TYPE DIRECTED TRANSLATION

Historically, compiler passes have been what's called *syntax directed*, where the translation is purely a function of the intermediate syntax provided. In *type directed translation*, the translation function is parameterized on both the syntax and its corresponding type.

This can be enormously useful for a number of reasons. Type information can be used for certain optimizations. Much research has been put into using type information as a sort of certificate for certain properties of a program.

Perhaps most importantly, type directed translation is a strong safe guard against compiler bugs. If one believes that strong, static typing improves programmer productivity (as most people reading this text will), one may as well believe it for each of his or her intermediate languages as well.

A note to the reader: nothing in this lecture is particularly formal. Everything is provided by way of example.

For instance, our judgment will now look something of the form

$$\Gamma \vdash e : \tau \rightsquigarrow e$$

where the colored text indicates terms in the target language.

There are two main properties we'd like to have in type directed translation. The first is

$$\begin{array}{l}\text{If } \Gamma \vdash e : \tau \rightsquigarrow e \\ \text{and } \tau \rightsquigarrow \tau \\ \text{and } \Gamma \rightsquigarrow \Gamma \\ \text{then } \Gamma \vdash e : \tau.\end{array}$$

Which is to say, if there is a typing derivation for a term, it then has some target translation. Context translation is simply a map of type translations.

The second property, called *coherence*, says that

$$\begin{array}{l}\text{If } \Gamma \vdash e : \tau \rightsquigarrow e \\ \text{and } \Gamma \vdash e : \tau \rightsquigarrow e', \\ \text{then } \Gamma \vdash e \cong e' : \tau.\end{array}$$

Here, $\cong$ is some equivalence relation on the target language. In some sense, this says: "our translation is independent of our typing derivation."

For sufficiently complex type systems, coherence results are apparently enormously tricky to prove, so will avoid them in this class. Even the literature dances around the topic. Our singleton kind calculus is "sufficiently complex."

For our purposes, translating type constructors is syntax-directed, which makes coherence results easier. We'll have the property that

$$\begin{array}{l}\text{If } \Gamma \vdash c : k \rightsquigarrow c \\ \text{and } \Gamma \vdash c : k \rightsquigarrow c', \\ \text{then } \Gamma \vdash e \equiv e' : k.\end{array}$$

An example was done in class, but the scribe did not find it particularly illuminating.