Zoea

Zoea Visual User Guide

Version 1.0

1. Introduction

Zoea Visual is a visual programming language. A visual language allows a user to produce software using a graphical notation - typically diagrams - instead of writing code. Zoea Visual is built on top of the Zoea language. It also extends Zoea with a number of new features that are described in the following sections.

This document assumes some familiarity with the concepts used in Zoea. Visit https://zoea.co.uk for more information on Zoea and Zoea Visual.

2. Concepts

2.1 Programs

Every executable unit of software in Zoea is called a program. A program has a set of inputs and a set of outputs. Other programming languages provide additional software constructs like subroutines and functions. In Zoea everything is a program. Each Zoea Visual program has a unique program name. The program name can be any length and can include spaces.

2.2 Cases

A Zoea Visual program is composed of one or more test cases. Test cases are often referred to as simply cases. Each case defines the behaviour of the program in terms of what outputs are expected for a given set of inputs.

2.3 Inputs and Outputs

A Zoea Visual program can have any number of inputs and any number of outputs. It is possible to have a program with no inputs and no outputs which does nothing. It is also possible to have a program with one or more outputs but no inputs. However, most programs will have both inputs and outputs.

2.4 Derived Values

As with Zoea it is possible to define one or more intermediate values between the inputs and outputs of a given case. In Zoea and Zoea Visual intermediate values are called derived values.

2.5 Data

Static reference data can be included in a Zoea Visual program so that it does not need to included as an additional input every time the program is run. Each Zoea Visual program can have any number of reference data elements.

3. Notation

3.1 Data Representation

Programs in Zoea Visual consist almost entirely of data. All data elements in Zoea Visual are represented visually. Figure 1 shows how the various data elements are represented.

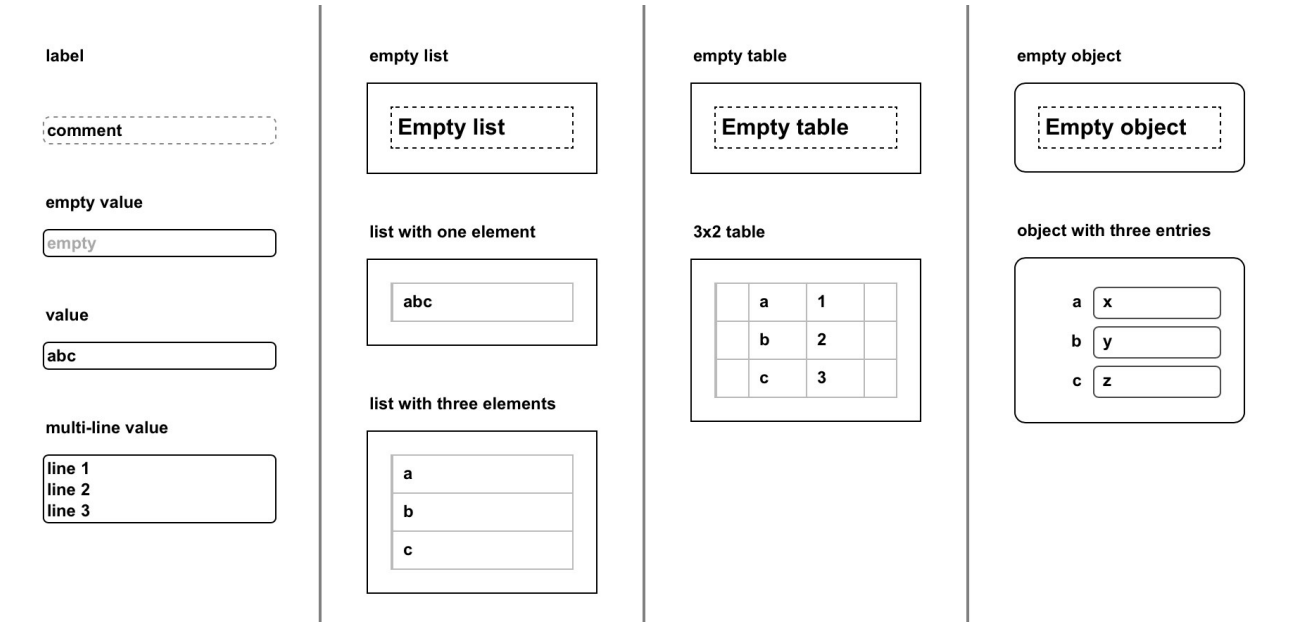

Figure 1 – Zoea Visual data elements

3.2 Label

A label is a fixed string that is ignored by the Zoea Compiler. Labels are used to describe other data elements in input and output forms.

3.3 Comment

A comment in Zoea Visual is represented as a box with a dashed border. This can contain any amount of text. Comments are ignored by the Zoea compiler.

3.4 Value

Strings and numbers consisting of a single value are represented as a textarea input field. A value can be any size and may include multiple lines of text. Values that are empty display the word 'empty' in grey. The value box grows and shrinks automatically to fit the enclosed text up to around 50% of the screen height. After this point scrolling will be necessary to see the entire value.

