Contents

Revision History

Date	Version	Description	Author(s)
30.10.2018	0.1	Initializing document structure	Reto Schönenberger
31.10.2018	0.2	Adding classes	Michael Blum
02.11.2018	0.3	Revision of classes	Yannick Trunz
07.11.2018	0.4	Adding tests	Reto Schönenberger
08.11.2018	0.5	Final Revision	Yannick Trunz/ Michael Blum

1 Public API Overview

1.1 Public classes to be tested

APPLICATION
TOPO_SORT_OBJECT
CONSTRAINT
ELEMENT

1.2 Public routines to be tested

add_element(elem: ELEMENT)
remove_element(elem: ELEMENT)
add_multiple_elements(list: ARRAYED_LIST)
remove_multiple_elements(list: ARRAYED_LIST)
add_constraint(cons: CONSTRAINT
remove_constraint(cons: CONSTRAINT)
Add_multiple_constraints(list: ARRAYED_LIST)
remove_multiple_constraints(list: ARRAYED_LIST
sort_topologically()
show_all_constraints()
show_all_elements()
display_graph()

2 Test Suite Description

Description of all test cases using the following format:

Test ID	ID of test
Requirements under test	ID(s) of the requirement(s) under test
Routines under test	Name(s) of the routine(s) under test
Description	Description of what is tested
Set-up	Operations before executing test (preparations)
Tear-Down	Operations after executing test (clean-ups)
Test data	Data used while executing test
Oracle	Pass/fail criteria

2.1 Test cases

2.1.1 Unit Tests

Test ID	3.1.001_01
Requirements under test	3.1.001
Routines under test	add_element(elem: ELEMENT)
Description	The user shall be able to add a new element {a} to the list of elements.
Set-up	None
Tear-Down	Remove element {a} from ARRAYED_LIST
Test data	ARRAYED_LIST
Oracle	- Element {a} is in ARRAYED_LIST - Success message displayed

Test ID	3.1.001_02
Requirements under test	3.1.001
Routines under test	add_element(elem: ELEMENT)
Description	If the element already exists in the list of elements, there shall be an error message. Add element {a}
Set-up	Add element {a} to ARRAYED_LIST
Tear-Down	Remove element {a} from ARRAYED_LIST
Test data	ARRAYED_LIST
Oracle	- Element {a} is not added again - Error message: element already exists

Test ID	3.1.001_03
Requirements under test	3.1.001
Routines under test	add_element(elem: ELEMENT)
Description	In the case of an empty input, there shall be an error message.
Set-up	None
Tear-Down	None
Test data	None
Oracle	- Error message: empty input

Test ID	3.1.002_01
Requirements under test	3.1.002
Routines under test	add_constraint(cons: CONSTRAINT)
Description	Add a new constraint in form of (a, b) to the list of constraints.
Set-up	Add elements {a, b}
Tear-Down	- Remove constraint - Remove elements
Test data	ARRAYED_LIST
Oracle	- Constraint is added to list of constraints - Success message

Test ID	3.1.002_02
Requirements under test	3.1.002
Routines under test	add_constraint(cons: CONSTRAINT)
Description	The two elements can not be the same. Add constraint (a, a) to the list of constraints.
Set-up	Add element {a}
Tear-Down	Remove element {a}
Test data	ARRAYED_LIST
Oracle	- Constraint is not added to list of constraints - Error message: same element

Test ID	3.1.002_03
Requirements under test	3.1.002
Routines under test	add_constraint(cons: CONSTRAINT)
Description	Both elements must exist in the list of elements. Add constraint (a, b) to the list of constraints.
Set-up	Add element {a}
Tear-Down	Remove element {a}
Test data	ARRAYED_LIST
Oracle	- The constraint is not added to the list of constraints - Error message: element not in list of elements

