Hi, and welcome to my presentation titled 'Constructing simple and mutual inductive types'. My name is Stefania and I was supervised by Professor Thorsten Altenkirch.

2)
Our dissertation takes place within Martin-Lof Type Theory, so we begin by briefly describing what this is about.
[CLICK] It was designed by mathematician and philosopher Per Martin-Lof to be used for programming as well as a foundation for constructive mathematics, which is a variant of mathematics that requires any existence proof to contain a witness.
[CLICK] Its type constructors were built to follow a one-to-one correspondence with logical connectives. We can see this correspondence in the following table showing logical connectives and their type theoretic equivalents.
[CLICK] If we look at the first line, on the logic side we have 'A implies B' where A and B are propositions. Its translation is the function of type 'A to B', meaning that propositions A and B were translated into the types A and B, hence why we call this paradigm propositions as types. We have a similar translation for the other predicate logic connectives.

3)
Our formalisation takes place in Agda, which is also closely related to Martin-Lof Type Theory. So what is Agda?
[CLICK] It implements dependent types and allows the user to do computer-assisted mathematical proofs.
[CLICK] Code in Agda is NOT run but type checked, the way to evaluate our results is to ensure that our code compiles in Agda.
[CLICK] Makes it perfect for our study.
[CLICK] In order to get a taste of what Agda code looks like, we have an example of a logical proposition written out as a type and proved in Agda. This is precisely the law of the contrapositive. We translated the logical statement into a type and then inhabited the type, or gave an element of the type. The fact that it type checks means the proof is correct.

4&5)
Our project focusses on inductive types.
[CLICK] Perhaps the best way to explain this is to show some examples.
[CLICK] A simple inductive type. N can be constructed in 2 ways.
[CLICK] A mutual inductive type. 2 sorts where their constructors refer to the other sort.
Listing the constructors does not give a full description of inductive types, we also need their eliminator.
[CLICK] We want to define a function f : (n : N) → M n. To do this, we need to define how M behaves on N's constructors zero and suc.
This corresponds to the induction principle for natural numbers.
If a predicate is conserved by all constructors, then it holds for all objects in the inductive type.

6)
An important inductive type that we also look into is the W-type.
[CLICK] We are interested in the W-type because by assuming the existence of this single inductive type, we can derive all inductive types. It is the type of well-founded trees and we can actually think of inductive types as trees. To illustrate this we have the following example.
[CLICK]

7)
The motivations for our dissertation are the following.

[CLICK] There has been work similar to what we do here for more general types, in which our results are said to be analogous but are not formalised. That is what we aim to do here.
[CLICK] We do not want seemingly simple results to be taken for granted. We want to be able to prove every assumption from our axioms. Formalisation is the key to rigour. We want to be rigorous so that our systems are reliable. This leads to our next point.
[CLICK] The reliability of a theorem prover or software verification tools increases significantly if its correctness depends on a small amount of code, and our aim is to increase the reliability.
Having explained our motivations, we now list our aims and what we did to achieve them.

8)
We want to define a framework in which we can specify inductive types. We do this by associating a signature to any inductive type.

9)
Next, for every signature, we want to construct the eliminator because as we said earlier, listing the constructors of an inductive type is not enough.
[CLICK] We have achieved this using induction on the structure of the theory of signatures, and this gives a complete specification of inductive types because..
[CLICK]

10)
Finally, we looked into indexed W-types, which are a variant of the W-types we discussed earlier, but which have been shown to be reducible to W-types.
[CLICK] The last two bullet points we started working on but unfortunately did not have time to finish them and hence did not complete our reduction, but we gave ideas in our dissertation of how to move forward to conclude the reduction from simple and mutual inductive types to W-types.