ColdCoffeeScript User Guide

James Curran and Shahaan Hassan

May 3, 2017

1 Introductory Haiku

We have distilled the
One and only Gennaro
Into a language

2 "Serious" Introduction

ColdCoffeeScript is a strongly and statically typed imperative, interpreted, domain-specific language, created for the purpose of manipulating regular languages. The language prides orthogonality over syntactic sugar so the developers have DELIBERATELY AIMED for a minimal syntax. All functions and operators can be treated as pass-by-value. Variables are only changed when explicitly re-assigned in the program. Four data types are supported: integer, boolean, string and set (of strings). For details on supported operations, see the "Syntax" section.

3 Features

ColdCoffeeScript boasts many features, guaranteed to make programming simply delightful. These include:

- Informative error messages: Delicious Gennaro-themed dialogue, for lexing, input, type and runtime errors
- Type checking: Rigorous type checker to whip those unruly variables into shape before they go into the program!
- Programmer convenience: Single line comments - include supplementary literature in your program by surrounding part of a line with hashes, like so #Shahaan remember to add more features under here#

4 Syntax

4.1 Ignored characters

Any characters surrounded by `#` symbols are treated as comments and ignored by the interpreter. Outside of these symbols, only whitespace and the newline character are ignored.

4.2 Program format

A ColdCoffeeScript program is a series of statements (if(-else), (re-)assignment, while or display), separated by a semicolon. Line comments can appear on any line of the program.

4.3 Types

- ColdCoffeeScript integers (named `int` in code) are specified by any sequence of the characters 0-9. Nota Bene: Integers and their arithmetic are bounded by the OCaml integer specification - see caml.inria.fr for more information. We certainly did.
- Booleans (`bool` in code) are standard boolean types, denoted by `true` and `false`.
- Strings (`string` in code) are sequences of characters from the lowercase English alphabet. The exception is `:`, which represents an empty string. Strings are encased in double quotes in the program, but not in program input or output.
- Sets (`set` in code) are ordered collections of strings. A set cannot contain two identical strings, and the set is reordered automatically if necessary. Strings in sets follow alphabetical order, and if : is in the set it will be at the start.

4.4 Arithmetic Operators

All arithmetic operators are infix and only involve integers. The standard `+`, `-`, `*` and `/` are used, as well as `<`, `==` and `>` for comparison. Our distaste for syntactic sugar is so merciless that a negative number can only be expressed as a result of integer subtraction.

4.5 Boolean Operators

All boolean operators are infix. The operators are the standard `not`, `and` and `or`, with `==` to check for equality.

4.6 String Operators

All string operators are infix. `+` is used to concatenate strings. `==` is used to check for equality.

4.7 Set Operators

The set operators are infix. ColdCoffeeScript provides `union`, `intersect` and `concat`. Also, `<set A> difference <set B` is used to remove all values in set B from set A.

4.8 Other Operators

`<string A> memberOf <set B>` is a boolean operator that checks if A is a member of B.
`display <set A> <integer B>;` will output the first B elements of A.

4.9 Variables

A variable is created by stating its type, then name, followed by an equals sign, followed by the desired value. This is a statement so a semicolon is also needed.

- `int MYNUMBER = 41;`
- `bool MYBOOL = true;`

- `string MYSTRING = "yawn";`
- `set MYSET = {"are", "you", "bored", "yet"};`

The variable's value is reassigned with the name, followed by an equals sign, followed by the new value, followed by a semicolon:
`MYSET = {"god", "yes","someone","help","me","please"};`
It is of course also possible to (re-)assign the result of some expression to a variable, provided the result is of the type the variable was declared with:
`MYSET = {"this", "joke", "has", "run"} union {"its", "co"+"urse"};`
Nota Bene: variable names can only consist of capital letters. This was DONE BY CHOICE to emphasise the importance of good variable naming.

4.10 Using Input

Input follows a strict structure where each line must be a set of strings, except for the last line which must be a number. Whitespace is ignored. For example:

```
{a, b, c,        d}
{:}
{words}
4
```

Input is then accessed through pre-assigned variables at the start of the program. `L1` is the language (set of strings) on line 1 of input, `L2` on line 2 and so on. `K` is the number at the end. These variables can be reassigned in code, but their original values cannot be retrieved unless they were saved elsewhere.

If a program attempts to access a language which has not been specified, e.g. tries to access L4 when only two languages were given in input, it will crash with a LookupError.

4.11 Conditionals and Iteration

This is a very brief overview of the if(-else) and while constructs in ColdCoffeeScript.

```
if <bool> do
    <body>
end;

if <bool> do
    <body1>
else
    <body2>
end;

while <bool> do
    <body>
end;
```

`<bool>` is an expression that evaluates to a boolean. `<body>`, `<body1>` and `<body2>` are series of statements, just like the overall program. For an example of conditionals, refer to the FizzBuzz program in the Appendix. For an example of iteration, refer to the Kleene Star program, also in the Appendix.
Indentation is not necessary; ColdCoffeeScript code can take any shape desired by the programmer. To encourage this, code which uses whitespace to be shaped like a steaming cup of Italian coffee is permitted by the interpreter to run 50% faster.[1]

[1]Completely untrue

5 Running the Interpreter

5.1 Windows

In the same directory as mysplinterpreter, the interpreter is run with the command

```
ocamlrun mysplinterpreter [program file]
```

and will then wait for your input. You may input any number of sets, and inputting a number will start the program running. The command

```
ocamlrun mysplinterpreter [program file] < [input file]
```

allows you to use an input file instead of entering the input.

5.2 Linux

In the same directory as mysplinterpreter, the interpreter is run with the command

```
./mysplinterpreter [program file]
```

and will then wait for your input. You may input any number of sets, and inputting a number will start the program running. The command

```
./mysplinterpreter [program file] < [input file]
```

allows you to use an input file instead of entering the input.

6 Appendix: Example Programs

6.1 Hello World

This is the anatomy of a Hello World Program in ColdCoffeeScript.

```
display {"hello","world"} 2;
```

The output of the program is as follows:

```
{ciao,mondo}
```

This is because the language automatically translates English strings to Italian.

6.2 Not Really

The actual output is:

```
{hello,world}
```

6.3 FizzBuzz

The best FizzBuzz algorithms in other languages have $\Omega(n)$ complexity; however, thanks to the properties of languages, the complexity in ColdCoffeeScript can be greatly reduced. Firstly, numbers are not permitted in languages so there is no need to write code to add integers to languages. Secondly, multiple instances of "Fizz", "Buzz" or "FizzBuzz" will be reduced to a single instance of each in the final set. We now therefore present an $O(1)$ ColdCoffeeScript FizzBuzz program, which will produce the same output as any other FizzBuzz program in ColdCoffeeScript:

```
if K > 14 do
    display {"buzz", "fizz", "fizzbuzz"} 3;
else if K > 4 do
    display {"buzz", "fizz"} 2;
else if K > 2 do
    display {"buzz"} 1;
else
    display {} 1;
end;
end;
end;
```

6.4 Kleene Star

There is no built-in function to generate the asterate of a language, but code can be made with relative ease.

```
#Generate first K elements of {"a"}*#
int I = 0;
set STAR = {};
while I < K do
   STAR = ({"a"} concat STAR) union {":"};
   I = I + 1;
end;
display STAR K;
```