33: Optimizations for Object-Oriented Languages

Viliam Kasala (xkasal01), Jirka Lang (xlangj00)

The goal of this presentation is to describe basic **optimizations** for Object-Oriented languages such as object layout, method invocation, devirtualization and escape analysis.

In the first part of the presentation, we will introduce the memory layout of an object for ***single*** and ***multiple inheritance****,* and how this layout supports dynamic dispatch. Dynamic dispatch is the process of selecting which implementation of polymorphic operation to call at runtime. The purpose of dynamic dispatch is to support cases, where appropriate implementation of a polymorphic operation can not be determined at compile time, because it depends on the runtime type. For single inheritance there will be presented technique ***using virtual method table for dispatch*** and for multiple inheritance, we will take a look at ***embedding superclasses technique***. Virtual method table, sometimes also called as dispatch table or vtable, is mechanism used in a programming language to implement dynamic dispatch. We will also take a look at ***Bidirectional object layout*** and example of ***simple Java class layout and its method invocation***. ***Bidirectional object layout*** can give us less indirection and smaller memory usage.

In the second part of the presentation, we will speak about **fast-type inclusion tests**, and related algorithms such as relative numbering or hierarchical encoding. Then we will take a look at **devirtualization**, the technique for reduce overhead of virtual method invocations. Different ways how to analyze program, and perform devirtualization will be discussed. Then we will present **escape analysis**, that is used to determine object lifetime in order to space allocation. There will be introduced three object states, to describe whether object escapes from a method and from a thread.