To: Matthias Felleisen, boss
From: Alex Knauth
Milo Turner
Subject: Implementing Module / Functor Systems with Macros

Our objective is to build an ML-style module and functor system for a simple typed language. We hope to show how Racket's metaprogramming facilities can be used to extend an existing typed language with a module and functor system. To do this, we will build a typed core language, then extend it with a module language without modifying the core.

Core Language

The *core language* will be the second-order lambda calculus, which features lambda functions, application, and polymorphism. The grammar for the core language is described in the grammar specification.

Module Language

Like ML, the *module language* will come with its own semantics and typing rules. "Values" in the module language are called modules, and the "types" of these modules are called signatures. The grammar for modules and signatures are described in the grammar specification.

A *module* can be either a mod or a functor. A *mod* (sometimes referred to as a *structure*) is a collection of type and value definitions. A *functor* is a module that is parameterized over another module, and can be conceptualized as a "function" from modules to modules.

Operations on modules include applying a functor to a module to produce another module, and sealing a module with a signature. The module language will not support features such as nested modules or recursive functors.

There is a corresponding *signature* form for each kind of module. A *sig* describes the bindings of a mod, and a *pi* describes the input and output of a functor.

Milestones

We want to build this prototype as a proof of concept. We plan to move on to adding modules and functors to Hackett if it is successful.

- Develop the grammar capabilities of the module language; anticipate edge cases by building significant examples.
- Build the typed core language.
- Create the first iteration of the module system, featuring only mod and sig forms.
- Implement functors and pis. Ensure that module resolution is consistent. At this point, we should be able to use our examples to set up a demo.

- Explore integration of this system into the Hackett language.