Test ID	3.1.002_04
Requirements under test	3.1.002
Routines under test	add_constraint(cons: CONSTRAINT)
Description	If the constraint already is in the list of constraints, there shall be an error message. Add constraint (a,b)
Set-up	- Add elements {a, b} - Add constraint (a, b)
Tear-Down	- remove constraint (a, b) - remove elements {a, b}
Test data	ARRAYED_LIST
Oracle	- constraint is not added to list of constraints - Error message: constraint already exists

Test ID	3.1.002_05
Requirements under test	3.1.002
Routines under test	add_constraint(cons: CONSTRAINT)
Description	If the constraint (a, b, c) has an invalid format, there shall be an error message.
Set-up	- Add elements {a, b, c}
Tear-Down	- remove elements {a, b, c}
Test data	ARRAYED_LIST
Oracle	- constraint is not added to list of constraints - Error message: Invalid input

Test ID	3.1.003_01
Requirements under test	3.1.003
Routines under test	remove_element(elem: ELEMENT)
Description	Remove an element {a} from the list of elements.
Set-up	Add element {a}
Tear-Down	None
Test data	ARRAYED_LIST
Oracle	- Element {a} is removed from the list of elements - Success message

Test ID	3.1.003_02
Requirements under test	3.1.003
Routines under test	remove_element(elem: ELEMENT)
Description	If element {a} is not in the list of elements, there should be an error meassage.
Set-up	None
Tear-Down	None
Test data	ARRAYED_LIST
Oracle	- Error message: element does not exist

Test ID	3.1.004_01
Requirements under test	3.1.004
Routines under test	remove_constraint(cons: constraint)
Description	Remove a constraint (a, b) from the list of constraints.
Set-up	- Add elements {a, b} - Add constraint (a, b)
Tear-Down	Remove elements {a, b}
Test data	ARRAYED_LIST
Oracle	- Constraint (a, b) is removed from the list of constraints - Success message

Test ID	3.1.004_01
Requirements under test	3.1.004
Routines under test	remove_constraint(cons: constraint)
Description	If the constraint (a, b) is not in the list of constraints, there shall be an error message.
Set-up	None
Tear-Down	None
Test data	ARRAYED_LIST
Oracle	- Error message: Constraint does not exist

Test ID	3.1.005_01
Requirements under test	3.1.005
Routines under test	add_multiple_elements(list: arrayed_list)
Description	The user shall be able to add multiple elements to the list of elements by passing a list X. The system will add every single element from list X to list of elements, given that the element is a valid input.
Set-up	None
Tear-Down	Remove list X of elements
Test data	ARRAYED_LIST
Oracle	- All valid elements of the list X are added to the list of elements - Success message

Test ID	3.1.006_01
Requirements under test	3.1.006
Routines under test	Add_multiple_constraints(list: ARRAYED_LIST)
Description	The user shall be able to add multiple constraints to the list of constraints by passing a list Y. The system will add every single constraint from list Y to the list of constraints, given that the constraint is a valid input.
Set-up	Add a list X of elements used in the list Y of constraints
Tear-Down	- Remove list Y of constraints - Remove list X of elements
Test data	ARRAYED_LIST
Oracle	- All valid constraints of list Y are added to the list of elements. - Success message

Test ID	3.1.007_01
Requirements under test	3.1.007
Routines under test	remove_multiple_elements(list: ARRAYED_LIST)
Description	The user shall be able to remove multiple elements from the list of elements by passing a list X. The system will remove every single element in list X from the list of elements, given that the element is a valid input.
Set-up	Add list X of elements
Tear-Down	None
Test data	ARRAYED_LIST
Oracle	- All valid elements of list X are removed from the list of elements - Success message

Test ID	3.1.008_01
Requirements under test	3.1.008
Routines under test	remove_multiple_constraints(list: ARRAYED_LIST)
Description	The user shall be able to remove multiple constraints from the list of constraints by passing a list Y. The system will remove every single constraint in list Y from the list of constraints, given that the constraint is a valid input.
Set-up	- Add a list X of elements used in the list Y of constraints - Add list Y of constraints
Tear-Down	Remove list X of elements
Test data	ARRAYED_LIST
Oracle	- All valid constraints of list Y are removed from the list of constraints - Success message

Test ID	3.1.009_01
Requirements under test	3.1.009
Routines under test	display_graph()
Description	Shows all elements in the list of elements and all constraints in the list of constraints according to these elements.
Set-up	- Add element {a, b} - Add constraint (a, b)
Tear-Down	- Remove element {a, b} - Remove constraint (a, b)
Test data	ARRAYED_LIST
Oracle	Shows a graphical representation of the elements {a, b} and the constraint (a, b).