3.5 List

A list in Zoea Visual corresponds to a one dimensional array. A special notation is used for an empty list.

3.6 Table

A table represents a two dimensional array. A special notation is used for an empty table.

3.7 Object

Associative arrays or hashmaps are called objects in Zoea. These are represented as a box with round corners. Objects contain zero or more key-value pairs. Both the key and the value are editable even though the key has no border.

3.8 Data composition

It is possible to create more complex data structures by replacing the elements in a composite structure with a different type. To do this the old element is first selected and the new element then created. This can be used for example to create multi-dimensional arrays or lists of objects.

3.9 Columns

In Zoea various tags are used to indicate whether a data element is an input, derived value or output. Zoea Visual provides a set of named columns in which to place data elements. Figure 2 shows the different types of columns provided. It is possible to have any number of derive columns. The data and derive columns can be hidden if they are empty. The input and output columns are always displayed.

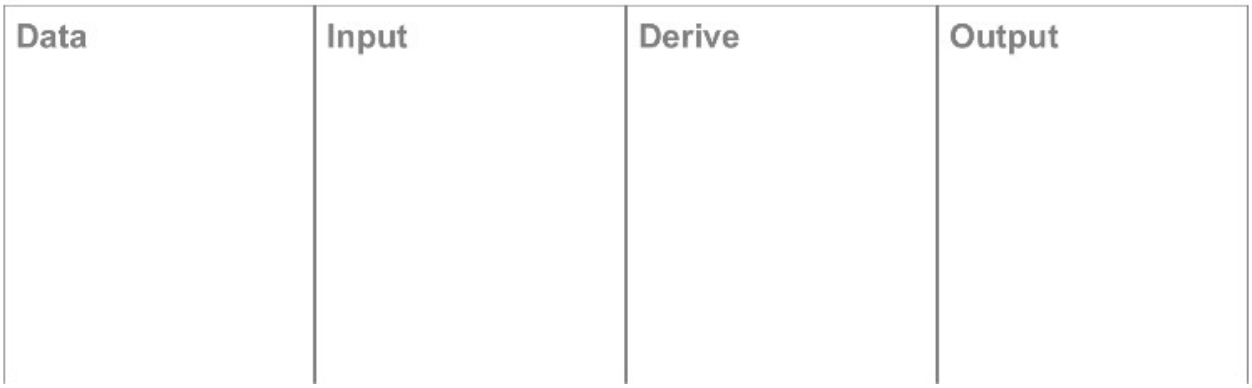

Figure 2 – Zoea Visual columns

4. Interaction

4.1 Element Selection

All data elements in Zoea Visual can be selected by the user. The user selects an element by clicking inside it with the mouse. Selected elements are coloured yellow. Elements are deselected by clicking again. Deselected elements revert to a white background. It is possible to select more than one element at the same time.

Each element in a list can be selected. It is also possible to select the entire list by clicking in the margin that surrounds it. Selection of objects behaves similarly.

Selection in tables is similar to lists except that it is also possible to select an entire row in a table. This is done by clicking on the small box at the start or end of each row.

4.2 Column Selection

A single column can be selected at any given time. If a column is selected then any data elements that are currently selected become deselected. Similarly is a column is already selected and any data element is then selected then the column will become deselected.

4.3 Element Creation

Elements are placed in columns by first selecting the desired column and then choosing the appropriate element type from the menu.

4.4 Element Deletion

Elements can also be removed from a column by selecting them and then choosing delete. Any number of elements can be deleted at the same time.

4.5 Resizing Elements

Composite element creation buttons have associated dropdown menus that are used to specify the size in terms of the number of elements required. Lists and objects have a single size value whereas tables have two corredponding to the number of rows and columns. Size can be specified either before element creation or at any later time by first selecting the data element and then selecting the required size.

4.6 Element Order

Elements can be moved up and down within a column to change their order. This is accomplished by first selecting the element and then selecting the up or down button.

4.7 Column Width

At any point in time all columns have equal widths. It is possible to increase or decrease the width of all columns simultaneously. This is useful either to be able to see more of the diagram or to view large data elements or values.

4.8 Column Offset

All of the elements in a single column can be moved down relative to the column top. This is done by first selecting the column and then choosing the down button. Elements can also be moved back up. The benefit of this facility will become apparent later when we cover diagrams.

5. Zoea Visual Concepts

5.1 Dependencies

Zoea Visual also allows for the creation of data flow relationships between data elements. These are referred to as dependencies. Dependencies can only be created between two elements which are located in different columns. A dependency indicates that the data element on the left is some how used in the formulation of the value of the data element on the right. The left side data element is called the source and the right side data element is called the target. Each data element in a case may be both the source and the target of different dependencies. A given data element may be the target of many different dependencies and also the source of many other dependencies.

