Reflexive Foundations for Computer Science

Roger Bishop Jones

ICL Defence Systems
REFLEXIVE FOUNDATIONS
for
COMPUTER SCIENCE

FOUNDATION SYSTEMS
The Analytic
Logic & Sets
Paradoxes
Resolution
Well Foundedness

REFLECTION
What is Reflexiveness?
The Lambda Calculus
Denotational Semantics
Abstraction
Metareflection

REFLECTION in CLASSICAL FOUNDATIONS
Functions as Graphs
Self Application
Functions as Rules
Syntactic models in Set Theory

REFLEXIVE FOUNDATIONS
Existing Systems
Recursion Theory
The Creative Theory
The Semantics of Quantification
Proof Theory
THE ANALYTIC

HUME (1739)

(An Enquiry Concerning Human Understanding"
Section IV part I)

All the objects of human reason or enquiry may be divided into two kinds, to wit, "Relations of Ideas" and "Matters of Fact".

Of the first kind are the sciences of Geometry, Algebra and Arithmetic, and in short, every affirmation which is either intuitively or demonstratively certain.

Matters of fact, which are the second objects of human reason, are not ascertained in the same manner, nor is our evidence of their truth, however great, of a like nature with the forgoing.
INFORMAL, SET THEORETIC, FOUNDATIONS

Set theory provides a means of deriving a very large body of mathematics from two principles:

EXTENSIONALITY

Two sets are equal if and only if they have exactly the same members:

\[ \forall s, t. \ s = t \iff (\forall x. \ x \in s \iff x \in t) \]

ABSTRACTION

For every propositional function P, there is a set containing just those elements for which P is true:

\[ \exists x. \ \forall y. \ y \in x \iff P(y) \]

This set is denoted by the expression \{y \mid P(y)\}. 
RUSSELL’s PARADOX

Consider the set $R = \{ x \mid x \notin x \}$.

We have:

$R \in R \Rightarrow R \notin R$

and:

$R \notin R \Rightarrow \neg R \notin R \Rightarrow R \in R$

revealing a contradiction in these very simple ideas.
THE CUMULATIVE HIERARCHY
of
SETS

Sets are considered to be formed from the empty set in some particular order using a small repertoire of set forming operations, viz:

- Pair \{a,b\}
- Power set \(\mathcal{P}a\)
- Separation \(\{x:A \mid P\}\)
- Union \(\bigcup a\)
- Replacement

A set may only contain sets previously formed.

Hence, the axiom of WELL FOUNDATION.

TYPE THEORY

A Type Universe is formed from the primitive types \textit{ind} and \textit{bool}, by repeated application of the function space type constructor \(\to\).

type ::= ind \mid bool \mid \text{type} \to \text{type}

All terms are typed, and variables range over the values in just one type.
FOUNDATION SYSTEMS

Formal systems sufficient for the development of mathematics without the introduction of any further axioms or rules. (definitions suffice)

They offer:

SAFETY

and

SUFFICIENCY
WHAT IS REFLEXIVENESS?

SELF APPLICATION of FUNCTIONS

A UNIVERSE $U$ such that $U \in U$

TYPE of TYPES, $type$ such that $type: type$

REASONING, WITHIN a FORMAL SYSTEM, about ITSELF
FUNCTIONS AS GRAPHS

In set theory a function $f$ defined

$$f \ x = \exp$$

is normally represented by the set:

$$\{(x,y) \mid y = \exp\}$$

where $(x,y) = \{\{x\}, \{x,y\}\}$

Consequently:

$$\text{dom } f \subseteq \bigcup \bigcup f$$

Well Foundedness

$$+$$

Functions as Graphs

$$\Rightarrow$$

No Self-Application
FUNCTIONS as RULES

We chose some language and assign a meaning to the terms of the language as operations over the same language. The most obvious example is the partial recursive functions over natural numbers, which can be encoded as natural numbers.

\[
\{ \} : \mathbb{N} \rightarrow (\mathbb{N} \times \mathbb{N})
\]

Alternatively, a partition of the terms of some language may be defined as the coarsest partition consistent with certain reduction rules. This is the technique used in the lambda-calculus and in combinatory logic.

\[
\text{conv} : \text{term} \rightarrow \text{term} \rightarrow \text{bool}
\]
RECURSION THEORY and FORMAL SYSTEMS

A class of problems is **decidable** (recursive) if there exists an algorithm for computing the answer to the problems which always terminates.

A class of problems is **semi-decidable** (r.e.) if there exists an algorithm for computing the answer to the problems which terminates if the answer to the problem is positive.

For finitary formal systems:

- The set of theorems is r.e.
- Every *formally representable* set is r.e.

For first order languages:

- every consistent theory has a **finite** or **countable** model

For foundation systems:

- All consistent formalisations of arithmetic are incomplete.
- Only a countable number of sets can be described in first order set theory.
THE LAMBDA CALCULUS
and
COMBINATORY LOGIC

K u v = u

S u v w = (u w) (v w)

K = \lambda u v.u

S = \lambda u v w.(u w)(v w)

term ::= variable | term term | \lambda variable . term

comb ::= K | S | comb comb

\lambda x.x = I = S K K

S K K x = (K x)(K x) = x
DOMAIN EQUATIONS

e.g.

\[ V = N + V \rightarrow V \]

This has NO SET THEORETIC SOLUTION because:

- requires self-application
- cardinality problems

SOLUTION

Abandon "functions as graphs"

settle for 'isomorphism'

\[ V \cong N + V \rightarrow V \]

and chose a smaller function space
METAREFLECTION

ML and PROOF DEVELOPMENT SYSTEMS

LCF, LCF-LSM, HOL, NUPRL are all proof development systems written in the functional language ML. Primitive axioms and rules of inference are embodied in an abstract data type defining the theorems of the logic. Derived inference rules may then be programmed in ML and their soundness is guaranteed by the type checker.

Using a METALOGIC

If the metalanguage were a logic with a usable executable sublanguage, then the logic could be used to prove the soundness of derived rules enabling primitives steps to be omitted from proofs.

Suppose OBJECT-language = META-language?
HOW TO DEVELOP

a

REFLEXIVE FOUNDATION SYSTEM

1 Choose a programming language
   (preferably a neat one)

2 Define the semantics of the language
   (maybe operationally)

3 Add restricted quantification

4 Worry about consistency

5 Pick up the proof rules from a classical foundation system
'SIMPLE' MODELS

Pure Combinatory Logic