



第二次Lab:

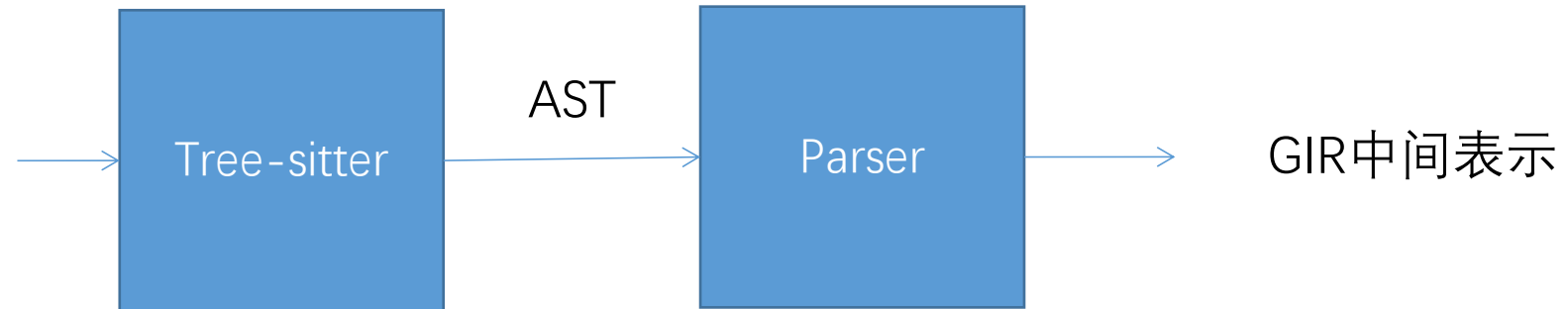
Typescript to GIR



什么是GIR、Parser

- Parser用于将语法解析树转换为统一格式的GIR
 - 输入：代码语法解析树
 - 输出：GIR
- GIR(General Intermediate Representation)是一种中间表示，支持不同的编程语言之间进行转换和优化

各种语言源代码
python
java
c++
typescript





一、环境配置

1.1 操作系统：Linux

1.2 安装实验代码

(1) Python==3.10

(2) 从作业仓库中下载本次实验的代码，仓库地址：
<https://gitee.com/fdu-ssr/compiler2025spring>

(3) 在code文件夹中，执行`pip install -r requirements.txt`。这条命令会自动安装requirements.txt中的依赖项，也可手动安装

二、项目文件介绍



| | | | | |
|--|-----------------|----------------|--------------------|------|
|  docs | 文档说明+grammar.js | 2025/4/7 10:41 | 文件夹 | |
|  scripts | 项目运行脚本 | 2025/4/7 10:41 | 文件夹 | |
|  src | 项目核心代码 | 2025/4/7 10:41 | 文件夹 | |
|  tests | 测试用例+输出结果 | 2025/4/7 10:50 | 文件夹 | |
|  .editorconfig | | 2025/4/7 10:41 | Editor Config 源... | 1 KB |
|  .gitignore | | 2025/4/7 10:41 | Git Ignore 源文件 | 5 KB |
|  .keep | | 2025/4/7 10:41 | KEEP 文件 | 0 KB |
|  .pylintrc | | 2025/4/7 10:41 | PYLINTRC 文件 | 1 KB |
|  LICENSE.txt | | 2025/4/7 10:41 | 文本文档 | 1 KB |
|  requirements.txt | 所需环境库 | 2025/4/7 10:41 | 文本文档 | 1 KB |



二、项目文件介绍

- src/lian/lang/parser
 - xxx_parser.py: 实现对具体语言AST解析, 需要同学们编写实现;
 - common_parser.py: 定义父类Parser, 第189行 parse()为入口函数:
 - 输入:
 - node: AST上的一个节点;
 - statements: 存放IR的中间结果;
 - replacement: 用于存储字符串插值过程中需要被替换的部分及其替换值;
 - 实现:
 - dispatch: 根据ast node类型分发处理函数



二、项目文件介绍

- `src/lian/lang/glang_parser.py`
 - 功能
 - 通过`tree_sitter`库解析指定文件，生成抽象语法树（AST）。
 - 使用具体语言的Parser将AST转换为GLangIR。
 - 提供将多层嵌套的AST节点扁平化为线性结构的方法，便于进一步的处理和分析。
 - 实现：
 - `parse(options, file_path)`:功能: 通过文件路径确定语言类型，并调用相应的解析器解析文件生成GLangIR。
 - 通过`tree_sitter`库解析源代码，生成AST。根据语言类型调用相应的解析器（如`java_parser`或`mir_parser`），将AST转化为GLangIR。使用语言函数（如`tree_sitter_java`）设置解析器的语言环境，并解析文件内容。最终生成的GLang IR语句存储在`glang_statements`列表中。

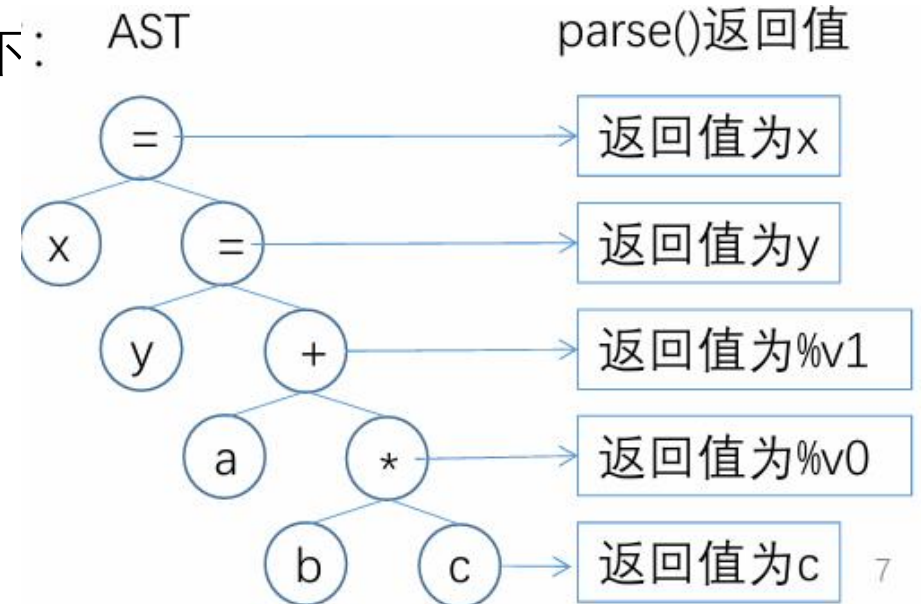


二、项目文件介绍

- 代码解释
 - `common_parser.py - parse()`:
 - 输出
 - 如果该节点为内部节点，则输出为最终临时变量或标识符的名字
 - 如果该节点是叶子节点，则输出其本身值

- 例如表达式: `x = y = a + b * c` 的IR结构如下:

- `%v0 = b * c;`
- `%v1 = a + %v0;`
- `y = %v1;`
- `x = y;`



三、代码运行方式



(1) 运行scripts/lian.sh脚本

\$. /lian.sh <待分析代码文件路径> -l 语言名称

例如:

./lian.sh /python/change.py -l python

(2) 运行结果为:

tests/lian_workspace/dataframe.html

将这个文件在网页中打开, 表格内容为GIR

gir_ir.bundle0
module_symbols

/home/corgi/lianspace/lian-langapi/lian-internal/tests/lian_workspace/gir/gir_ir.bundle0

| | operation | parent stmt id | stmt id | data type | name | unit id | attrs | parameters | body | target | operand |
|----|----------------|----------------|---------|-----------|------------|---------|-------|------------|------|--------|---------|
| 0 | variable_decl | 0 | 10 | | a | 4 | | | | | |
| 1 | method_decl | 0 | 12 | | f1 | 4 | | | 13.0 | | |
| 2 | block_start | 12 | 13 | | | 4 | | | | | |
| 3 | global_stmt | 13 | 14 | | a | 4 | | | | | |
| 4 | variable_decl | 13 | 15 | | b | 4 | | | | | |
| 5 | assign_stmt | 13 | 16 | | | 4 | | | | b | a |
| 6 | assign_stmt | 13 | 17 | | | 4 | | | | a | 4 |
| 7 | block_end | 12 | 13 | | | 4 | | | | | |
| 8 | method_decl | 0 | 19 | | %unit_init | 4 | | | 20.0 | | |
| 9 | block_start | 19 | 20 | | | 4 | | | | | |
| 10 | assign_stmt | 20 | 11 | | | 4 | | | | a | 3 |
| 11 | call_stmt | 20 | 18 | | f1 | 4 | | | | %vv1 | |
| 12 | block_end | 19 | 20 | | | 4 | | | | | |
| 13 | method_decl | 0 | 39 | | append | 7 | | 40.0 | 42.0 | | |
| 14 | block_start | 39 | 40 | | | 7 | | | | | |
| 15 | parameter_decl | 40 | 41 | | e | 7 | | | | | |
| 16 | block_end | 39 | 40 | | | 7 | | | | | |
| 17 | block_start | 39 | 42 | | | 7 | | | | | |
| 18 | array_write | 42 | 43 | | | 7 | | | | | |
| 19 | block_end | 39 | 42 | | | 7 | | | | | |

/home/corgi/lianspace/lian-langapi/lian-internal/tests/lian_workspace/module_symbols

| | module id | symbol name | unit ext | lang | parent module id | symbol type | unit path |
|---|-----------|-------------|----------|--------|------------------|-------------|---|
| 0 | 4 | change | .py | python | 0 | 1 | /home/corgi/lianspace/lian-langapi/lian-internal/tests/lian_workspace/src/cha |
| 1 | 5 | javascript | | | 0 | 12 | /home/corgi/lianspace/lian-langapi/lian-internal/tests/lian_workspace/externs |
| 2 | 6 | python | | | 0 | 12 | /home/corgi/lianspace/lian-langapi/lian-internal/tests/lian_workspace/externs |
| 3 | 7 | pybuiltin | .py | python | 6 | 1 | /home/corgi/lianspace/lian-langapi/lian-internal/tests/lian_workspace/externs |

