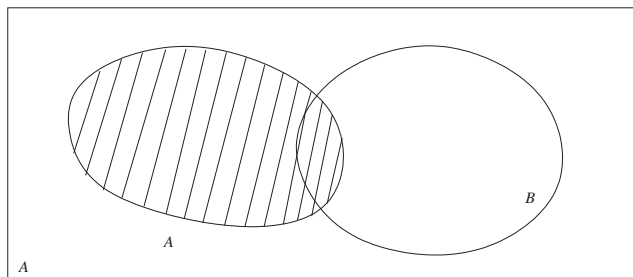


Solutions to second in-class test

Question 1

(a)



(b) The result is just the set A .

(c) The set $B \setminus A$ only contains elements that don't belong to A . Therefore, if we take away from A the set $B \setminus A$, nothing happens and the result is just A .

Question 2

(a) A set A is countable if its elements can be aligned one-to-one to the elements of \mathbb{N} . (More technically, if there is a bijective function between A and \mathbb{N} .)

(b) Bookwork (Handout 10, Box 78).

(c) The consequence is that not all functions from \mathbb{N} to \mathbb{N} can be implemented in Java. Some are *non-computable*.

Question 3

(a) Definedness: For every $a \in \mathbb{R}$ we must have some b such that $(a, b) \in F$. This is the case, because the pair $(a, 2a)$ belongs to F .

Single-valuedness: By the definition of F , the pair $(a, 2a)$ is the only one in F with first component a .

(b) F is injective, because if we assume $F(a) = F(a')$ then this means $2a = 2a'$ and from this it follows that $a = a'$.

F is also surjective, because for a given target element b we can choose the input element $a = b/2$ and get $F(b/2) = 2(b/2) = b$.

A function that is injective and surjective is by definition bijective.

(c) There are four differences (I required you to list at least two for full marks):

- The datatype `float` does not contain all real numbers, but only a finite subset.
- On the other hand, `float` contains the elements $+0$, -0 , $+\infty$, $-\infty$, and `NaN`, which are not real numbers.
- The implementation is not injective as for very large float's the result will be $+\infty$ and this is also the result when the input is $+\infty$.
- The implementation is not surjective. The smallest positive float can not be obtained as twice a smaller number.

As a consequence, the implemented function `javaF` can not be inverted whereas the bijective function F can be.

Question 4

- (a) The relation is reflexive because a student's mark is equal to its own mark. It is transitive because if a has a mark higher or equal to that of student b , and b has a mark higher or equal to that of student c , then the mark of a is also higher or equal to that of c . But the relation is not anti-symmetric because two different students can have exactly the same mark average.
- (b) This relation is reflexive and symmetric, but it is not transitive: The difference between a and b could be 0.8% and that between b and c could be 0.7% but then the difference between a and c would be 1.5%, more than is allowed for two related students.
- (c) It illustrates that degree classifications do *not* group students of similar ability together. In fact, two students could be 9% away from each other and be in the same class, and on the other hand, two other students could be as close as 0.5%, yet get different degree classifications. This is clearly a nonsensical (and cruel) system.

Question 5

(a) First derivation:

- `true` is in `BoolExp` because of Rule 1a.
- Because `true` is in `BoolExp`, then so is `NOT true` by Rule 2.
- `false` is in `BoolExp` because of Rule 1b.
- Because `NOT true` and `false` are in `BoolExp`, so is `NOT true AND false` by Rule 3.

Second derivation

- `true` is in `BoolExp` because of Rule 1a.
- `false` is in `BoolExp` because of Rule 1b.
- Because `true` and `false` are in `BoolExp`, so is `true AND false` by Rule 3.
- Because `true AND false` is in `BoolExp`, so is `NOT true AND false` by Rule 2.

Using additional brackets, the two derivations lead to the expressions `(NOT true) AND false` and `NOT (true AND false)`.

- (b) Since the inductive definition of ev contains a clause for every clause in the definition of `BoolExp`, the relation is guaranteed to be defined for all elements of `BoolExp`. The problem is single-valuedness. Following the two derivations for `NOT true AND false` found in (a), we get the two conflicting pairs `(NOT true AND false,0)` and `(NOT true AND false,1)` in ev .
- (c) We need an inductive definition that allows only one derivation for each expression. Here is an attempt (written as a grammar for brevity and readability):

$$\begin{array}{lcl} S & ::= & A \mid C \\ A & ::= & \text{true} \mid \text{false} \mid (C) \mid \text{NOT } A \\ C & ::= & A \mid A \text{ AND } C \end{array}$$

The idea is to distinguish between “atomic” and “composite” expressions and to allow negation only to be applied to atomic ones.