Programmers, like pen-and-paper mathematicians, often seek to define new syntactic sugar that lowers the syntactic cost of common idioms. Unfortunately, the most direct mechanisms of syntactic control available to programmers do not support modular reasoning about determinism (i.e. separately defined forms can conflict syntactically with one another), and they obscure the type and binding structure of the program. We argue therefore that these mechanisms are unreasonable for programming "in the large". Typed term-rewriting macro systems (e.g. as available in Scala) are more reasonable, but afford programmers limited syntactic control.

This thesis formally describes typed syntax macros (TSMs), which give library providers programmatic control over both the parsing and expansion of expressions and patterns of generalized literal form at a specified type or parameterized family of types. Expansion is strictly hygienic, meaning that 1) expansions can refer to external bindings only via spliced terms or supplied module parameters; and 2) expansions do not reveal internal bindings to spliced terms or to the remainder of the program. We argue that this mechanism occupies a "sweet spot" in the design space, in that it captures many common syntactic idioms while avoiding the problem of syntactic conflicts by construction and supplying clients with clear abstract reasoning principles.