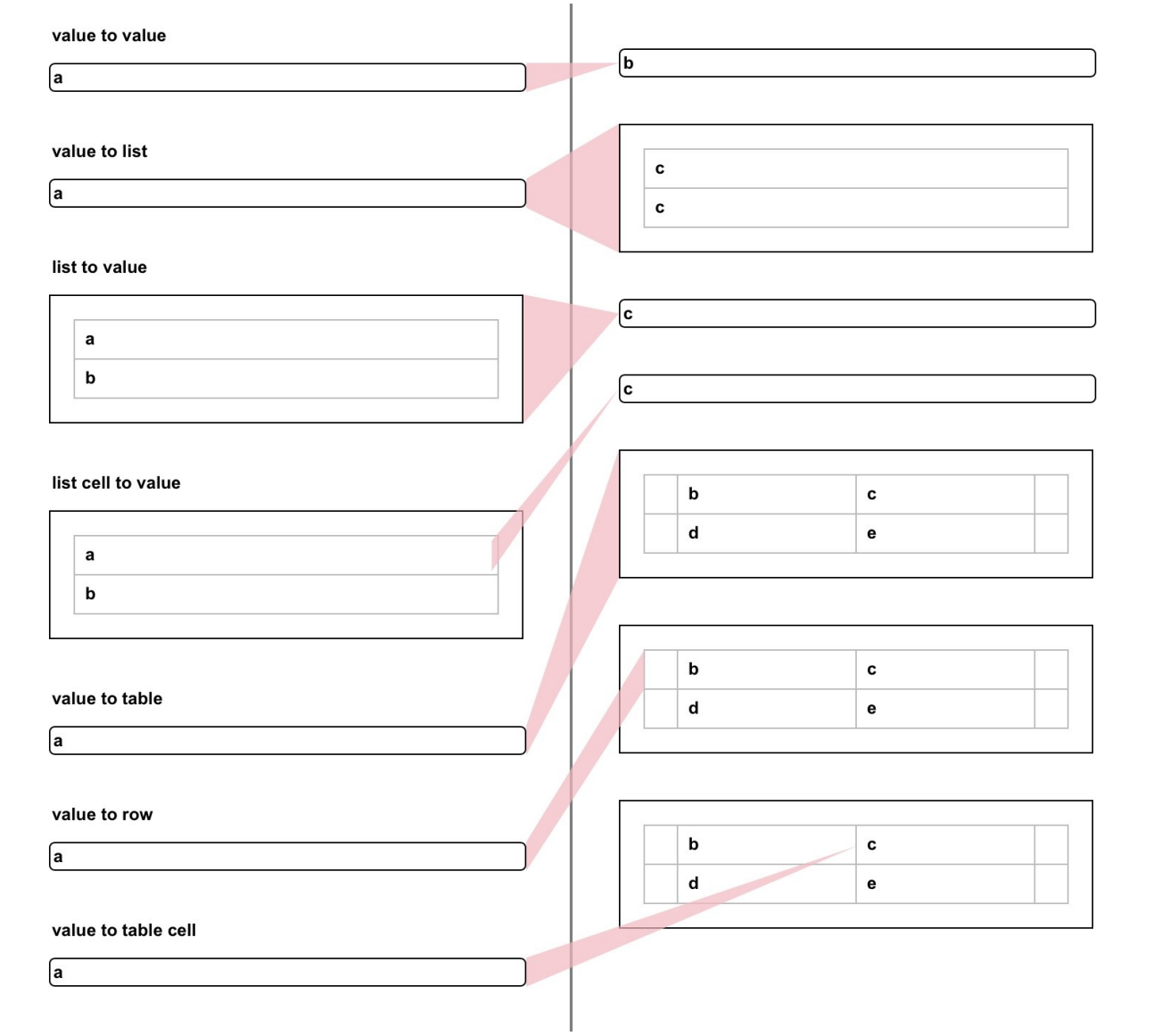

Figure 3 – Dependency examples

Dependencies provide the Zoea compiler with valuable information about which data elements are involved in each step of the computation. This means that Zoea Visual can generate much larger programs using dependencies and also do so more quickly. When dependencies are created they are coloured red.

Dependencies are created by selecting two or more data elements in different columns and then choosing the link button. The target elements are those in the farthest right hand column. Dependencies are created for every other source element that is selected.

The use of dependencies is entirely optional. Zoea Visual will still work without dependencies but it will take longer to compile equivalent programs than if dependencies were used.

5.2 Dependency Notation

A dependency is represented using a semi-transparent coloured polygon that connects two data elements. Figure 3 shows examples of dependencies between different types of data element. It is possible to create dependencies between single value or composite data elements in any combination. Note that in composite data elements the source or target may be the entire element or any one of its children. If the target is a single value then the dependency will resemble a triangle. A composite value target results in a dependency that has four sides.

