Zoea Manual

Introduction

Zoea is a simple declarative programming language based on the concept of inductive programming. Instead of writing code in the conventional sense the zoea developer describes the behaviour of a program as a set of scenarios consisting of inputs and associated outputs. As a result zoea programs resemble a set of functional test cases. The zoea compiler uses AI to turn a zoea program into executable code.

The zoea language takes little time to learn. Programming in zoea mostly involves creating examples of data that demonstrate the required behaviour.

Zoea includes a number of features that allow it to produce larger programs. These include the ability to specify one or more intermediate values between an input and an output. It is also possible for smaller programs to be combined to form larger programs. These facilities enable zoea to be used to create programs of any size.

Visit https://zoea.co.uk/ for more information on zoea.

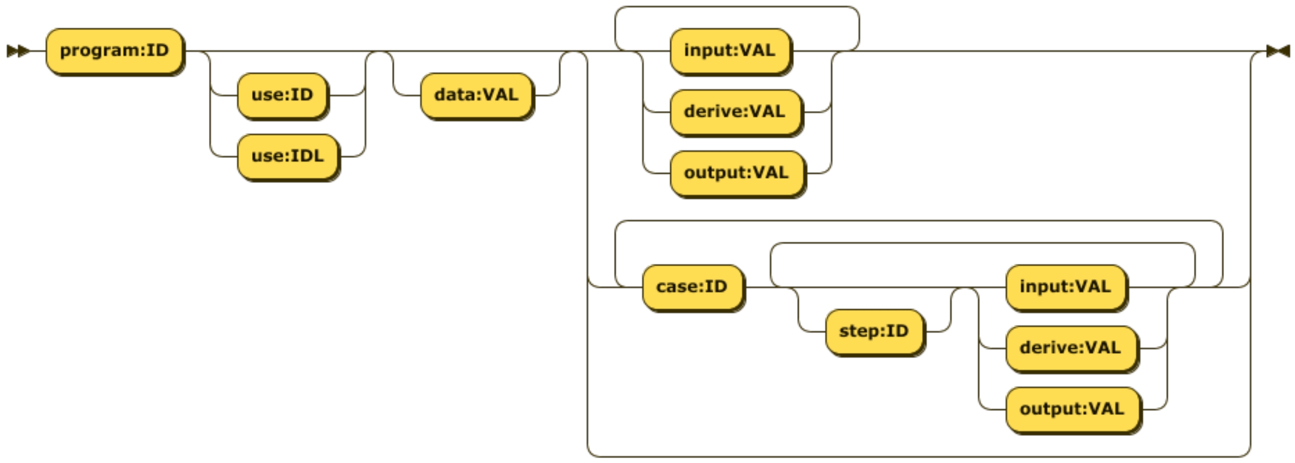
Overview

Zoea programs are composed of a set of one or more terms that have the form:

<keyword> : <argument>

Terms are separated by spaces. The layout of zoea programs is not significant.

The following syntax diagram describes the structure of the entire zoea language.



This uses the following abbreviations:

- ID = an identifier (any non-empty string or a positive integer)
- IDL = a list of identifiers
- VAL = a zoea value

The behaviour of all zoea terms is summarised in the following table:

Keyword	Argument	Description
program	identifier	Introduces a new program and gives it a name
use	identifier or list	The current program should invoke these other existing programs
data	value	The program will need this data that will not be input by the user
case	identifier	Introduces a new test case and gives it a name
step	identifier	Introduces a new test case step and gives it a name
input	value	In the current step this value is input by the user
derive	value	In the current step this value is derived by the program
output	value	In the current step this value is output by the program

Single line comments begin with the hash character (#) and terminate at the end of the current line.

Values

Values in zoea are very similar to JSON. JSON is a subset of JavaScript that is widely used for data representation. See https://json.org for more details on JSON. One significant difference is that quotes can be omitted from strings that do not need them. For example the string 'fred' can also be written without the quotes as it contains no spaces or special characters whereas the string "hello world" always requires quotes as it contains a space. Single or double quotes can be used provided they are balanced.
Some examples of values in zoea are:

Value	Description
''	Empty string (single quotes)
a	unquoted string of length one (or a character)
fred	unquoted string
"bread and butter pudding"	quoted string (double quotes)
123	integer
3.142	real number
[]	empty list
[tom,dick,harry]	list of strings
[1,2,3,4]	list of integers
{}	empty hashmap (or equivalent concept)
{ landmark: "Big Ben", location: "London" }	Hashmap with some data

Another significant difference between zoea and JSON is that any value or any element of a composite value can be replaced by a single unquoted underscore character. This construct means that we do not care what value this element has in the current test case as it will not be used in the corresponding generated code. For example [1,2,_] is a list of three elements but we do not care about value of the third element – in effect it can take any value. An underscore character in quotes or in strings of two or more characters is a normal underscore.

Every zoea term has a single argument which may be an identifier or a zoea value depending on the keyword. If multiple values are required for program input or output then these are combined to form a single list.

