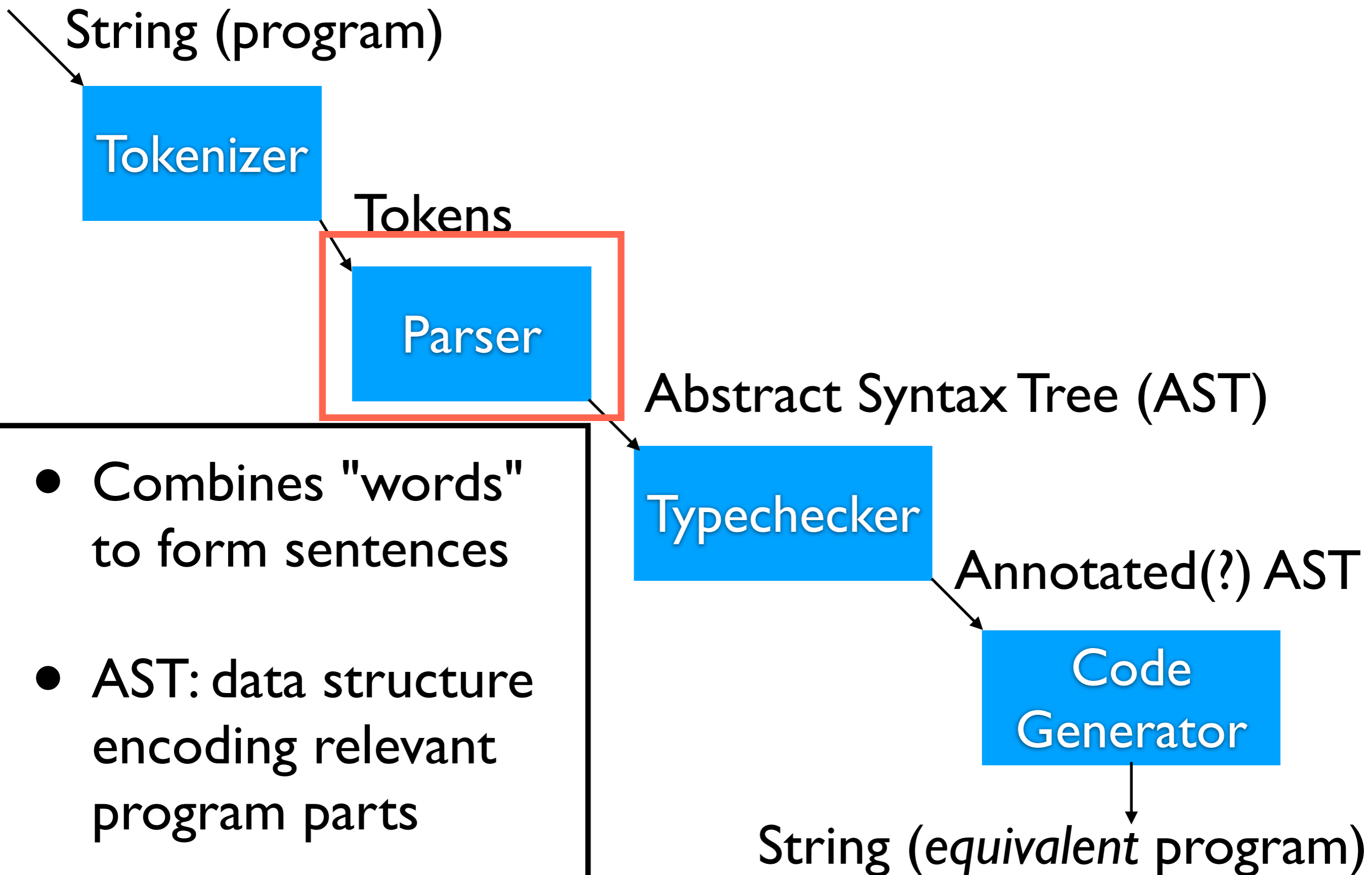
Compiler Architecture



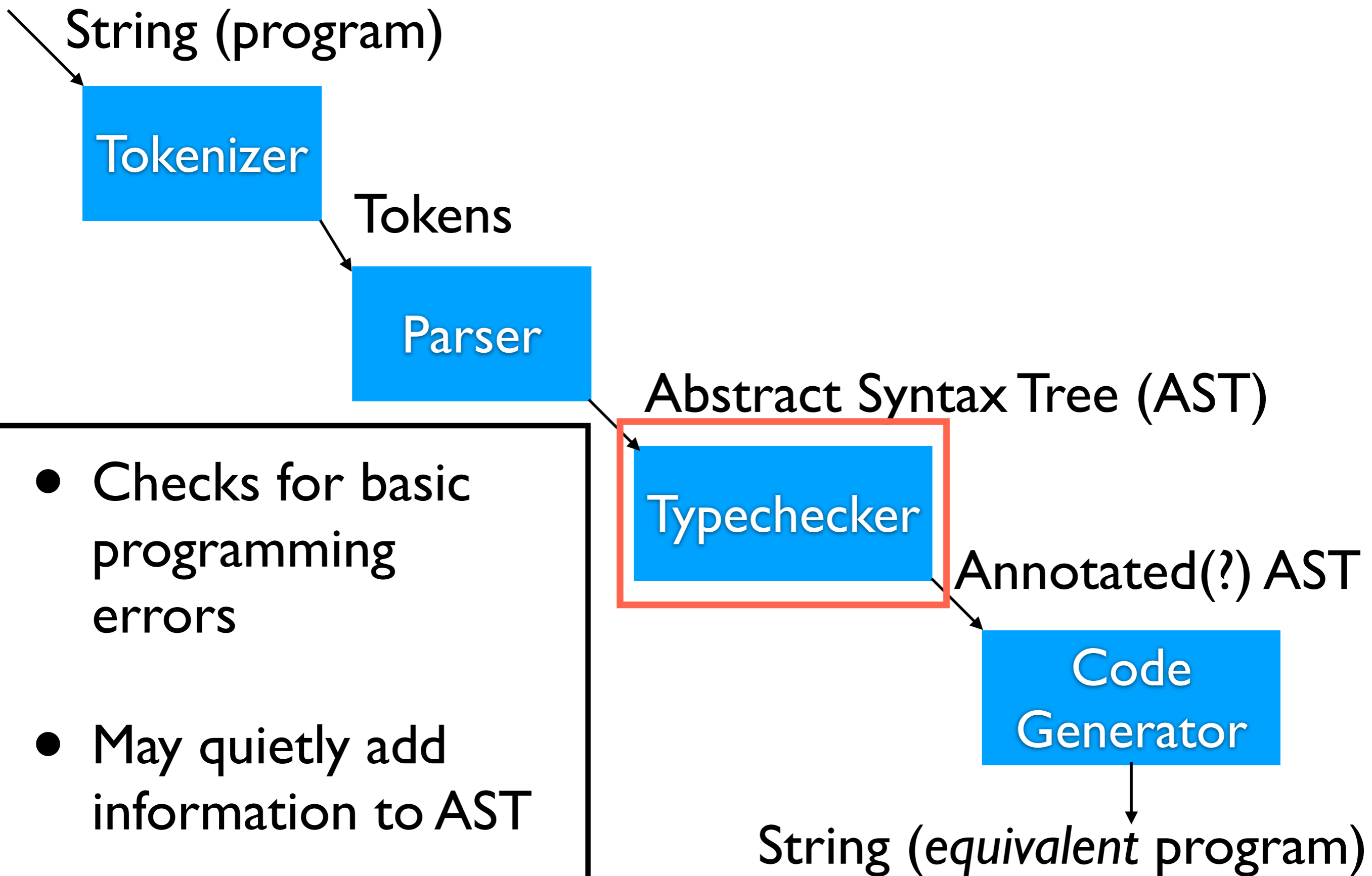
Compiler Architecture



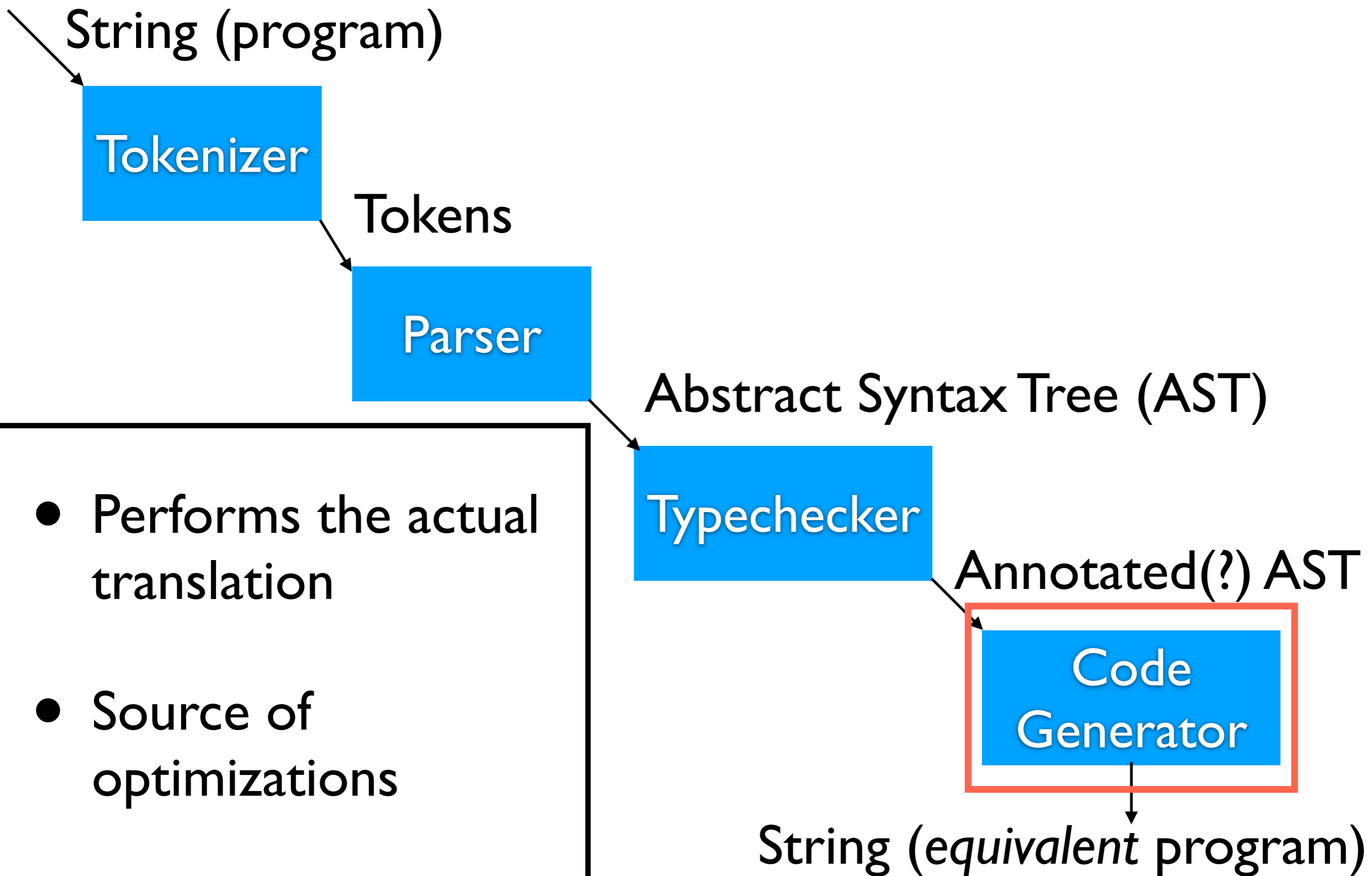
