Appendix D

Common Language Infrastructure

The *Common Language Infrastructure* (*CLI*), not to be confused with *Command Line Interface* with the same acronym, is a technical standard developed by Microsoft [4, 3]. The standard specifies a language, its format, and a runtime environment that can execute the code. The main feature is that it provides a common interface between many languages and many platforms, such that programs can collaborate in a language-agnostic manner and can be executed on different platforms without having to be recompiled. Main features of the standard are:

Common Type System (CTS) which defines a common set of types that can be used across different languages as if it were their own.

Metadata which defines a common method for referencing programming structures such as values and functions in a language-independent manner.

Common Intermediate Language (CIL) which is a platform-independent, stack-based, object-oriented assembly language that can be executed by the Virtual Execution System.

Virtual Execution System (VES) which is a platform dependent, virtual machine, which combines the above into code that can be executed at runtime. Microsoft's implementation of VES is called *Common Language Runtime* (*CLR*) and uses *just-in-time* compilation. In this book, we have been using the `mono` command.

The process of running an F# program is shown in Figure D.1. First the F# code is compiled or interpreted to CIL. This code possibly combined with other CIL code is then converted to a machine-readable code, and the result is then executed on the platform.

CLI defines a *module* as a single file containing executable code by VES. Hence, CLI's notion of a module is somewhat related to F#'s notion of module, but the

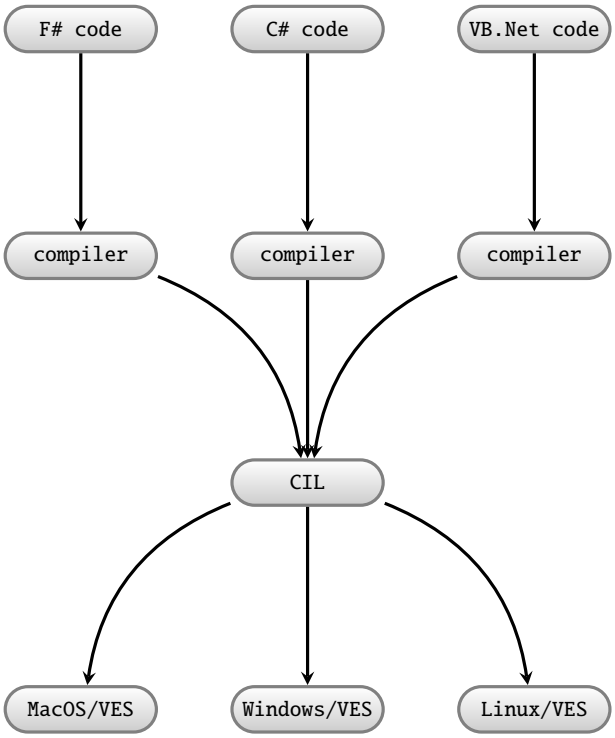

Fig. D.1 The relation between some .NET/Mono languages with the Common intermediate language (CIL), and the Virtual execution systems (VES) on some operating system (`mono`).

two should not be confused. A collection of modules, a *manifest*, and possibly other resources, which jointly define a complete program is called an *assembly*. The manifest is the description of which files are included in the assembly together with its version, name, security information, and other bookkeeping information.