► CHUNG-CHIEH SHAN, *Functional modularity in the lambda calculus*.
Department of Linguistics, Cornell University, 203 Morrill Hall, Ithaca, NY 14853-4701, USA.
E-mail: ccshan@post.harvard.edu.

A module is a part of a *description* of a system, not necessarily a physical part. For example, a program that is stored only in compiled form by the computer running it may nevertheless be better described by source code, so the program may have source modules that are hard to recover at run time. This functional notion of modularity is relevant for organisms, species, and scientists because they all need to adapt to changes in the environment without re-learning, re-evolving, or re-discovering each new system from scratch.

Lambda calculi and type systems offer expressive ways to describe natural language and thus carve out its modules. In particular, lambda calculi can express modules that operate on other modules, and type systems can circumscribe information flow among modules whose operation is tightly intertwined. I will illustrate the use of this expressivity with two examples: First, Abstract Categorial Grammars can carve out PF and LF as a collection of modules mediated by syntax. Second, monad transformers can carve out side effects such as continuations and state as a collection of modules mediated by lexical items.