ST: Lab Assignment 4

Elias El Khaldi Ahanach, Dylan Bartels, Wojciech Czabanski, Quinten Heijn

October 1, 2017

Assignment 1

Time required: 4 hours

1. "The Comprehension Principle. A set is a collection into a whole of definite, distinct objects of our intuition or of our thought." What is meant by definite, distinct objects of our intuition?
2. flexibility of the abstraction notation: $x \in A \mid P(x)$
3. "In most cases you will have that a != a". Why not all cases? What is a counter example?
4. In example 4.6, why does F not belong to F in the example?
5. Why does 'halts' take 2 arguments: 'funny' and 'funny'? What are the types of arguments in this reasoning?
6. How to show that a set containing an empty set (0) and a set containing a set, containing an empty set (0) are different?
7. How to determine how many elements a power set has of a set with n elements?
8. Would it be possible to prove that if every number in the domain of input for the program, after transforming using the following formula: n1 = 3n0 + 1, converges, at some point to a power of 2 then the program can halt?

Assignment 2

Time required: 4 hours 30 min

See code for implementation

Assignment 3

Time required: 3 hours

Testable properties setIntersection:

- Commutativity = A (Intersect) B = B (Intersect) A (p.130)

setUnion:

- Commutativity = A (union) B = B (union) A (p.130)
- Contains itself = A (union) B should contain every element from A (p.130)

setDifference:

- Difference = A - B (p.127)

Assignment 4

Time required: 3 hours

1. How to show that there are only 13 transitive relations on 0, 1?

Assignment 5

Time required: 55 min

See code for implementation

Assignment 6

Time required: 30 minutes

See code for implementation

Assignment 7

Time required: 90 minutes

If you interpret a set of relations as a graph, n is the number of individual nodes in that graph. In the code, we implemented a function that lists all the nodes, and a function that calculates the number of nodes. The number of nodes will be of interest when defining properties for the trClos and symClos functions.

For the Symmetric Closure function (symClos) we defined the following properties:

- Length constraint: The length of a symClos of x can at max be double the length of x, and at least the length of x
- After applying symClos once to x, it should always remain the same no matter how many more times you apply it
- All relations in x should be in (symClos x)
- No new nodes should be included

For the Transitive Closure function (traClos) we defined the following properties:

- Length constraint: The length of trClos x should be at least as big as length x and at most n squared (n = number of nodes)
- After applying trClos once to x, it should always remain the same no matter how many more times you apply it
- All relations in x should be in (trClos x)
- No new nodes should be included

For both properties, we implemented a quickCheck test method. The way the test and the properties were implemented can be seen in the code. Example outputs are given below:

```
*Assignment7> quickCheck testTra
+++ OK, passed 100 tests.
*Assignment7> quickCheck testSym
+++ OK, passed 100 tests.
```

Assignment 8

Time required: 15 minutes

After some time brain storming we were able to come up with a counter example:
take *x = [(1,2)]*
then: *trClos (symClos x) = [(1,1),(1,2),(2,1),(2,2)]*
but: *symClos (trClos x) = [(1,2),(2,1)]*

Bonus Assignment 9

Time required: 90 minutes

Properties to test both implementations are:

- read (show statement) == statement
- show (read textStatement) == textStatement

Extra Bonus Assignment 10

Time required: 40

See code for implementation