外国語要旨

学位論文題目：Abstracting Control with Dependent Types
氏　　名：Youyou Cong

Dependent types are a powerful tool for ensuring safety. By interacting with the term language, dependent types are able to precisely encode program specifications, guaranteeing the absence of runtime errors and other unexpected behaviors. Meanwhile, control operators have been extensively used to increase convenience. By talking about the surroundings of programs, control operators enable sophisticated manipulation of control flow, yielding a wide range of practical applications.

The two language ingredients are however known to pose various difficulties when mixed up together. Intuitively, the disharmony stems from their contrasting nature: dependent types are used for reasoning purposes and thus must be determined statically, whereas control operators are used to implement dynamic, non-local behaviors. To make their combination meaningful, previous work has imposed a purity restriction on type dependency, that is, types may depend only on effect-free terms.

In this thesis, we build a dependently typed, effectful language called Dellina. Dellina has support for essential features from the mainstream proof assistants, as well as the delimited control operators shift and reset. Similarly to the existing studies, we restrict types to depend only on pure terms, but additionally, we impose two constraints on the type of contexts surrounding effectful terms, in order to cope with the flexibility of the control operators. These restrictions make the resulting language type sound. We also define a selective CPS translation of the language, and prove that the translation preserves typing. Our key observation is that, in a dependently typed setting, selective translations not only yield efficient programs, but simplify the proof of the type preservation property.

Dellina is the first non-toy language where dependent types and control operators co-exist. To demonstrate its utility, we implement a type-safe evaluator that uses shift, reset, and dependent types all in a non-trivial manner. Our result further opens the door to integrating ¥shift and ¥reset into proof assistants. We discuss how we should extend the ``proofs-as-programs'' view to a language with delimited control, and what we can prove with the control operators.