四、GIR简介



GIR 采用极简设计理念，目前仅包含 78 条基础指令，具体可参见(GIR 说明文档)。这些指令命名遵循直观的语义映射规则，例如 class_decl（类声明）、call_stmt（函数调用）、assign_stmt（赋值语句）。下面是java源代码与对应的GIR：

```
public class VarargsExample {
    public static void printNumbers(int...
numbers) {
        for (int number : numbers) {
            System.out.print(number + " ");
        }
    }
    public static void main(String[] args) {
        printNumbers(1, 2, 3, 4); // 输出: 1 2 3
4        printNumbers(5, 6);    // 输出: 5 6
    }
}
```

| | operation | parent_stmt_id | stmt_id | attrs | fields | nested | supers | methods | name | unit_id |
|----|----------------|----------------|---------|----------------------|--------|--------|--------|---------|----------------|---------|
| 0 | class_decl | 0 | 20 | ['class', 'public'] | | | | 21.0 | VarargsExample | 4 |
| 1 | block_start | 20 | 21 | | | | | | | 4 |
| 2 | method_decl | 21 | 22 | ['public', 'static'] | | | | | printNumbers | 4 |
| 3 | block_start | 22 | 23 | | | | | | | 4 |
| 4 | parameter_decl | 23 | 24 | ['%packed_pos_pmt'] | | | | | numbers | 4 |
| 5 | block_end | 22 | 23 | | | | | | | 4 |
| 6 | block_start | 22 | 25 | | | | | | | 4 |
| 7 | forin_stmt | 25 | 26 | | | | | | number | 4 |
| 8 | block_start | 26 | 27 | | | | | | | 4 |
| 9 | field_read | 27 | 28 | | | | | | | 4 |
| 10 | field_read | 27 | 29 | | | | | | | 4 |
| 11 | assign_stmt | 27 | 30 | | | | | | | 4 |
| 12 | call_stmt | 27 | 31 | | | | | | %vv2 | 4 |
| 13 | block_end | 26 | 27 | | | | | | | 4 |
| 14 | block_end | 22 | 25 | | | | | | | 4 |
| 15 | method_decl | 21 | 32 | ['public', 'static'] | | | | | main | 4 |
| 16 | block_start | 32 | 33 | | | | | | | 4 |
| 17 | parameter_decl | 33 | 34 | ['array'] | | | | | args | 4 |
| 18 | block_end | 32 | 33 | | | | | | | 4 |
| 19 | block_start | 32 | 35 | | | | | | | 4 |
| 20 | call_stmt | 35 | 36 | | | | | | printNumbers | 4 |
| 21 | call_stmt | 35 | 37 | | | | | | printNumbers | 4 |
| 22 | block_end | 32 | 35 | | | | | | | 4 |
| 23 | block_end | 20 | 21 | | | | | | | 4 |



五、任务

- 本学期代码仓库:<https://gitee.com/fdu-ssr/compiler2025spring>
- GIR参考文档<https://docs.qq.com/sheet/DTXBCSIZZS25mQnhQ?tab=urh0bh>
- 本次任务要求:
 - 编写`typescript_parser`, 解析`function_declaration`与`assignment_stmt`
 - `function`无需解析参数, `assign_stmt`只需要考虑左右值为简单变量或常量即可, 无需考虑表达式, 数组等等



六、编写parser

- Parser常用操作:
 - 读取节点文本: `read_node_text(node)`
 - 功能: 获取节点中的源代码文本内容, 通常用于进一步处理或输出。
 - 例如: `value = self.read_node_text(node)`
 - 遍历子节点: `node.named_children`、`find_child_by_type(node, "type")`、`find_child_by_field(node, "field")` 等
 - 功能: 遍历或查找特定类型或字段的子节点, 用于解析复杂结构。
 - 例如: `for child in node.named_children:`

`self.parse(child, statements, replacement)`



六、编写parser

- Parser常用操作：
 - 递归解析节点： `self.parse(node, statements)`
 - 功能：递归调用解析函数，解析子节点，并将结果存储到statements 中。
 - 例如： `self.parse(child, statements)`
 - 字面量处理： `string_literal(node, statements, replacement)`、 `escape_string(value)`、 `handle_hex_string(value)`
 - 功能：解析字符串字面量，并处理转义字符或字符串插值等特殊情况。
 - 例如： `ret = self.read_node_text(node)` `ret = self.escape_string(ret)`
 - 向GIR列表中添加一条新的GIR： `statements.append({"assign_stmt": {"target": shadow_left, "operand": shadow_right}})`