Manual
Verse Interpreter

Course: Logic programming

Date: 26. September 2023

Participants: Onuralp Mete, Clemens Hafenscher

Inhaltsverzeichnis

1 General **1**

2 Code Execution **2**

3 Features **5**

4 Syntax **7**

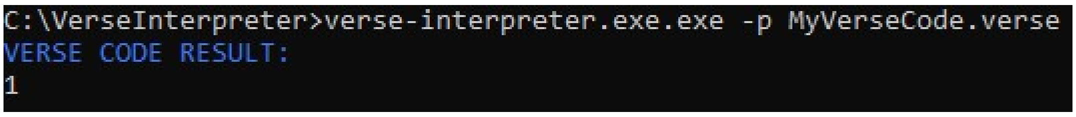
1 General

The following document serves as guidance and documentation for code execution, missing and implemented features as well as syntax deviations from the expected verse syntax. The following points are explained in more detail below:

- *Code execution*
- *Implemented and missing features*
- *Syntax deviations*

2 Code Execution

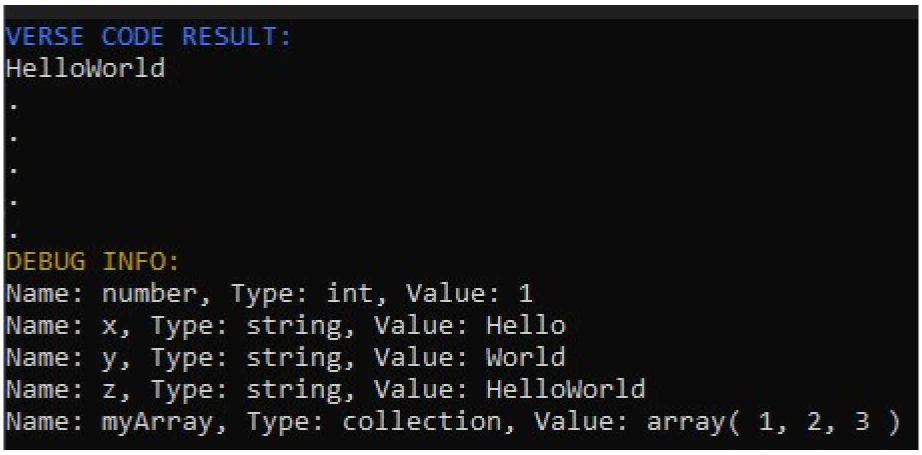
The interpreter executes verse code by specifying a path with command line arguments in the console. Specifying a path for a .verse file is done by -p as in Abb. 1:



```
C:\VerseInterpreter>verse-interpreter.exe.exe -p MyVerseCode.verse
VERSE CODE RESULT:
1
```

Abbildung 1: Example for code execution

With the specification of -d it is possible to activate the debug mode, which lists all variables, their type and values in the console. Abb. 2 serves as an example for the debug mode:



```
VERSE CODE RESULT:
HelloWorld
.
.
.
.
.
DEBUG INFO:
Name: number, Type: int, Value: 1
Name: x, Type: string, Value: Hello
Name: y, Type: string, Value: World
Name: z, Type: string, Value: HelloWorld
Name: myArray, Type: collection, Value: array( 1, 2, 3 )
```

Abbildung 2: Example for Debug-Mode

3 Features

All implemented features are listed below:

- Variables and Bindings
- Arithmetic
- Typesystem
- Custom data types
- Arrays and Indexing
- Functions, Function Composition and Recursion
- Lambda-Functions
- If-Conditionals, Conjunction and Disjunction
- for
- ?-operator
- Choice
- false?
- Range expressions
- Narrowing
- Partial Values
- Lenience
- Equality
- Predefined functions (e.g. head or snoc)

  The predefined functions are written in verse code and can be found in the standard library.verse file, which is automatically loaded by the interpreter. The only exception is Print, this is implemented with the C# internal *Console.WriteLine* method.

- Code-Comments

  Single-Line: # ...

  Multiline: #* ... *#

All missing features are listed below:

- Choice pops to the top
- for do
- Higher-order functions
- Predefined function 'map'

4 Syntax

Syntax deviations from the expected verse syntax are listed and described below:

- Arrays can only be created in the long form with the 'array' keyword and round brackets instead of curly brackets.
- If expressions do not require a 'then' keyword, 'else' blocks are optional and curly braces are required for the blocks.
- Function bodies are not defined with ':=' but via curly brackets.
- 4 Whitespaces Rule: Function bodies need 4 whitespaces as indentation in case of a line break with curly brackets or the function body can also be defined without a line break with curly brackets in only one line.
- With for, variables must be declared beforehand when narrowing. It is not possible to create variables directly during indexing.
- For the definition of own data types the keyword 'data' and for the instantiation 'instance' is needed. In addition, the 4 Whitespaces Rule applies.
- Collection: To give a variable or parameter the 'collection' type :collection must be used.