

# MetaL — A Library for Formalised Metatheory in Agda

Robin Adams

March 21, 2017

## 1 Design Criteria

This library was produced with the following design goals.

- The library should be *modular*. There should be a type `Grammar`, and results such as the Substitution Lemma `To do (??)` should be provable ‘once and for all’ for all grammars.<sup>1</sup>
- It should be possible for the user to define their own operations, such as path substitution `To do (??)`
- Operations which are defined by induction on expressions should be definable by induction in Agda. Results which are proved by induction on expressions should be proved by induction in Agda.

## 2 Grammar

For a running example, we will construct the grammar of the simply-typed lambda-calculus, with Church-typing and one constant ground type  $\perp$ . On paper, in BNF-style, we write the grammar as follows:

$$\begin{array}{ll} \text{Type } A & ::= \perp \mid A \rightarrow A \\ \text{Term } M & ::= x \mid MM \mid \lambda x : A.M \end{array}$$

### 2.1 Taxonomy

A *taxonomy* is a set of *expression kinds*, divided into *variable kinds* and *non-variable kinds*.

---

<sup>1</sup>For future versions of the library, we wish to have a type of reduction rules over a grammar, and a type of theories (sets of rules of deduction) over a grammar.