

# A Translator for OO-Constructs

## From Minijava to $\mu$ -LLVM

---

### Exercise 5

Group 03

Daniele Gadler, Gopal Praveen  
Alka Scaria, Stephen Banin Payin

July 2, 2017

### OVERVIEW

In this report, we describe our group's translator for Object-Oriented constructs from Minijava to *mu*-LLVM. The resulting Translator successfully converts classes into *TypeStructs*, storing fields of base classes and parents' classes through the usage of replication for supporting inheritance. Furthermore, methods are supported through the creation of a Virtual Method Table per class, where methods of the base class, along with the ones inherited from parent classes, are stored. Inheritance and overriding of methods in classes is hence supported through this virtual method table, which is stored as first field of the *TypeStruct* of the corresponding class. New Objects' instantiation, along with fields' access on the left and right-hand side of an assignment operation is also supported. Methods are translated as global procedures.

### FUNCTIONALITIES

- **Class' Fields to StructFieldList in TypeStruct:** A Class is translated into a *TypeStruct*, containing a *StructFieldList*, where all the different fields accessible in a class are stored. In order to allow fields from parent classes to be accessible through children classes, the layout of parents was made compatible with the one of children. Figure 0.1 shows the *TypeStructs* generated from two Java classes A and B, where class B extends class A. These classes contain respectively fields x and y (A) and y (B). Hence, the *TypeStruct* for B contains all fields of A according to the principles of *replication*.



Figure 0.1: Compatible *TypeStructs*' fields layout for supporting inheritance of fields.

- **Class' Methods to Procedures in Virtual Method Table:** All methods contained in a class are translated into procedures, which are stored into a *Virtual Method Table*. A Virtual Method Table is a particular TypeStruct that contains pointers to all the different procedures addressable in a certain class. Furthermore, in order to allow referencing of Virtual Method Tables and the procedures therein contained, these are stored as global variables in the program. Every class declaration has its own Virtual Method Table, which is stored as first field of the class' TypeStruct, as shown Figure 0.2. It is also worth noting that the layout of Virtual Method Tables was designed so as to support inheritance and overriding among classes' methods. For example, if a base class A contains function f() and extends class B, which also contains a function f(), then the method f() in A is considered to be overriding method f() in B and a single procedure reference to f() is stored both in A's and B's Virtual Method Tables.

SomeObject

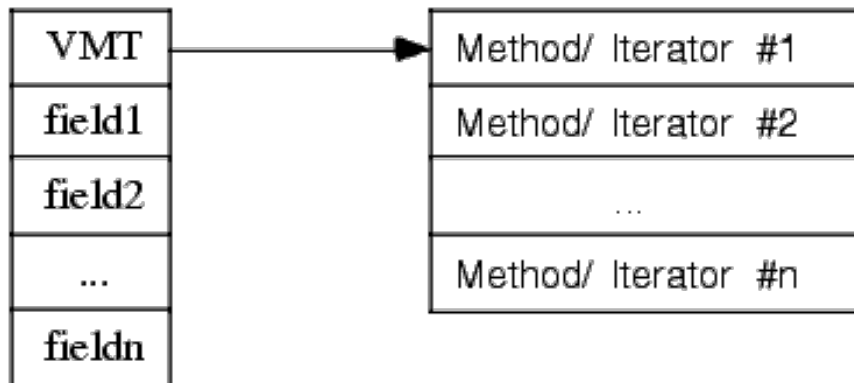


Figure 0.2: Location of the Virtual Method Table inside a Class' Type Struct

- **Object instantiation:** Upon initialization of an object, all fields of the object are initialized to their default value( int -> 0, boolean -> false, int[] and Class -> null): the class' type of the variable the class is being assigned to is checked as well. (For example: A a = new B(), where B extends class A). Should classes on the right-hand and the left-hand side not have the same type, then a Bitcast operation takes place and the left-hand side's type is converted to the right-hand side's type.
- **Field Access:** Should a Field Access take place on the left or right-hand side of an expression, then respectively a value or a reference to the value store inside the field being accessed is returned. Field accessing is done by retrieving the Object's address on the heap and shifting the pointer's offset by an amount equal to the index of the field (starting from index 1, as the Virtual Method Table is stored at index 0). Since fields are stored together with their class name in the class' TypeStruct, then their name is unique (EX: class A has field x -> Field has name A\_x 0.1). Consequently, lookup of the correct field is performed by checking through the following string:

Field\_Name = ClassName + "\_" + FieldName