Compiler Architecture



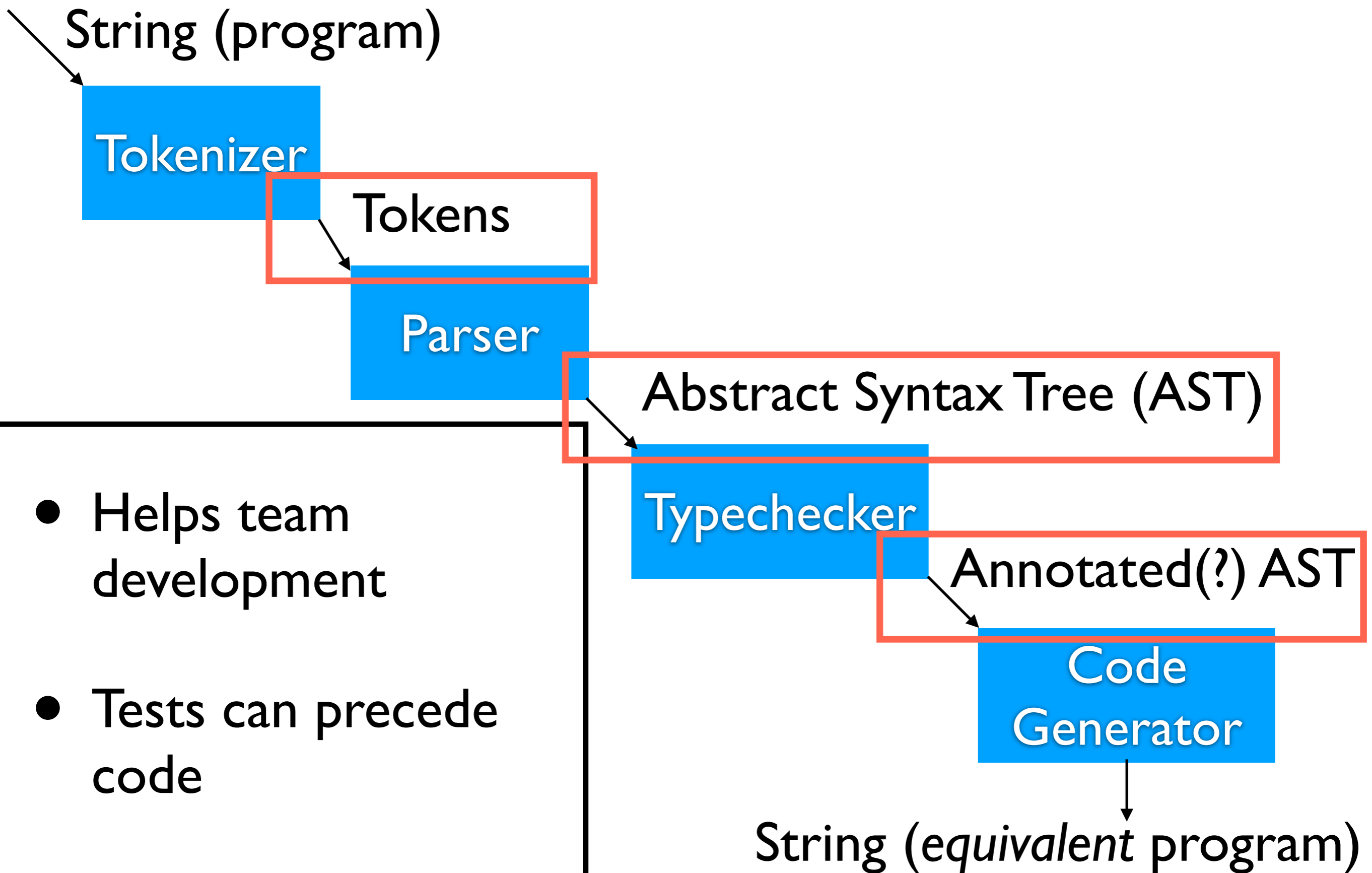
Compiler Architecture



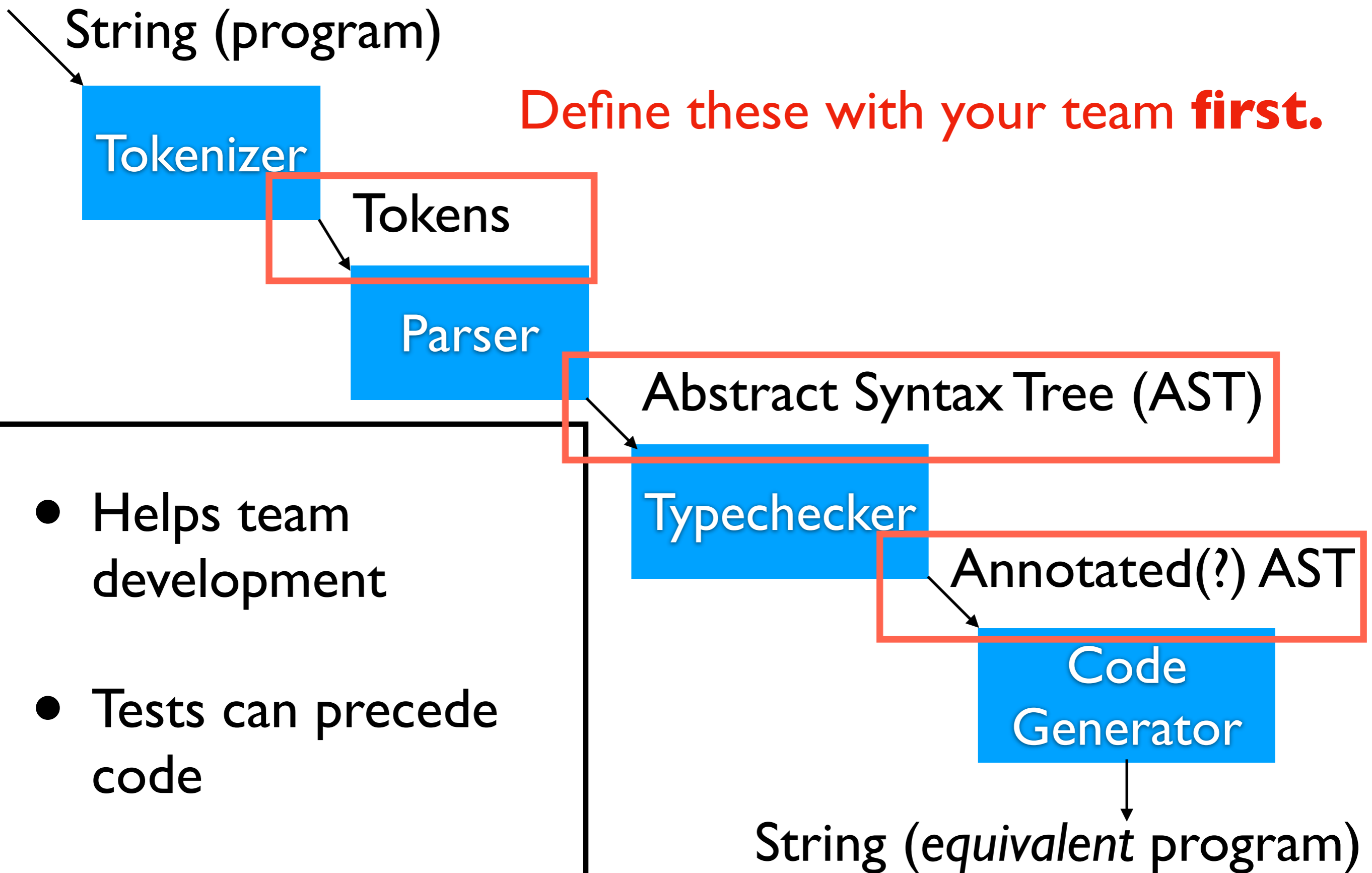
Compiler Architecture



Well-Defined Interfaces



Well-Defined Interfaces



Into the Lexer / Tokenizer

Basic Idea

- Break input into words, called "tokens"
- Every language has its own specific set of tokens

Example

```
if (x < 7) {  
    y = true;  
} else {  
    y = false;  
}
```

Example

```
if (x < 7) {  
    y = true;  
} else {  
    y = false;  
}
```

if	(var("x")	<
int(7))	{	var("y")
=	true	;	}
else	{	var("y")	=
false	;	}	

Tokenization Handout

Livecoded Tokenizer