

# Compiler Construction

## Recap

Rina Zviel-Girshin and Ohad Shacham  
School of Computer Science  
Tel-Aviv University

## Exam

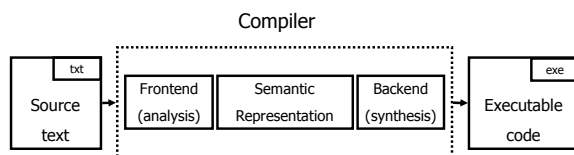
- 31/01/2010 at 9:00
- Look for previous exams on website

- Possible questions

- Parsing
- Extend IC
- Activation records
- ...

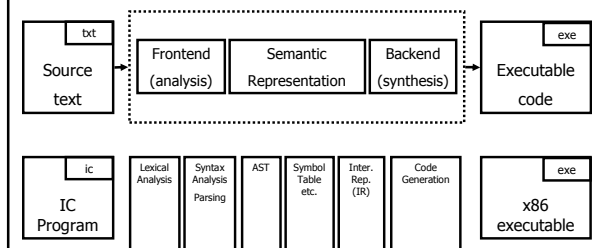
2

## Generic compiler structure



3

## IC compiler Compiler



4

## Lexical Analysis

- Input
  - program text (file)
- Output
  - sequence of tokens
- Read input file
- Identify language keywords
- Count line numbers
- Remove whitespaces
- Report illegal symbols

5

## Scanning

```
// An example program
class Hello {
  boolean state;
  static void main(String[] args) {
    Hello h = new Hello();
    boolean s = h.rise();
    Library.println(s);
    h.setState(false);
  }
  boolean rise() {
    boolean oldState = state;
    state = true;
    return oldState;
  }
  void setState(boolean newState) {
    state = newState;
  }
}
```

Issues in lexical analysis:

- Language changes:
  - New keywords,
  - New operators,
  - New meta-language features, e.g., annotations



```
CLASS, CLASS_ID(Hello), LB, BOOLEAN, ID(state), SEMI ...
```

6

## Tools

- Automatically generate lexical analyzers
- Less error prone
- Many people used these tools
- Easily updated
- Easily to port code between developers
- Lex, Flex, Jlex, JFlex, ...

7

## Parsing

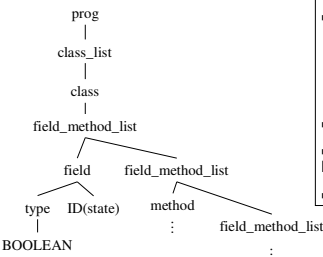
- Input
  - A context free grammar
  - A stream of tokens
- Output
  - An abstract syntax tree or error

8

## Parsing and AST

CLASS, CLASS\_ID(Hello), LB, BOOLEAN, ID(state), SEMI ...

parser uses stream of tokens and generate derivation tree



### Issues in syntax analysis:

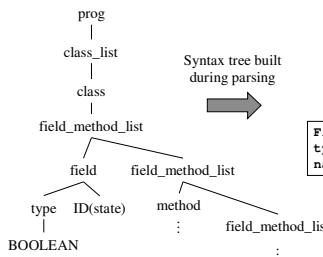
- Grammars: LL(1), LR(0), SLR(1), LALR(1), LR(1)
- Building parsers
  - Transition diagram
  - Parse table
  - Running automaton
- Conflict resolution
- Write LR grammar for a language
- Ambiguity

9

## Parsing and AST

CLASS, CLASS\_ID(Hello), LB, BOOLEAN, ID(state), SEMI ...

parser uses stream of token and generate derivation tree



- Should know difference between derivation tree and AST
- Know how to build AST from input

10

FieldsOrMethods ::=

Field:field FieldsOrMethods:next

```
{: RESULT = next;
  RESULT.addField(field); :}
```

|

Method:method FieldsOrMethods:next

```
{: RESULT = next;
  RESULT.addMethod(method); :}
```

|

/\* empty \*/

```
{: RESULT = new FieldsMethods(); :};
```

11

## Tools

- Automatically generate parsers
- Mostly LR (LALR(1))
- Less error prone
- Many people used these tools
- Easily updated
- Ease the process of switching code between developers
- YACC, Bison, Cup, ...

12

### Questions

- Build an LR grammar for the language
- Is the following grammar in LR(0), SLR(1), LALR(1), LR(1) ?
- Build a parser for the grammar
- Run an input string using your parser

13

### Question

- Is the following grammar LR(0)?