5.3 Dependency Visibility

Dependencies are always displayed as somewhat transparent. This is to allow information that they might otherwise obscure to be visible. When a dependency is created it is more opaque and appears darker. Also when a data element is selected any dependencies in which it plays the role of source or target are highlighted. All other dependencies become more transparent. At any time the user can cause all dependencies to become more visible by selecting the show button. If this is pressed a second time then all dependencies become more transparent.

5.4 Case Diagram

The data elements and associated dependencies form a diagram that represents a case. Figure 4 shows an example of a complete case diagram. A case diagram is a form of data flow diagram. The case diagram shows the data flow from each input or reference data element to every output data element. The dependencies in this example are blue as the diagram has come from the listing view which is described later. Note also that the first derive column in Figure 4 has been shifted down to improve the layout of the diagram.

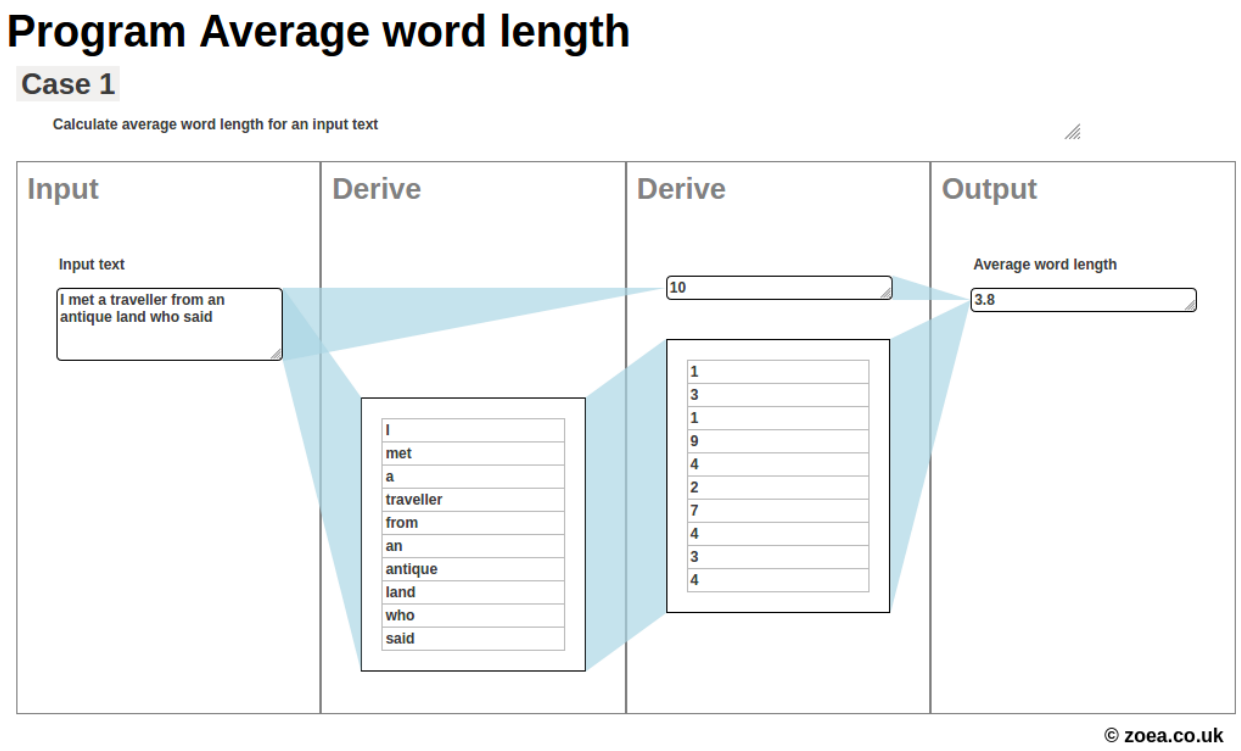

Figure 4 – Case diagram

5.5 Subsidiary Test Cases

Zoea Visual provides the ability to define subsidiary test cases. A subsidiary test case is effectively the definition of another complete program with its own set of case diagrams. Each subsidiary test case is associated with a specific target data element and the set of dependencies in which that element plays the role of target.

Subsidiary test cases are used to describe complex logic or data transformation. They can include any amount of detail and this is kept separate from the context of the case in which the subsidiary is included. Subsidiary test cases serve a similar purpose to the 'use' facility in Zoea. That is they enable composition but because they are more explicit in terms of source and target data elements they allow compilation to proceed more rapidly.

Subsidiary test cases are created by selecting a target element and pressing the sub button. This creates a new set of test cases that are populated with the target data element together with all of the source data elements and their corresponding values and dependencies. Regardless of where they occur in the parent test case all of the source elements are copied to the input column of the subsidiary and the target element is copied to the output column. Any data column elements from the parent test case are also copied to the subsidiary data column.

Within the subsidiary cases can be deleted or added as required. All of the data element values can be changed and the data column elements removed from the subsidiary if they are not required. In addition the inherited dependencies can be deleted and replaced with new dependencies and any number of derived values. The only restrictions on subsidiaries are that input and output elements cannot be added, deleted or reordered. It is also possible to define subsidiaries within subsidiaries so long as they are not created in the output column.

