Universiteit Utrecht

[Faculty of **Science**
Information and Computing Sciences]

Compiler Construction

WWW: `http://www.cs.uu.nl/wiki/Cco`

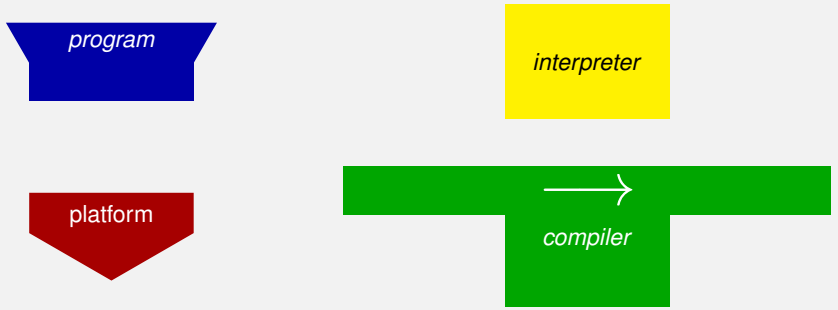
Edition 2010/2011

Agenda

Compilers and interpreters

1. Compilers and interpreters

T-diagrams



T-diagrams are a means to visualise the interactions and relationships between programs, platforms, interpreters, and compilers.

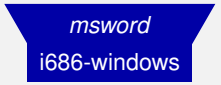
Programs

§1

A program *P*, implemented in an implementation language L.

Programs: examples

§1



Platforms

§1

A (hardware) platform *M*.

Platforms (cont'd)

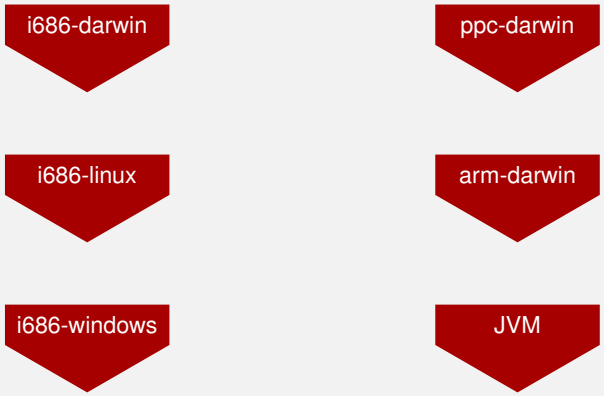
§1

Typically, a platform is a combination of a CPU and an operating system (or, more general, a run-time environment):

- each type of CPU has its own instruction set,
- each operating system has its own conventions for dealing with executables.

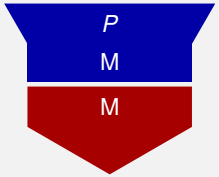
Platforms: examples

§1



Executing a program

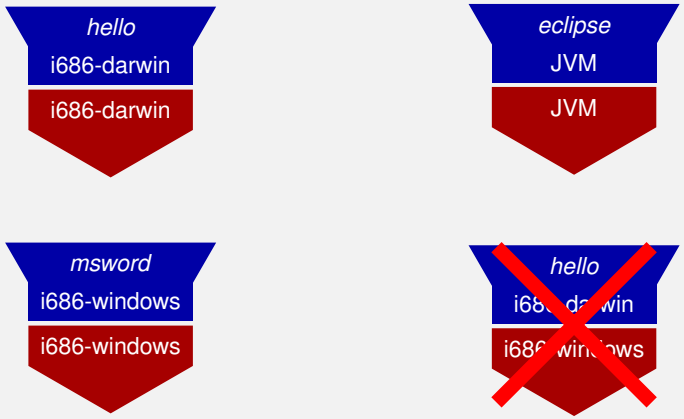
§1



A program *P* can (only) be executed on a platform M if it was implemented in a language that matches M.

Executing a program: examples

§1



(One cannot just run a program for Mac OS on Windows.)

High-level programming languages

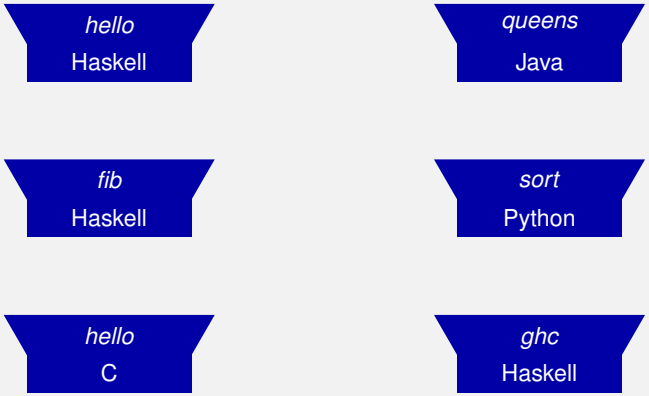
§1

Usually programs are not directly written in the machine language that is supported by the platform on which it is to be executed.

