Christian-Albrechts-Universität zu Kiel

Bachelor Thesis

Title: TBD

Niels Bunkenburg

SS 2016

Contents

1 Preliminaries

1.1 Coq

The formalization of Curry programs requires a language that allows us to express the code itself and the propositions we intend to prove. Coq[1] is an interactive proof management system that meets these requirements, thus it will be the main tool used in the following chapters.

TODO: Kommasetzung?

Coq's predefined definitions, contrary to e.g. Haskell's Prelude, are very limited. However, being a functional language, there is a powerful mechanism for defining new data types. A definition of polymorphic lists could look like this:

```
Inductive list {X:Type} : Type :=
| nil  : list
| cons : X -> list -> list.
```

We defined a type named 'list' with two members: the constant nil, which represents an empty list, and a binary constructor cons that takes an element and a list of the same type as arguments. Note that X is a type variable and enclosed in curly brackets, which declares X as an implicit argument. Coq's type inference system deduces the type of list automatically when writing expressions:

```
Check (cons 1 (cons 2 nil)).
(* Evaluates to cons 1 (cons 2 nil) : list *)
Check (cons 1 (cons nil nil)).
(* Error: The term "cons nil nil" has type "@list (@list ?X0)"
while it is expected to have type "@list nat". *)
```

Based on this, we can write a function that determines if a list is empty:

[1] https://coq.inria.fr/