CREATIVE FOUNDATIONS
FOR PROGRAM VERIFICATION

introduction and motivation

philosophical background

a primitive formal system

why creative?
VERIFYING VERIFICATION SYSTEMS

Problems:
  Proving correctness of implementation
  Establishing consistency of foundations

Could we attach any value to a proof within the same system?

NO

Therefore, we must have an ULTIMATE FORMAL FOUNDATION which is not itself formally verified.

How do we maximise confidence in such an "ultimate formal foundation"?

* make it simple
* make it "transparent to intuitions"
* give it prolonged theoretical scrutiny and practical exposure
A FORMALISATION OF
THE CREATIVE THEORY

// abstract syntax
term ::= K | S | term $Ap term

// auxiliary definitions
|| We define the type proforma, and the operation st which
|| substitutes a term into a proforma yielding a term.
proforma ::= M | T term | proforma $Pap proforma
st :: proforma -> term -> term
st M u = u
st (T t) u = t
st (p $Pap q) u = (st p u) $Ap (st q u)

// signature
abstype theorem
with kaxiom :: theorem
    krule :: (theorem, proforma, term, term) -> theorem
    srule :: (theorem, proforma, term, term, term) -> theorem
theorem == term

// axiom
kaxiom = K

// inference rules
krule (th,p,u,v) = st p ((K $Ap u) $Ap v),
\( th = st\ p\ u \)

\[
srule (th,p,u,v,w) = st\ p\ (((S\ $Ap\ u)\ $Ap\ v)\ $Ap\ w),
\]
\[
  th = st\ p\ ((u\ $Ap\ w)\ $Ap\ (v\ $Ap\ w))
\]
PURE COMBINATORY LOGIC

// abstract syntax
term ::= K | S | term $Ap term

// interpreting combinators

Each combinator represents a (partial) function. "$Ap" represents function application.

K and S are primitive functions defined as follows:

\[(\text{K } \text{A}) \text{B} = \text{A}\]

\[(((\text{S } \text{A}) \text{B}) \text{C}) = (\text{A} \text{B} \text{C})\]

// representing computable functions

By choosing suitable terms in the pure combinatory logic we can represent the natural numbers as combinators. Under such an interpretation combinators may be interpreted as partial computable functions over the natural numbers. It transpires that this formalism is computationally universal, i.e. every partial computable function over the natural numbers is represented by some combinator. It follows that, again by the use of suitable encodings, all the (partial) computable functions over combinators are also representable. (note that the identity function is not a suitable encoding)

// representing partial characteristic functions
By choosing suitable representatives for the truth values partial characteristic functions over encoded integers, or over encoded combinators may be represented. The combinator representing a partial characteristic function is one which when applied to the combinator representing some object in the domain will yield "True" if and only if the object is in the set of which the first combinator represents a partial characteristic function.