Instead, programs are written in so-called high-level programming languages, such as C, Java, and Haskell.

To execute a program implemented in a high-level language one needs an interpreter for that language. Alternatively, one can use a compiler that translates from the high-level language into the language matched by the target platform.

High-level programming languages: examples §1



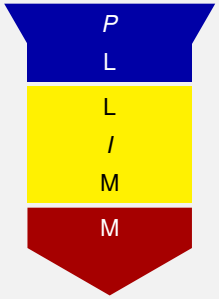
Interpreters §1

An interpreter *I* for an object language L, implemented in an implementation language M.

Interpreters: examples §1

Haskell *hugs* i686-darwin	Python *CPython* i686-linux	JScript *CScript.exe* i686-windows
Haskell *hugs* i686-linux	Perl *perl* i686-linux	VBScript *CScript.exe* i686-windows
Haskell *runghc* i686-darwin	Lua *lua* i686-linux	Ruby *JRuby* JVM

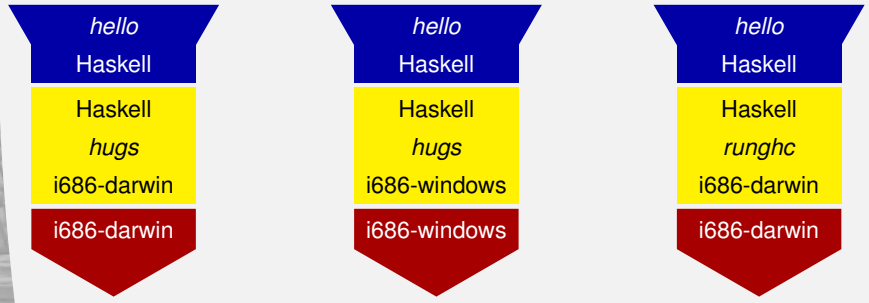
Interpreting a program §1



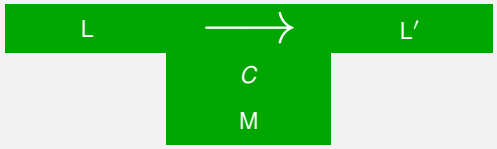
An interpreter *I* for an object langauge L is itself a program that can be executed on a platform for its implementation language M.

Mediating between L and M, it provides a platform on which programs *P* implemented in *L* can be executed.

Interpreting a program: examples

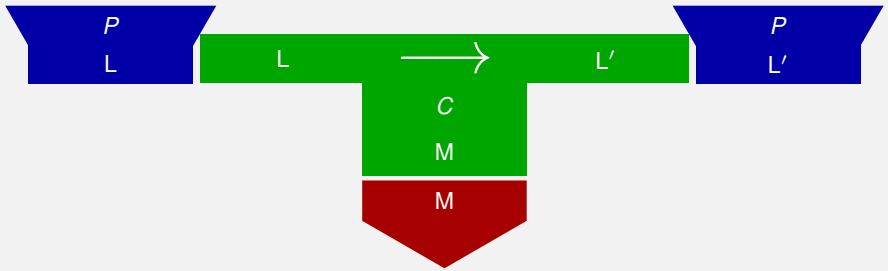


Compilers



A compiler *C*, implemented in an implementation language M, that translates from a source language L into a target language L′.

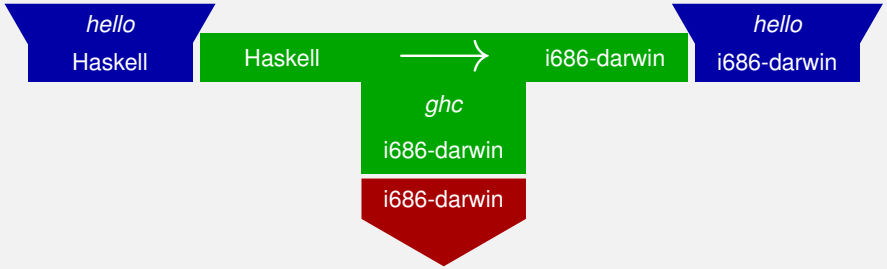
Compiling a program



A compiler *C* for a source language L and a target language L′ is itself a program that can be executed on a platform for its implementation langauge M.

