KOOL-TYPED-STATIC

KOOL | Type! – Static Type and Trace Errors (extended) | jul-nov, | Karl Hamill [Cosmic, wdc]

Abstract

This is an OL extension of the Type! KOOL language. It extends the core constructs of Lyon KOOL0 with: new primitive types, new operations, new control flow constructs, and a new type system. The new type system is a static type system. It uses a trace-based process of the type flow of the underlying program, and thus, it is not a type system in the traditional sense. It is a type system that is based on the trace of the program. Accordingly, it offers the following: (1) Type Inference, (2) Type Checking, (3) Type Coercion, (4) Type Casting, (5) Type Tracing, (6) Type Profiling, (7) Type Debugging, (8) Type Optimization, (9) Type Reflection, (10) Type Introspection.

1. Static Type! KOOL is a Statement-type, which can be explicitly used in programs or type variables and attributes, and implicitly in type inference and type checking.

2. There are two new type-related keywords: `type` and `trace`. `type` is used to declare a type, and `trace` is used to trace the type of a variable or attribute.

3. Unlike variables and field types (that is, class type instances), Static type! KOOL does not have a type environment. Static type! KOOL does not have a type environment because it does not need one. Instead, it uses the trace of the program to infer the type of a variable or attribute. Thus, the trace of the program is the type environment. In other words, the trace of the program is the type environment.

4. If an identifier is not found in the local type environment, it will be searched for in the current trace type environment. If it is found in the current trace type environment, the type of the identifier is inferred from the trace. Otherwise, the identifier is not a variable (unless it is a function).

5. Static type! KOOL uses a trace-based type inference system. This system uses the trace of the program to infer the type of a variable or attribute. It uses the trace of the program to type the assignment statement, and then, the trace is used to type all the other statements in the program.

6. Functions are integrated into Static type! KOOL in a unique fashion. Functions are treated like first-class objects in Static type! KOOL. Functions are treated like first-class objects in Static type! KOOL. Functions are treated like objects in Static type! KOOL. Functions are treated like objects in Static type! KOOL. Functions are treated like