1. Languages, Abstract Machines and Execution models

Slides: 03 Languages, Abstract machines, and Execution models

Programming Languages

A programming language is defined by:

- syntax (the form of the program)
- semantics (meaning of the program): how is expected to behave
- pragmatics (conventions and guidelines): how is intended to be used in practise.

Paradigms: styles of programming.

- Imperative: variables, commands, procedures.
- Object-oriented: objects, methods, classes.
- Concurrent: processes, communication.
- Functional: values, expressions, functions.
- Logic: assertions, relations.

Implementation of a PL

Every language L implicitly defines an Abstract Machine ML having L as machine language. Implementing ML on an existing host machine M0 (via compilation, interpretation or both) makes programs written in L executable.

Abstract Machines

An Abstract Machine ML for L is a collection of data structures and algorithms which can perform the storage and execution of programs written in L.

Java Virtual Machine

- Language: bytecode
- Memory: Heap + Stack + Permanent
- Interpreter
- Operations and Data Structures for:
    - Primitive data processing
    - Sequence control

    - Data transfer control
    - Memory management

Execution models

Pure Interpretation

ML is interpreted over M0: every instruction of ML is translated in real time into a (set of) instruction of M0.

Is not very efficient, but allows to run programs on-the-go, facilitating interactive debugging and testing.

Pure Compilation

Programs written in L are translated into equivalent programs written in L0, the machine language of M0.

The translated programs can be executed directly on M0, hence ML is not realized at all.
The execution is more efficient, but the produced code is larger. Every time we change the code we must recompile it.

Compilation allows aggressive hardware optimization to exploit hardware features.

After the compilation, the linking phase binds the libraries to the translated code and pack everything into an executable.

Case studies

- C++ has a pre-processor that compiles C++ into C code, then the C code is compiled.
- JVM exploits both compilation and interpretation: the Java code is compiled into bytecode, interpreted by the Java Virtual Machine.
- Similarly, Microsoft compiles all its languages (C#, J#, F#, VisualBasic.NET, ...) into a Common Intermediate Language (CIL) that runs on a Common Language Interpreter (CLI).
- Virtual Machines, such as JVM and CLI, uses Dynamic Linking: the linking is executed when needed at runtime instead of at compile time.