Test ID	3.1.009_02
Requirements under test	3.1.009
Routines under test	display_graph()
Description	If the list of elements is empty, there shall be a message.
Set-up	None
Tear-Down	None
Test data	ARRAYED_LIST
Oracle	Message: empty list of elements

Test ID	3.1.010_01
Requirements under test	3.1.010
Routines under test	sort_topologically()
Description	The system shall provide a topological order using a topological sort algorithm.
Set-up	- Add elements {a, b, c} - Add constraints {(b, a), (b, c)}
Tear-Down	- Remove all elements - Remove all constraints
Test data	ARRAYED_LIST
Oracle	- Sorted List - Confirmation message

Test ID	3.1.010_02
Requirements under test	3.1.014
Routines under test	sort_topologically()
Description	If there are multiple possible for a certain input, the system shall always produce the same result.
Set-up A	- Add elements {a, b} - Add constraints {(b, a), (a, b)} - Do topological order
Set up B	- Add elements {a, b} - Add constraints {(b, a), (a, b)} - Do topological order
Tear-Down	- Remove all elements - Remove all constraints
Test data	ARRAYED_LIST
Oracle	The topological order of Set Up A equals the topological order of Set Up B.

Test ID	3.1.010_03
Requirements under test	3.1.010
Routines under test	sort_topologically()
Description	If the list of elements is empty, there shall be a message.
Set-up	None
Tear-Down	None
Test data	ARRAYED_LIST
Oracle	Message: empty list

Test ID	3.1.011_01
Requirements under test	3.1.011
Routines under test	sort_topologically()
Description	If there are any cycles in the topological order the system shall document them.
Set-up	- Add elements {a, b, c} - Add constraints {(b, a), (b, c), (c, b)} - Do topological order
Tear-Down	- Remove all elements - Remove all constraints
Test data	ARRAYED_LIST
Oracle	A documentation of the cycle produced by the constraints {(b, c), (c, b)}.

Test ID	3.1.012_01
Requirements under test	3.1.012
Routines under test	sort_topologically()
Description	If there are cycles, there shall be an output just regarding the non-cycle part of the input.
Set-up	- Add elements {a, b, c} - Add constraints {(b, a), (b, c), (c, b)} - Do topological order
Tear-Down	- Remove all elements - Remove all constraints
Test data	ARRAYED_LIST
Oracle	- Sorted List of the elements {a, b} - Message: cycles detected

Test ID	3.1.012_02
Requirements under test	3.1.012
Routines under test	sort_topologically()
Description	If there is no non-cycle part, there shall be a message.
Set-up	- Add elements {a, b} - Add constraints {(b, a), (a, b)} - Do topological order
Tear-Down	- Remove all elements - Remove all constraints
Test data	ARRAYED_LIST
Oracle	Message: Order failed due no non-cycle part

Test ID	3.1.013_01
Requirements under test	3.1.013
Routines under test	show_all_constraints()
Description	The system should allow the user to display all existing constraints.
Set-up	- Add elements {a, b, c} - Add constraints {(b, a), (b, c)}
Tear-Down	- Remove all elements - Remove all constraints
Test data	ARRAYED_LIST
Oracle	Displays constraints {(b, a), (b, c)}

Test ID	3.1.013_02
Requirements under test	3.1.013
Routines under test	show_all_constraints()
Description	If the list of constraints is empty, there shall be a message.
Set-up	None
Tear-Down	None
Test data	ARRAYED_LIST
Oracle	Message: list is empty

Test ID	3.1.013_02
Requirements under test	3.1.013
Routines under test	show_all_elements()
Description	The system should allow the user to display all existing elements.
Set-up	Add elements {a, b, c}
Tear-Down	Remove all elements
Test data	ARRAYED_LIST
Oracle	Displays elements {a, b, c}

Test ID	3.1.013_02
Requirements under test	3.1.013
Routines under test	show_all_elements()
Description	If the list of elements is empty, there shall be a message.
Set-up	None
Tear-Down	None
Test data	ARRAYED_LIST
Oracle	Message: list is empty

2.1.2 End-to-End Tests

Test ID	3.2.002_01
Requirements under test	3.2.002
Routines under test	- add_element(elem: ELEMENT) - add_multiple_elements(list: ARRAYED_LIST) - add_constraint(cons: CONSTRAINT) - Add_multiple_constraints(list: ARRAYED_LIST) - sort_topologically()
Description	The system takes the input from the Rosetta Code website.
Set-up	None
Tear-Down	- Remove all elements - Remove all constraints
Test data	ARRAYED_LIST
Oracle	All elements ordered on the Rosetta Code website.

Test ID	3.2.003_01
Requirements under test	3.1.003
Routines under test	- add_element(elem: ELEMENT) - add_multiple_elements(list: ARRAYED_LIST) - add_constraint(cons: CONSTRAINT) - Add_multiple_constraints(list: ARRAYED_LIST) - sort_topologically()
Description	The system takes as input 4 elements and 10 constraints. This should show how the system works with a relatively small number of elements and constraints.
Set-up	None
Tear-Down	- Remove all elements - Remove all constraints
Test data	ARRAYED_LIST
Oracle	Elements ordered in topological order.

Test ID	3.2.004_01
Requirements under test	3.1.004
Routines under test	- add_element(elem: ELEMENT) - add_multiple_elements(list: ARRAYED_LIST) - add_constraint(cons: CONSTRAINT) - Add_multiple_constraints(list: ARRAYED_LIST) - sort_topologically()
Description	The system takes as input 200 elements and 1000 constraints. This should show how the system works with a medium number of elements and constraints.
Set-up	None
Tear-Down	- Remove all elements - Remove all constraints
Test data	ARRAYED_LIST
Oracle	Elements ordered in topological order.

Test ID	3.2.005_01
Requirements under test	3.1.005
Routines under test	- add_element(elem: ELEMENT) - add_multiple_elements(list: ARRAYED_LIST) - add_constraint(cons: CONSTRAINT) - Add_multiple_constraints(list: ARRAYED_LIST) - sort_topologically()
Description	The system takes as input 2000 elements and 100'000 constraints. This should show how the system works with a relatively large number of elements and constraints.
Set-up	None
Tear-Down	- Remove all elements - Remove all constraints
Test data	ARRAYED_LIST
Oracle	Elements ordered in topological order.

Test ID	3.2.006_01
Requirements under test	3.1.006
Routines under test	- add_element(elem: ELEMENT) - add_multiple_elements(list: ARRAYED_LIST) - add_constraint(cons: CONSTRAINT) - Add_multiple_constraints(list: ARRAYED_LIST) - sort_topologically()
Description	The inputs are different lines in form of: • $elem_0$: $elem_1$, $elem_2$, $elem_n$ This line means that $elem_0$ depends on $elem_1$, $elem_2$, $elem_n$. The output should contain all elements in the input in such an order that all the constraints are considered and the output is in correct topological order.
Set-up	None
Tear-Down	- Remove all elements - Remove all constraints
Test data	ARRAYED_LIST
Oracle	Elements ordered in topological order.

3 Expected Coverage

We expect a unit test coverage of 70 % for our system. Since we are just getting started testing and getting used to write good tests, we can not make precise estimations about the expected coverage yet.

4 Benchmarks

The implementation of the topological sort shall have a linear running time *O(n)*.