When a data element has a subsidiary it is marked with a blue dot in the case diagram. Existing subsidiaries can be edited by selecting the target data element and selecting the sub button. Inside a subsidiary the back button navigates to the parent test case rather than to the program list.

5.6 Data element identifier

Each data element has a unique numeric identifier. The value of this identifier is not normally significant to the user but it is important for the Zoea compiler. This is because the identifier can signify that data elements in different cases either refer to the same thing or that they are different.

If a test case is cloned or a subsidiary test case is created then all of the elements in both test cases will have the same identity. When a new test case is created from scratch or elements have been added to a cloned test case then elements will have different identities. This situation is not visible on the cases screen but is obvious on the overview screen. It is also possible for the user to manage element identity on the overview screen which is described later.

6. User Interface

6.1 Navigation

The navigation structure of Zoea Visual is shown in Figure 5. The various screens are described below.

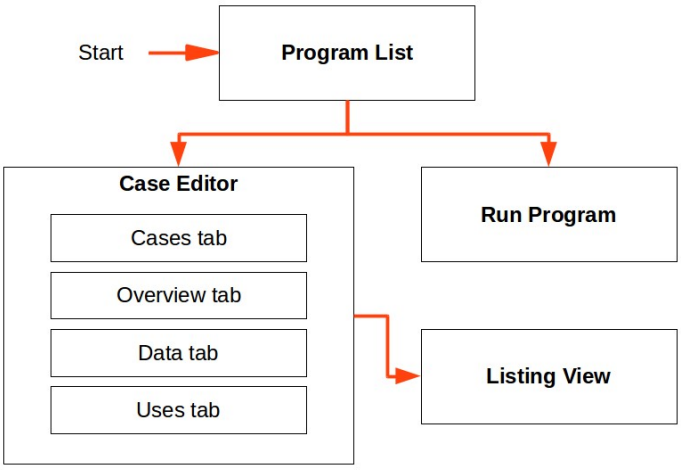

Figure 5 – Zoea Visual navigation structure

6.2 Program List

The program list screen (Fig. 6) is the entry point for Zoea Visual when it is launched. It displays a list of existing program names sorted alphabetically. The '+' icon is used to create a new program. Each program also has a menu accessed via the '...' button on the right. The program list menu is described in the table below.

Menu Item	Function
Edit	Open the selected program in the Cases screen.
Run	Open the selected program in the Run screen if it has been compiled.
Rename	Rename the selected program. Prompts the user for the new name.
Delete	Delete the selected program. Prompts the user for confirmation.

Figure 6 – Program list

6.2 Cases Screen

The cases screen provides the entry point for case editing and supports editing the case diagram. Figure 7 shows the case screen layout.

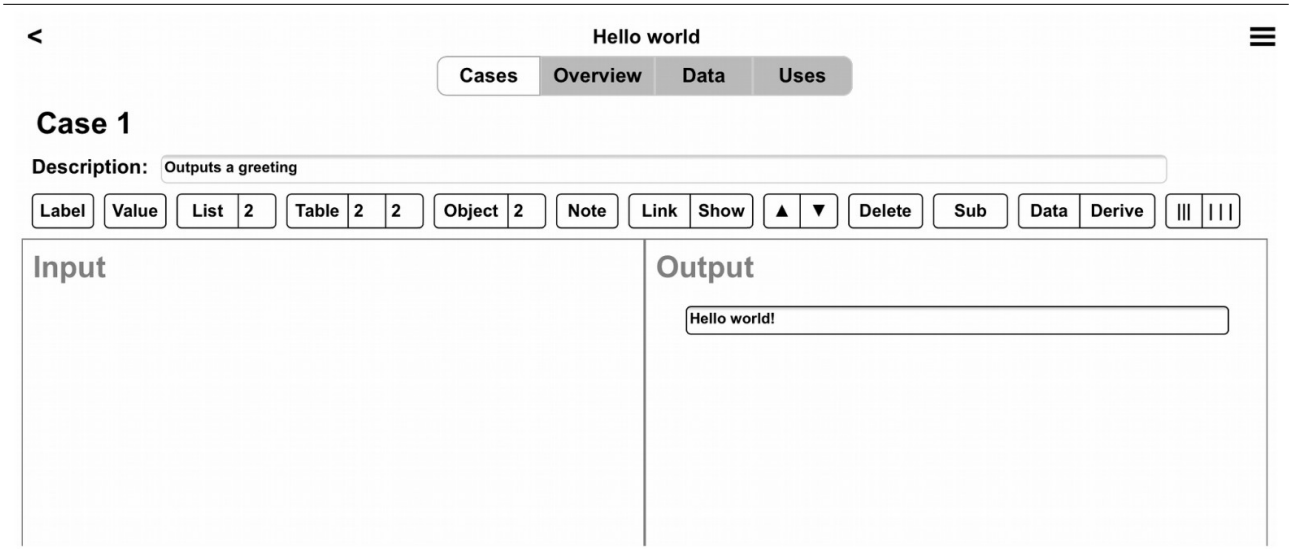

Figure 7 – Case screen

The current program name appears at the top with the back button top-left and the case menu top right. Below the program name are four tabs: Cases; Overview; Data; and Uses.

The case number ('Case 1' in Fig. 7) is a menu that can be used to select any existing case. The description field can be used to provide a comment about the current case.

Messages are displayed in the area to the right of the case selector.

6.3 Case Screen Toolbar

The case screen toolbar buttons are described in the following table.

Toolbar item	Function
Label	Create a label element in the currently selected column.
Value	Create a value element in the currently selected column.
List	Create a list element in the currently selected column. The associated number specifies the size of the list. Size can also be modified after creation by selecting the element and changing the number.
Table	Create a table element in the currently selected column. The two associated numbers specify the size of the table in terms of rows and columns. Size can also be modified after creation by selecting the element and changing the numbers.
Object	Create an object element in the currently selected column. The associated number specifies the number of key-value pairs (entries) in the object. Size can also be modified after creation by selecting the element and changing the number.
Note	Create a comment element in the currently selected column.
Link	Create a dependency between selected elements. Dependencies can also be removed by selecting the relevant elements and pressing 'Link' again. Note that dependencies cannot be created between elements in the same column. Also dependencies cannot be created to or from comments or labels.
Show / Hide	Button toggles between Show and Hide. Show highlights all dependencies by increasing their opacity. Hide fades all dependencies by decreasing their opacity.
Up	Move the selected element up within its column. If a column is selected then move all elements within that column up.
Down	Move the selected element down within its column. If a column is selected then move all elements within that column down.
Delete	Delete the selected elements. A prompt dialogue asks the user to confirm deletion. Any dependencies in which the deleted elements play the role of source or target are also deleted.
Sub	Create a subsidiary test case if the selected element is not already a subsidiary. Note that subsidiary test cases cannot be created for data elements in the data or input columns. Also a subsidiary cannot be created in the output column within any subsidiary. If the selected element is already a subsidiary then the Sub button is used to navigate into it.
Data	Show the data column or hides it if it is empty.
Derive	Display the next empty derive column or hides it if it is empty.
Zoom in ('|||')	Decrease the width of all columns.
Zoom out ('| | |')	Increase the width of all columns.

6.4 Case Screen Menu

The case screen menu has the following functions:

Menu item	Function
New case	Create and edit a new empty case diagram
Clone case	Create a new case diagram that is a structural copy of the current diagram. The values are not copied with the exception of the data column values.
Delete case	Delete the current case diagram. A prompt dialogue asks for confirmation.
Save	Save the entire program
Revert	Discard all changes and reload the last saved copy of the program. A prompt dialogue asks for confirmation.
Compile	Compile the current program.
Listing	Show the listing view of the current program.

6.5 Data Column Editing

The structure and values of the data column elements are automatically synchronised across all cases. This means that any changes to the data column in any case within a given program are applied to every data column in all cases. The reason for this is that there is really just one data column that is shared between all cases. This is displayed in every case diagram so that any dependencies that are required can be created.

6.6 Compilation

During compilation the colour of data elements and dependencies will change depending on the current state. While an element is being compiled it will turn gold. If compilation associated with an element is successful then it will be coloured green. Otherwise if compilation failed then it will be coloured red. Generally compilation proceeds from left to right across the diagram except that subsidiary test cases are compiled first. Also the compiler will switch between different case diagrams depending on whether later cases contain data elements with different identities.

When compilation is complete a message will be displayed indicating success or failure. If compilation should fail then the source of the problem will be apparent as the left most data element that is coloured red.

6.7 Automatic Saving

In order to provide a seamless user experience programs are saved automatically when navigating back to the program list. Automatic saving also occurs when navigating to or from subsidiary test cases.

6.7 Overview Screen

The overview screen (Fig. 8) provides a tabular view of all data elements across all test cases. It does not show dependencies but it does display the numeric data element identifiers. Data element values cannot be edited on the overview screen but top level data elements can be selected. That is a value or complete list element can be selected but a list element, table row or table cell cannot be selected. Note that data elements are displayed with fewer borders on this screen.

Figure 8 – Overview screen

Aside from providing a useful perspective across all test cases, the overview screen can also be used to manage data element identity. Two data elements on different cases share the same identity if they are in the same column. On the other hand if two data elements on different cases are in different columns then they have different identities. For a variety of reasons either of these situations may or may not correspond to what the user intended.

Two buttons are used to manage data element identity. If the user needs to make two elements have different identities then the procedure is to select one of them and press the split button. This will cause the selected element to assume a new unique identifier. It will also move to a new column within the overview.

