Agda lecture 1 Q&A (Solved)

HoTTEST Summer School 2022

The HoTTEST TAs

6th July 2022

What does the λ mean?

$\lambda x \to y$ means the function that takes x as an argument and returns y. A mathematician might write: $x \mapsto y$. In the HoTT lectures it might be written as $\lambda x\,.\,y$.

Why do we need `open import` instead of just `import`?

Writing `import` will bring the module into scope (meaning that everything defined in the imported module is available for use) but we would need to prefix every construction in the imported module with the module name to access it (for example, accessing the function `bar` from the imported module `foo` would require us to write `foo.bar`). "Opening" the module brings the functions into the top level scope (so we could access the function `bar` by simply writing `bar`).

Can any type with a unary and a nullary constructor be used as the builtin natural numbers?

Yep! (where the unary constructor's domain is the type being defined)

Is using `Builtin Natural` more efficient than using `succ (succ ...)`?

Adding the pragma makes the internal representation of naturals more efficient. You can also declare the standard arithmetic operations as builtin in a similar way, which speeds them up.

Does mix-fix notation mean we can't have names with underscores?

That's right. `_` is reserved for mix-fix functions. Underscores are one of the very few characters that can't be in names. (You also can't have names with semicolons, or braces, or parentheses). But you can have names with dashes (i.e. `kebab-case`), or names with brackets (i.e. `this[is-one-variable-name]`), etc.

Can we use ∀ to decorate Π-types for better readability?

Yes. When you use ∀, you can also skip the type and Agda will try to figure it out by itself. For example, these:

```
f : (a : A) → ...
f : ∀ (a : A) → ...
f : ∀ a → ...
```

are all equivalent (assuming it can figure out the type of `a` from the context)

Can you express the normal if-then-else using the dependent one?

Yes, in the case when P is constant the dependent one is the same as the normal one.

What does it mean if you start a line with →?

That's a function type `(x : A) → B x` split over multiple lines. You can choose where you put the → between the types, either at the end of the previous line or at the start of the next line.

That thing about Martin-Löf type theory only allowing $\mathbb{N}$-elimination and no other form of recursion, does that mean that `List-elim` should be definable in terms of $\mathbb{N}$-elim?

It only allows $\mathbb{N}$ elimination for natural numbers! We get other elimination principles for other datypes, like `List-elim` for lists.

What was the input for the $\equiv$ sign?

You can type `\===` to get this. More generally, to find out how to input a specific character, e.g. from the standard library, position the cursor over the character and type `M-x describe-char` or `C-u C-x =`.

If we wouldn't define $\equiv$, then would we be having many non-equivalent copies of natural numbers?

No, in fact, we can show that $\equiv$ coincides with actual equality.