Concepts of Programming Language Design

Linear Type Systems

Tom Smeding, Gabriele Keller

January 27, 2025

1. **Linear Types** Linear type systems can be used to model stateful computations in a declarative way. An alternative way, implemented in Haskell, is to do so is by using a monad.

   (a) What could a linear type equivalent of the (in this example monomorphic) `IORef` interface below look like?

   ```
   newIORef   :: Int -> IO (IORef Int)
   readIORef  :: IORef Int -> IO Int
   writeIORef :: IORef Int -> Int -> IO ()
   ```

   > **Solution:** There are different possible solutions. The `IORef` type constructor needs to be linear, but the `Int` type should be a regular type. There is no need anymore to wrap the results in an `IO` type, but the `IORef` has to be threaded through. That means, all three operations have to return an `IORef` as result. Also, either the order of the arguments of `writeIORef` have to be swapped, or the second `->` has to be a linear function type (otherwise the linear reference could be copied via partial application).

   (b) In your opinion, which one is more convienient from the user perspective? Why?

   > **Solution:** The linear functions are still pure functions, so you don't have the problem that every caller will have `IO` monad type. On the other hand, the reference has to be threaded through the computations.

   (c) What is the type of $¡\lambda(\texttt{x} : \texttt{Int}) \to \texttt{get x arr}$ under the environment $\Gamma = \{\texttt{get} : \texttt{Int} \to ¡\texttt{Array} \to ¡\texttt{Pair Float } ¡\texttt{Array}, \texttt{arr} : ¡\texttt{Array}\}$.

   > **Solution:** `Int ¡ → ¡Pair Float ¡Array`

2. **Linear Types and Type Constructors** Linear data types can contain linear and non-linear types, but non-linear types can only contain non-linear types.

   (a) Explain the statement above. Why would it be a problem if a non-linear data type contains items of linear type, and why is not a problem if a linear data type contains non-linear fields?

   > **Solution:** Just as with function types, this could lead to linear values stored in non-linear types be copied or discarded. However, if a linear type contains non-linear values, then there are restrictions on what you can do with those values, but there won't be any illegal operations.

   (b) How does that affect functions which accept linear values as arguments? For example, what would be the type of a function which accepts to linear lists of identical length and returns a list of the pair-wise sum (i.e., a linear version of `zipWith (+)`)?

   > **Solution:** The second function arrow has to be linear (just as with `writeIORef`. Also, note that if the lists wouldn't have to have the same length, the remaining tail of the longer list cannot just be ignored.