Two elements can also be made to have the same identity. To do this the user must note the identifier of the first element. The second element is then selected. Pressing the merge button will display a drop down menu with all of the current identifiers. The user then selects the same identifier as the first element. This causes the selected element to assume the same identifier and it will move to the same column as the first element. Note that it is not permitted to create two elements in the same test case with the same identifier in this way.

6.8 Data Screen

The data screen is used to provide a different version of static data for runtime. An example of this is shown in Figure 9 where the test data includes a couple of English diphthongs while the runtime data provides all of them.

It is often simpler and faster to use dummy data within test cases. For example the test cases for a spell checker program could be defined using a dictionary that contains very few words. At runtime a complete dictionary could be substituted without any need to change the code.

The data screen has a dropdown menu that allows the user to select whether test data or runtime data should be used at runtime. The data screen also supports resizing of any composite runtime data elements.

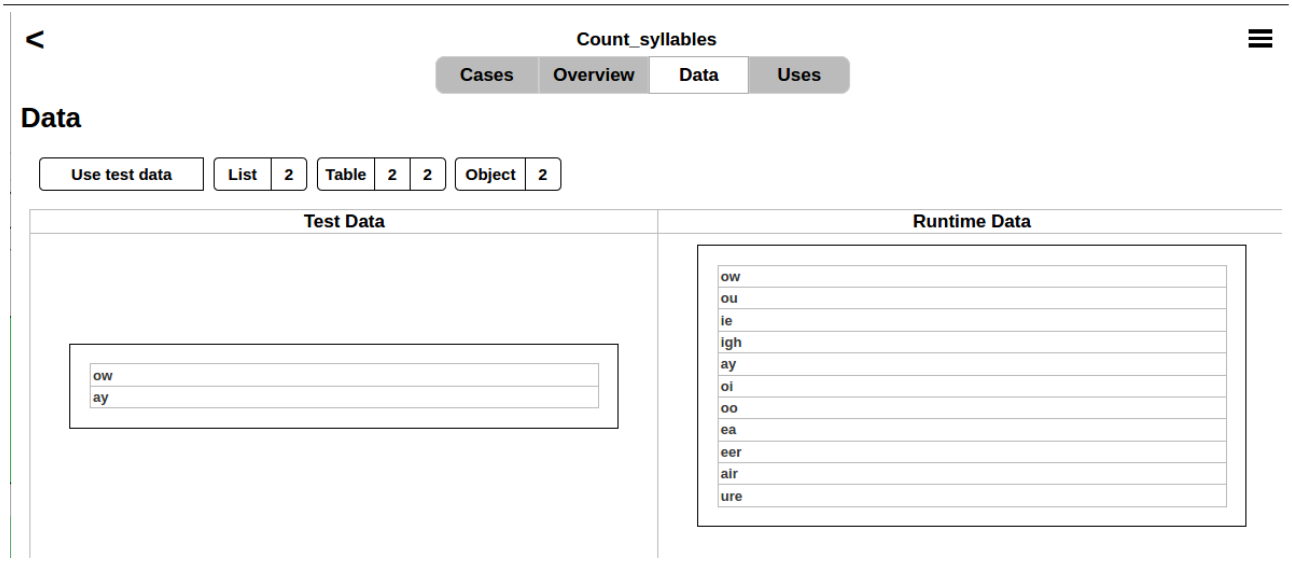

Figure 9 – Data screen

6.9 Uses Screen

The uses screen is used to manage the list of programs that the Zoea compiler should try to incorporate into the current program. Figure 10 shows the layout of the uses screen. On the left is a list of all current programs. On the right is a list of programs that have been selected. Programs are moved between these two lists by clicking on them.