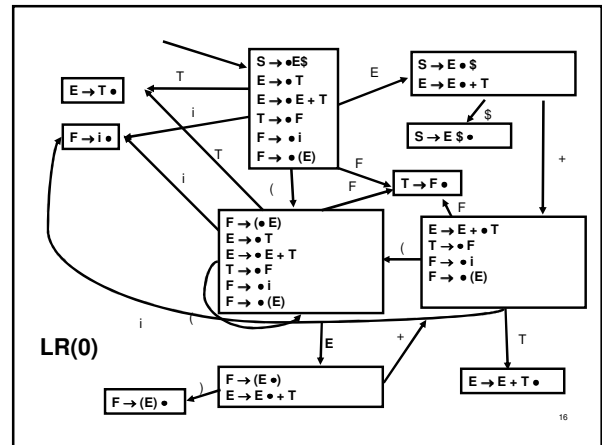
$E \rightarrow E + T$   
 $E \rightarrow T$   
 $T \rightarrow F$   
 $F \rightarrow id$   
 $F \rightarrow (E)$

14

### Answer

- Add a production  $S \rightarrow E\$$
- Construct a finite automaton
- States are set of items  $A \rightarrow \alpha \bullet \beta$

15



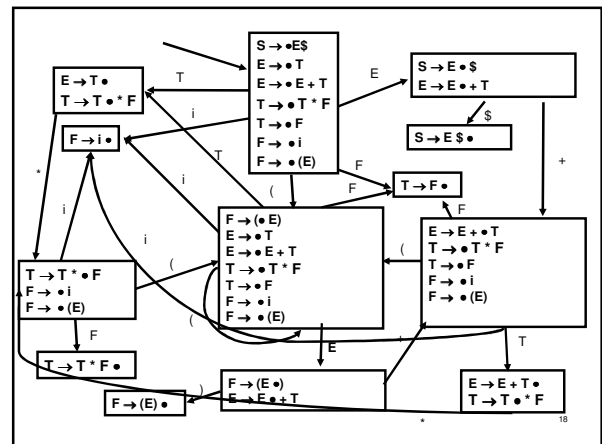
16

### Question

- Is the following grammar LR(0)?

$E \rightarrow E + T$   
 $T \rightarrow T * F$   
 $E \rightarrow T$   
 $T \rightarrow F$   
 $F \rightarrow id$   
 $F \rightarrow (E)$

17



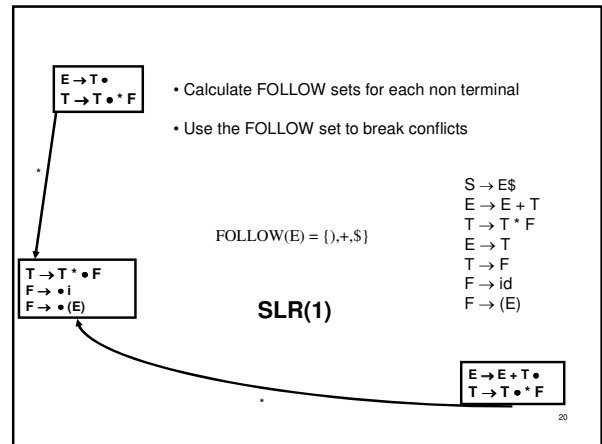
18

### Question

- Is the following grammar SLR(1)?

```

E → E + T
T → T * F
E → T
T → F
F → id
F → (E)
    
```

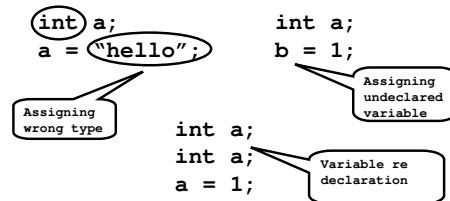


### Semantic analysis

- Context analysis
  - Does break and continue appear only inside while statement?
- Scope analysis
  - Every var is predefined
  - No double definitions
  - Bound var usage to its definition
- Type checking
  - Every expression is well type from the type of its definition
  - Every statement is well type