It takes as input a program *P* implemented in L.
It produces as output an implementation of *P* in L′.

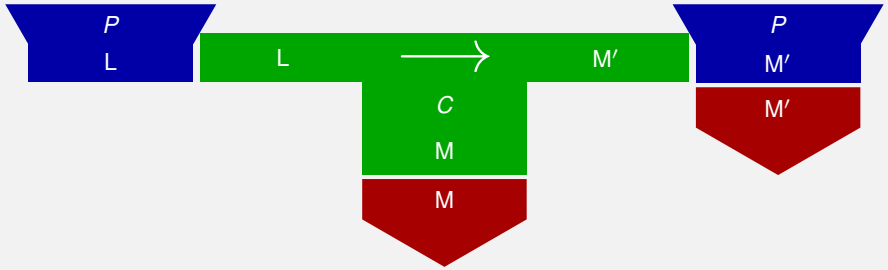
Compiling a program: example



Executables

§1

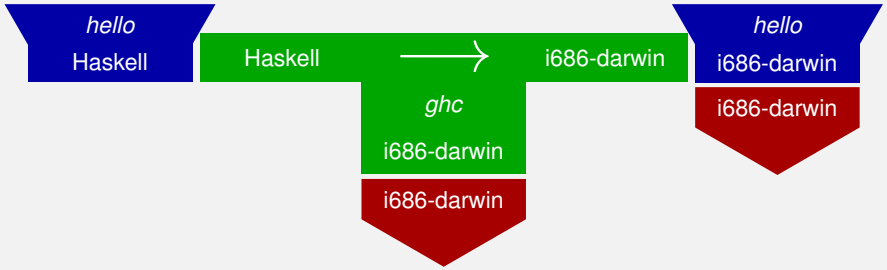
Often—but not always—the compilation target is a machine-executable program:



Such an executable is a lowest-level representation of the source program for the target platform and consists of instructions that can be directly executed by the target machine. Hence, execution can be very fast.

Executables: example

§1



Nonexecutable targets

§1

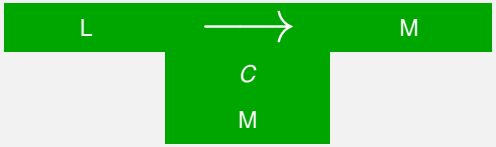
The target of a compiler need not be an executable; any target language will do.

A compiler that translates from one high-level language into another is called a source-to-source compiler.

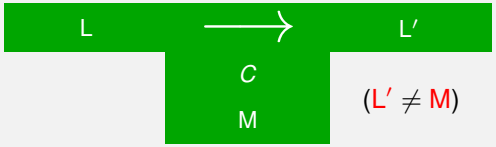
Cross compilers

§1

Often, a compiler is targeted at the same platform it runs on, i.e., its implementation language and target language are the same:



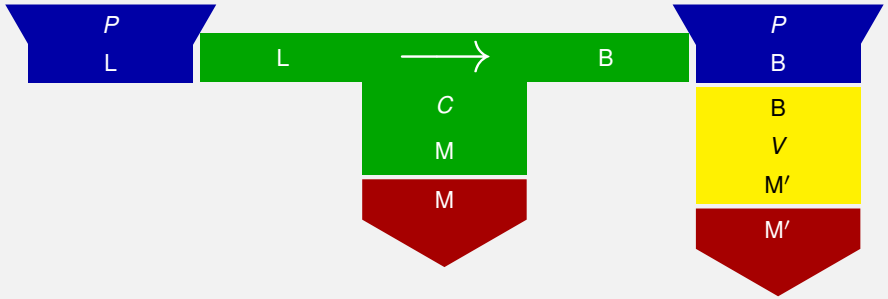
A compiler that generates code for a platform other than the one it runs on itself, is called a cross compiler:



$(L' \neq M)$

Virtual machines

As an alternative to producing machine-executable code directly, sometimes a source program implemented in a high-level programming language is translated into low-level byte code for a so-called virtual machine:



The virtual machine is not a real hardware platform: it is implemented as a byte-code interpreter.

Virtual machines: example

The Java Virtual Machine: