语法树定义

我的语法树使用普通的class来定义，输出成了 Json 格式的文档。

您可以在 <https://github.com/Liu233w/ToyCompiler/tree/master/src/Experiments/EX2/Nodes> 看到所有的语法树节点定义

下面是一个标准的Json语法树格式：

{

"@AstType": "Application", // 语法树节点的类型

"Threads": [ // 每个节点各自的字段

{

"@AstType": "ThreadSpec",

"Identifier": { // 如果是一个 Identifier 或 Decimal 格式的 Token，会直接输出当前Token的定义，包含类型、词素、token的起始和终止位置

"Lexeme": "Thread1",

"TokenType": "identifier",

"TokenBeginIdx": 7,

"TokenEndIdx": 14

},

"Features": {

"@AstType": "PortSpec", // 对于某些可能不存在的节点，比如这里的port在语法定义中可以省略，假如源代码中不存在这一部分，其类型是 NonePortSpec

"Identifier": {

"Lexeme": "AP\_Position\_Input",

"TokenType": "identifier",

"TokenBeginIdx": 27,

"TokenEndIdx": 44

},

"IoType": "in",

"Associations": { // 对于含有多条语句的语句块，不管有没有实际写上花括号和语句，这个AST节点都会存在。假如没有写这条语句，这个节点中的 Associations 会是一个空的列表

"@AstType": "AssociationBlock",

"Associations": [],

"BeginPosition": 89, // 每个AST节点都会包含起始位置和终止位置。假如在源代码中这一部分不存在，起始位置和终止位置会相同。

"EndPosition": 89

},

"PortType": {

"@AstType": "EventDataPort", // PortType 有多种类型，这里每种类型的 @AstType都是不同的

"Reference": {

"@AstType": "Reference",

"PackageName": {

"@AstType": "PackageName", // 假如Reference没有PackageName，这里的类型会是 NonePackageName

"Identifiers": [

{

"Lexeme": "Nav\_Types",

"TokenType": "identifier",

"TokenBeginIdx": 66,

"TokenEndIdx": 75

}

],

"BeginPosition": 66,

"EndPosition": 77

},

"Identifier": {

"Lexeme": "Position\_GPS",

"TokenType": "identifier",

"TokenBeginIdx": 77,

"TokenEndIdx": 89

},

"BeginPosition": 66,

"EndPosition": 89

},

"BeginPosition": 50,

"EndPosition": 89

},

"BeginPosition": 27,

"EndPosition": 90

},

"Flows": {

"@AstType": "FlowPathSpec",

"PreIdentifier": {

"Lexeme": "flow1",

"TokenType": "identifier",

"TokenBeginIdx": 107,

"TokenEndIdx": 112

},

"Identifier": {

"Lexeme": "signal",

"TokenType": "identifier",

"TokenBeginIdx": 124,

"TokenEndIdx": 130

},

"DestIdentifier": {

"Lexeme": "result1",

"TokenType": "identifier",

"TokenBeginIdx": 134,

"TokenEndIdx": 141

},

"BeginPosition": 107,

"EndPosition": 142

},

"Properties": {

"@AstType": "Association",

"PreIdentifier": [],

"Identifier": {

"Lexeme": "dispatch\_protocol",

"TokenType": "identifier",

"TokenBeginIdx": 157,

"TokenEndIdx": 174

},

"Splitter": "=>",

"Constant": false,

"Decimal": {

"Lexeme": "50.0",

"TokenType": "decimal",

"TokenBeginIdx": 185,

"TokenEndIdx": 189

},

"BeginPosition": 157,

"EndPosition": 190

},

"BeginPosition": 0,

"EndPosition": 204

}

]

“BeginPosition”: 0,

“EndPosition”: 204

}