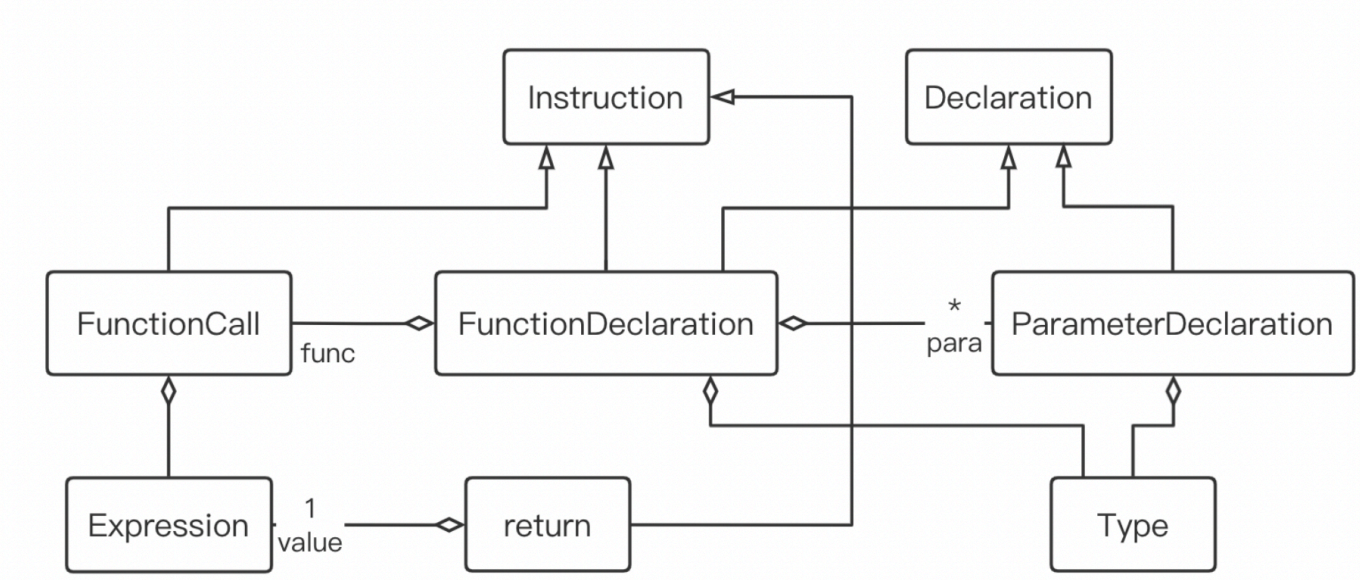
Sémantique et TDL. : Déclarassion et Appel de fonction

1. Déclarassion de founction

1.Proposer des classes pour representer la déclaration de fonction dans l'arbre abstrait.



- $Fun \rightarrow T\ id\ (PO)\ B$

  $\#\{Fun.ast = new\ FunctionDeclaration(T.ast,\ id.txt,\ PO.ast,\ B.ast)\}$
- $I \rightarrow return\ E$:

  $\#\{PO.ast\ = E.ast\}$
- $PO \rightarrow \Lambda$:

  $\#\{PO.ast = new\ ParameterDeclaration()\}$
- $PO \rightarrow LP$:

  $\#\{PO.ast = new\ ParameterDeclaration(LP.ast)\}$
- $LP \rightarrow P,\ SP$:

  $\#\{LP.ast = new\ ListParameterDeclaration(P.ast,\ SP.ast)\}$
- $LP \rightarrow P$:

  $\#\{LP.ast = P.ast\}$
- $P \rightarrow T\ NI$:

  $\#\{P.ast = new\ Declaration(T.ast,\ NI.ast)\}$
- $E \rightarrow id(LEO)$:

  $\#\{E.ast = new\ ExcutionCall(id.txt,\ LEO.ast)\}$
- $LEO \rightarrow \Lambda$:

$\#\{LEO.ast = null\}$

- $LEO \to LE$:

  $\#\{LEO.ast = LE.ast)\}$

3. Function declaration.

```
// Do this first
public boolean resolve(TDS tds) {
  if(!tds.contains(fname)) {
    tds.register(fname);
  }
  SymbolTable n_tds = new Symboltable();
  for (Parameter p:parameter) {
    if(!n_tds.contains(p.getName())) {
      n_tds.register(p);
    }
    else {
      Error(...);
      return false;
    }
  }
  return body.resolve(n_tds);
}
```

```
// Second one.
public boolean fullResolve(TDS tds) {
  boolean ok = result.resolve(tds);
  for (Parameter p:parameter) {
    ok = ok && p.resolve(tds)
  }
}
```

```
public Type getType() {
  List<Type>_para = new ArrayList<Type>();
  for (Parameter p:parameter) {
    _para.add(p.getType());
  }
  return new FunctionType(_para, res_type);
}
```

```
public int allocateMemory(Register reg, int _offset) {
  int offset = 0;
  for(Parameter p:parameter) {
    offset += p.getType().length();
  }
  body.allocateMemory(Register.LB, offset); // Register.LB -> creat a new one
  for(Parameter p:parameter) {
    p.setOffset(offest);
    offset -= p.getType.length();
  }
}
```

5. TAM virtuel machine

```
LOCAL 47
LOCAL 53
CALL (SB) pgcd
SUBR IOUT
HALT
#---------------
pgcd
LOAD (1) -2[LB] ; reading of a Lb->LoadByte
LOAD (1) -1[LB] ; reading of b
SUBR IEQ
JUMPIF(0) else_condition_1
LOAD (1) -2[LB]
RETURN (1) 2
JUMP end_condition_2
#---------------
else_condition_1
LOAD (1) -2[LB]
LOAD (1) -1[LB]
SUBR ILSS
JUMPIF(0) else_condition_3
PUSH 1
LOAD (1) -2[LB]
LOAD (1) -1[LB]
SUBR ISUB
STORE (1) 3[LB]
LOAD (1) -2[LB]
LOAD (1) -3[LB]
CALL (LB) pgcd
RETURN (1) 2
JUMP end_condition_4
#---------------
else_condition_3
PUSH 1
LOAD (1) -2[LB]
LOAD (1) -1[LB]
SUBR ISUB
STORE (1) 3[LB]
```

```
LOAD (1) 3[LB]
LOAD (1) -1[LB]
CALL (1) pgcd
RETURN (1) 2
#---------------
end_condition_4
end_condition_2
LOADL 0
RETURN (1) 1
```

```
fact {
  int fact (int n) {
    if (n = 0) {
      return 1;
    }
    else {
      return n*fact(n-1);
    }
  }
  print fact(5);
}
```

```
LOADL 5
CALL (SB) fact
SUBR IOUT
HALT
#----------
fact
LOAD 1 -1[LB]
LOADL 0
SUBR IEQ
JUMPIF(0) else_condition_1
LOADL 1
RETURN (1) 1
JUMP end_condition_2
#-----------
else_condition_1
LOAD 1 -1[LB]
LOAD 1 -1[LB]
LOADL 1
SUBR ISUB
CALL (LB) fact
SUBR IMUL
RETURN (1)
#-------------
LOADL 0
RETURN (1) 1
```