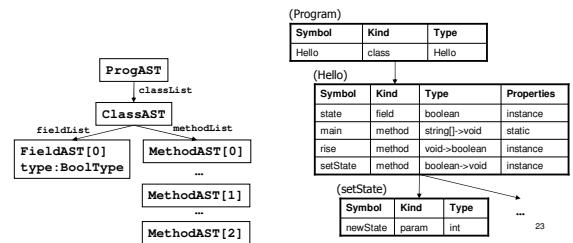
### Semantic analysis motivation

Syntax analysis is not enough

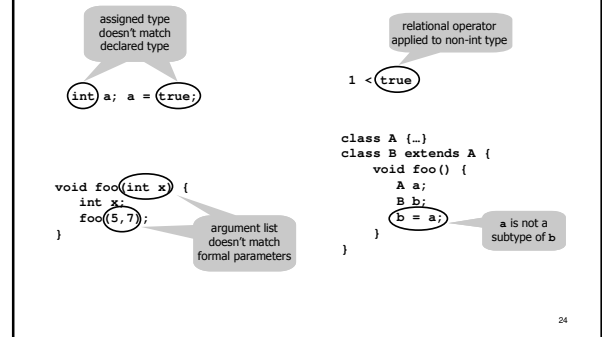


### Semantic analysis

- Representing scopes
- Type-checking
- Semantic checks



### Examples of type errors



## Type rules

$E \vdash \text{true} : \text{bool}$	$E \vdash \text{false} : \text{bool}$	
$E \vdash \text{int-literal} : \text{int}$	$E \vdash \text{string-literal} : \text{string}$	
$E \vdash e1 : \text{int}$	$E \vdash e2 : \text{int}$	$op \in \{ +, -, /, *, \% \}$
$E \vdash e1 \ op \ e2 : \text{int}$		
$E \vdash e1 : \text{int}$	$E \vdash e2 : \text{int}$	$rop \in \{ <=, <, >, >= \}$
$E \vdash e1 \ rop \ e2 : \text{bool}$		
$E \vdash e1 : T$	$E \vdash e2 : T$	$rop \in \{ =, != \}$
$E \vdash e1 \ rop \ e2 : \text{bool}$		

25

## Semantic conditions

- What is checked in compile-time and what is checked in runtime?

Event	C/R
Program execution halts	
Break/continue inside a while statement	
Array index within bound	
In Java the cast statement <b>(A) f</b> is legal	
In Java method o.m(...) is illegal since m is private	

26

## Semantic conditions

- What is checked in compile-time and what is checked in runtime?

Event	C/R
Program execution halts	<b>R</b> (undecidable in general)
Break/continue inside a while statement	<b>C</b>
Array index within bound	<b>R</b> (undecidable in general)
In Java the cast statement <b>(A) f</b> is legal	Depends: if <b>A</b> is sub-type of <b>f</b> then checked during runtime (raising exception), otherwise flagged as an error during compilation
In Java method o.m(...) is illegal since m is private	<b>C</b>

27

## Question – extend IC

- Support Java override annotation *inside comments*
  - // @Override
  - Annotation is written above method to indicate it overrides a method in superclass
- Describe the phases in the compiler affected by the change and the changes themselves

### Legal program

```
class A {
    void rise() {...}
}
class B extends A {
    // @Override
    void rise() {...}
}
```

### Illegal program

```
class A {
    void rise() {...}
}
class B extends A {
    // @Override
    void ris() {...}
}
```

28

## Answer

- The change affects the lexical analysis, syntax analysis and semantic analysis
- Does not affect later phases
  - User semantic condition

29

## Changes to scanner

- Add pattern for @Override inside comment state patterns
- Add Java code to action to comments – instead of not returning any token, we now return a token for the annotation

```
boolean override=false;
%%
<INITIAL> // { override=false; yybegin(comment); }
<comment> @Override { override=true; }
<comment> \n { if (override)
                return new Token(..., override,...)
            }
```

30

## Changes to parser and AST

method → static type name params '{' mbody '}'  
 | type name params '{' mbody '}'  
 | OVERRIDE type name params '{' mbody '}'

Add a Boolean flag to the method AST node to indicate that the method is annotated

31

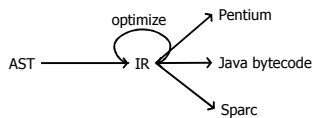
## Changes to semantic analysis

- Suppose we have an override annotation above a method  $m$  in class  $A$
- We check the following semantic conditions
  1. class  $A$  extends a superclass (otherwise it does not make sense to override a method)
  2. Traverse the superclasses of  $A$  by going up the class hierarchy until we find the first method  $m$  and check that it has the same signature as  $A.m$   
 If we fail to find such a method we report an error

32

## Intermediate representation

- Allows language-independent, machine independent optimizations and transformations
- Easy to translate from AST
- Easy to translate to assembly



33

## Translation to IR

- Accept AST and translate functions into lists of instructions
  - Compute offsets for fields and virtual functions
- Dispatch vectors
- Register allocation

34

## Translation to IR

Question: give the method tables for **Rectangle** and **Square**

```
class Shape {
  boolean isShape() {return true;}
  boolean isRectangle() {return false;}
  boolean isSquare() {return false;}
  double surfaceArea() {...}
}
class Rectangle extends Shape {
  double surfaceArea() {...}
  boolean isRectangle() {return true;}
}
class Square extends Rectangle {
  boolean isSquare() {return true;}
}
```

35

## Answer

Method table for rectangle

Shape_isShape
Rectangle_isRectangle
Shape_isSquare
Rectangle_surfaceArea

Method table for square

Shape_isShape
Rectangle_isRectangle
Sqaure_isSqaure
Rectangle_surfaceArea

36