Language rules

Every program:

- must have one program identifier
- may have one use value (identifier or list of identifiers)
- may have one data value
- may have any number of cases (including zero)

Every case:

- may have one optional case identifier if there is just one case
- must have one case identifier if there is more than one case
- must have one or more steps

Every step:

- may have one optional step identifier
- must have either one input value or one derive value or one output value

Keywords

This section contains details of each zoea keyword. These are presented in an order that supports learning in the sense that later concepts build on earlier ones.

Program

Format	program : <identifier>
Purpose	The program term is used to define the name of a program. Every zoea program has a name that also needs to be unique for each user. As a result every zoea program must include one program term which also must be the first term.
Value	Identifier – non-empty string or integer. Must be unique for each program for a given user
Frequency	Mandatory and can only occur once within each program
Occurs	As the first term of each program
Example	# this is a valid zoea program that does nothing program: do_nothing

Output

Format	output : <value>
Purpose	The output term defines the output value of a test case step. Each case consists of one or more steps and each step includes a single input, derive or output term.
Value	Zoea value
Frequency	As needed
Occurs after	• Step • case (in the first step if the step term is omitted) • program, use, data (in a single case if the case and step terms are omitted) • input, derive, output (after the first step if the step term is omitted)
Example	# this zoea program always produces the same output program: hello output: "hello world"

Input

Format	input : <value>
Purpose	The input term defines the input value of a test case step. Each case consists of one or more steps and each step includes a single input, derive or output term.
Value	Zoea value
Frequency	As needed
Occurs after	• Step • case (in the first step if the step keyword is omitted) • program, use, data (in a single case if the case and step keywords are omitted) • input, derive, output (after the first step if the step keyword is omitted)

Example	# this zoea program always copies the input to the output
	program: copy_input_to_output
	  input: fred
	  output: fred

Derive

Format	derive : <value>
Purpose	The derive term defines the derived value of a test case step. A derived value is an internal value that is intermediate in some way between an input and an output. Each case consists of one or more steps and each step includes a single input, derive or output term.
Value	Zoea value
Frequency	As needed
Occurs after	• Step • case (in the first step if the step keyword is omitted) • program, use, data (in a single case if the case and step keywords are omitted) • input, derive, output (after the first step if the step keyword is omitted)
Example	# example zoea program showing a derived value between an input and an output program: remove_spaces_and sort_letters input: " the quick brown fox jumps over the lazy dog " derive: "thequickbrownfoxjumpsoverthelazydog" output: "abcdefghijklmnopqrstuvwxyz"

Case

Format	case : <identifier>
Purpose	The case term defines the identifier for a case. Each program can contain zero or more cases.
Value	Identifier – non-empty string or integer. Must be unique within a given program.
Frequency	Optional if there is only one case in the program otherwise mandatory for every case.
Occurs after	• Program (in the first case if use and data are omitted) • use (in the first case if data is omitted) • data (in the first case) • input, derive, output (after the first case)

Example	# zoea program with two cases
	program: number_squared
	  case:1 input: 2 output: 4
	  case:2 input: 3 output: 9

Step

Format	step : <identifier>
Purpose	The step term defines the identifier for a test case step. Each case consists of one or more steps.
Value	Identifier – non-empty string or integer. Must be unique within each case
Frequency	Step is optional in cases where the case term is present. While it is possible to provide a step term for a subset of steps this is not recommended. In cases where the case term is present a step term should either be present for every step or else for none of them. Step cannot be used if case is omitted.
Occurs after	• Case (first step) • input, derive, output (after first step)

Example	# zoea program with multiple steps with step identifiers
	Program: reverse_input
	  case: 1
	    step:1 input: banana
	    step:2 output: ananab

Data

Format	data : <value>
Purpose	The data term defines the value of static reference data. Reference data is information that is used by a program but which does not change over time and thus does not need to be input by the user. If multiple items of reference data are required then they can be joined into a list.
Value	Zoea value – underscores have no practical use in this context but are harmless if included
Frequency	Optional or once per program.
Occurs after	• program (if use is omitted) • use

Example	`# zoea program with reference data`
	`program: g20_member`
	`  data: [ar, au, br, ca, cn, fr, de, in, id, it, jp, kr, mx, ru, sa, za, tr, uk, us, eu]`
	`  case: 1 input: us output: is_g20_member`
	`  case: 2 input: ie output: not_g20_member`

Use

Format	Use : <identifier or list of identifiers>
Purpose	The use term identifies the names of one or more existing programs that are to be used as additional instructions in the current program. All existing programs included in a use term must have been compiled successfully before the current program can be compiled. Note that use is treated as a hint by the user that the given programs should be utilised if possible. As a result valid solutions may be generated that do not utilise some or any of the identified programs.
Value	Identifier or list of identifiers
Frequency	Optional or once per program.
Occurs after	• program

Example	`# example of one zoea program used by another`
	`program: area_of_circle`
	`  data: 3.142`
	`  input: 10 output: 314.2`
	
	`program: volume_of_cylinder`
	`  use: area_of_circle`
	`  input: [10,2] output: 628.4`