CS 671, Automated Reasoning

Lesson 1: *Why Constructive Type Theory?* January 21, 2001

Bob Constable taught the first 35 Minutes, providing motivation and background.

The formal theory that we will study in this course, is called *Computational Type Theory*.

Like programming languages, computational type theory is a *formal language*, that is a language that follows certain rules wrt. how things may be expressed and where all expressions have a precise, unambiguous meaning. In addition to that, it is also a *computational* language, that is certain expressions can be computed, or evaluated, until we get another, simpler expression, which describes the result, or value, of the computation. Furthermore, it is also a *formal logic*, that is it gives us formal rules for analyzing the meaning of expressions through symbolic manipulations of the formal expressions. Last, but not least, computational type theory is a theory of (data) types, that is it introduces a classification principle, which associates elements denoted by expressions with types, or classes, to which they belong.

When discussing type theory, we therefore have to look at concepts such as *syntax, semantics, evaluation, reasoning rules, types*, and *membership*. These concepts are intertwined in various ways: Evaluation and rules will be expressed in terms of the syntax of type theory, but have to respect its semantics. On the other hand syntactical evaluation and type membership will be one of the fundamental concepts to express our semantics, which makes sure that the way we describe the semantics itself leaves no room for ambiguities.

In addition to the theory, we will also study a system, the NUPRL proof development system, which is capable of integrating a variety of different reasoning techniques that may be used in the verification, synthesis, and optimization of programs.

Roughly, the course will have three major sections

1. In the first few weeks, that is from today until about February 25, we will discuss the core of type theory and how it is used in the NUPRL system for computer-aided, interactive formal reasoning.
2. In the following weeks until Spring break, we will look at techniques for building automated theorem provers on top of that.

   You may then use the Spring break to think about a possible projects that you want to pursue.
3. In the last weeks, we will look at more advanced topics of type theory, which make it possible to formalize complex computational concepts in a natural way, and conclude with the discussion of a few application projects that were built on that basis.

Remarks: