Interpreted Languages and Hybrid Languages

PyLP 361075 Carlos Eduardo Sánchez Torres

March 10, 2021

1 Pure Interpretation

A piece of software (interpreter) translates (i.e. interpret) to an intermediary abstract code and provides a virtual machine on a real machine. The interpreter usually is created a different language (procedural) and executes steps while running on a real machine. Mostly pure interpreted languages are 10 to 100 times slower [1] than in compiled languages by code must be decoded line-to-line, however, this paradigm tends to be flexible, offer dynamic typing and the code itself is platform-independent. Some popular interpreted languages are scripting languages, such as JavaScript, PHP, Python, Bash, etc.

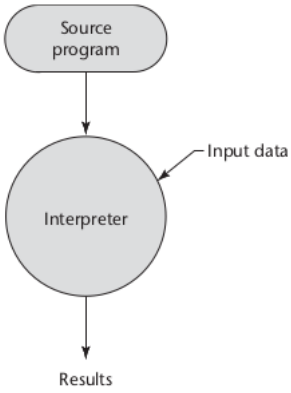

Figure 1: Pure interpretation [1].

1.1 Lisp

LISP derives from ”LISt processor” [4], whose M.I.T. group’s goal was to create a language thought for Artificial Intelligence with his Symbolic Computation and independent of all electronic devices, meaning the first interpreted language in 1968 [5]. Although the first dialect ”LISP 1” runs on IBM 704. The LISP dialects can be written in several languages, such that C++, C o Assembly.

This paper analyzes the dialect Common LISP, which was published in 1994 by ANSI [6]. Common LISP is a general-purpose language. It supports functional and object-oriented programming paradigms. LISP is commonly interpreted by the LIST interpreter which reads code (LISP expressions), maintains data structures, and evaluates them, and shows out the results [7]. It can maintain data structures thanks to the atom table and number table where these lists are recursive.

2 Hybrid Implementation Systems

As purely interpreted language are slower than compiled languages by the type checker and syntactic analyzer, a brings up solution it's the development of Just-In-Time (JIT) compilation and an intermediate language, where the gap is shirking.

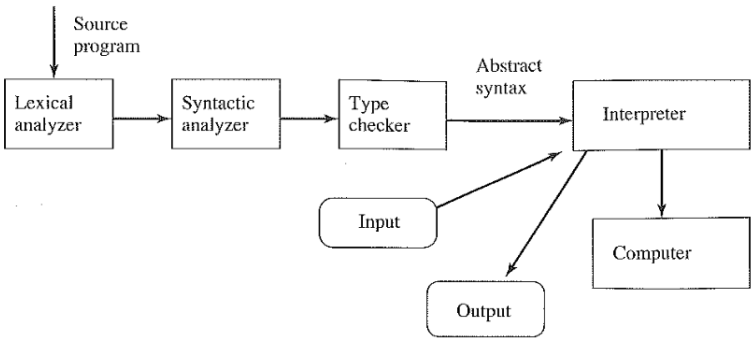

Figure 2: Virtual machines and Interpreters [2].

2.1 Java

Java, whose goal is write once, run anywhere [3], then Java interprets by be platform-independent (i.e. portability). Thus, his applications compile to bytecode, also called portable code or p-code, that can be run on Java Virtual Machine, which translates to the Java bytecode into the host machine. So bytecode is an intermediate representation.

3 Summary

Table 1: What is the difference between a compiled and an interpreted language?

	Paradigms		
	Compiled languages	***Pure interpreted languages***	***Hybrid Implementation Systems***
Advantages	The fastest by native machine code	Flexible and portable	Type checker and portable
Disadvantages	Platform-dependency	The slowest	Slower than compiled languages

References

[1] Tucker, A. B., &; Noonan, R. (2007). Programming languages: Principles and paradigms. In Programming languages: Principles and paradigms. New York: McGraw-Hill Higher Education.

[2] Sebesta, R. W. (2016). Concepts of programming languages. In Concepts of programming languages. Upper Saddle River: Pearson.

[3] "Write once, run anywhere?". Computer Weekly. May 2, 2002. Retrieved July 27, 2009.

[4] Jones, Robin; Maynard, Clive; Stewart, Ian (December 6, 2012). The Art of Lisp Programming. Springer Science & Business Media. p. 2. ISBN 9781447117193.

[5] John McCarthy. "Recursive Functions of Symbolic Expressions and Their Computation by Machine, Part I". Archived from the original on 2013-10-04. Retrieved 2006-10-13.

[6] 2021. ANSI INCITS 226-1994 (R1999) - Information Technology - Programming Language - Common Lisp (formerly ANSI X3.226-1994 (R1999)). Ansi.org. Retrieved March 10, 2021 from https://webstore.ansi.org/standards/incits/ansiincits2261994r1999

[7] Gary Knott. 2008. Interpreting LISP. Retrieved March 10, 2021 from http://www.civilized.com/files/lispbook.pdf