Contents

My Report!

First year review report

Zhili Tian

Supervised by Prof. Thorsten Altenkich

& Prof. Ulrik Buchholtz

Funtional Programming Lab

School of Computer Science

University of Nottingham

May 22, 2025

Abstract

Giving a short overview of the work in your project.[1]

```
open import Relation.Binary.PropositionalEquality
```

0.1 Natural Numbers

First, we define the type of natural numbers inductively:

```
data ℕ : Set where
  zero : ℕ
  suc : ℕ → ℕ
```

Here, ℕ is the type of natural numbers, with two constructors:

- zero represents 0.
- suc represents the successor function (i.e., $n + 1$).

0.2 Addition

Next, we define addition recursively:

```
_+_ : ℕ → ℕ → ℕ
zero + n = n
suc m + n = suc (m + n)
```

This definition states:

- 0 + n = n (base case).
- (m + 1) + n = (m + n) + 1 (recursive case).

0.3 A Simple Proof

We now prove that 2 + 2 = 4. First, we define the numbers:

```
two : ℕ
two = suc (suc zero)

four : ℕ
four = suc (suc (suc (suc zero)))
```

Now, the proof reduces by computation:

```
proof : two + two ≡ four
proof = refl
```

Since Agda's definitional equality handles reduction, refl suffices.

0.4 Conclusion

This example shows how Agda and LaTeX can be combined for formal proofs in papers. The full output is rendered with syntax highlighting.

Bibliography

[1] ABBOTT, M., ALTENKIRCH, T., AND GHANI, N. Containers: Constructing strictly positive types. *Theoretical Computer Science 342*, 1 (2005), 3–27. Applied Semantics: Selected Topics.