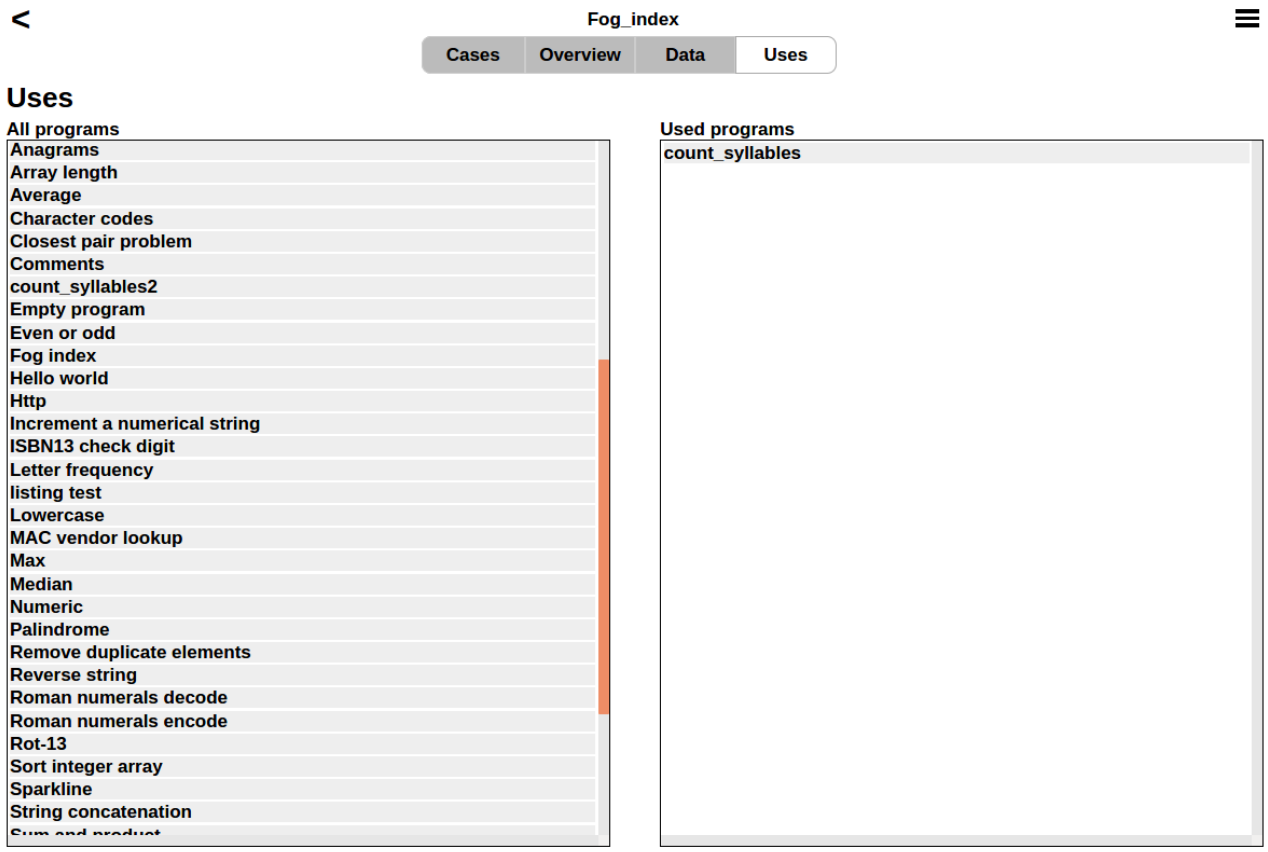

Figure 10 – Uses screen

6.10 Run Screen

The run screen allows a successfully compiled program to be executed. The input and output fields specified in the test cases are displayed. Composite input data elements can also be resized if necessary. Figure 11 shows the layout of the run screen.

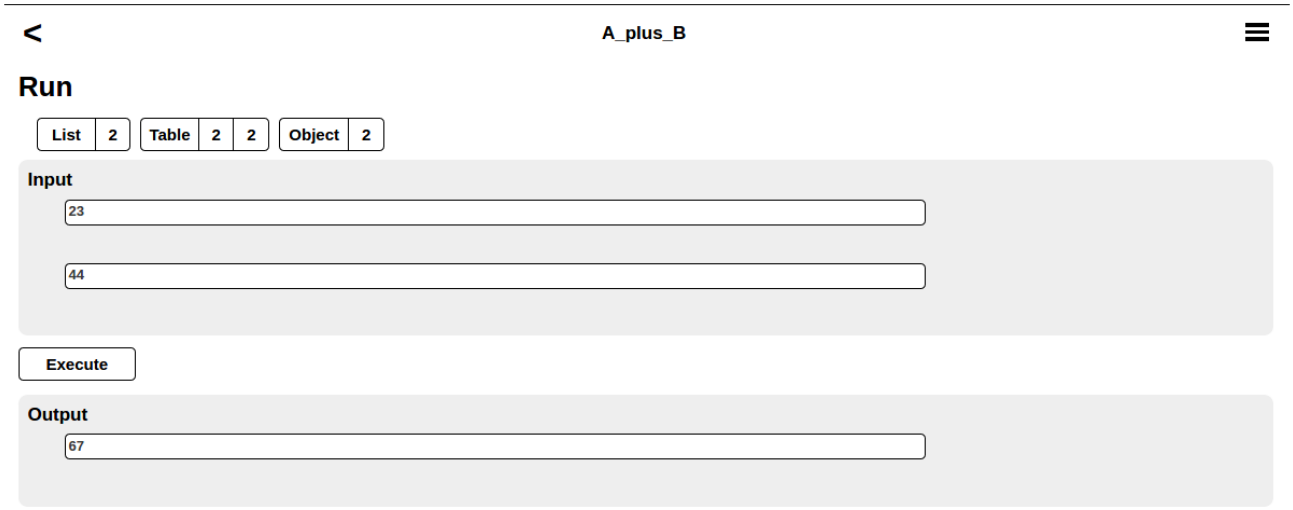

Figure 11 – Run screen

6.11 Listing Screen

The listing screen provides a printer friendly version of the current program showing all case diagrams on a single page. An example of listing screen output is provided in Figure 4.

7. Acknowledgements