A Combinatory Theory of Partial Functions

Roger Bishop Jones

ICL Defence Systems

1. INTRODUCTION
This document specifies semi-formally and formally a version of illiative combinatorial logic.
The system is at bottom a theory of "restricted generality". In this formalisation we attempt to
explore the idea that the terms of the language represent computable functions over the terms of
the language, (or perhaps over a quotient of a primitive sublanguage) and that a true proposition is
an expression which evaluates to the boolean value "true".
In this spirit the combinator Ξ is interpreted as yielding a computable function from the terms of
the language which only sometimes yields a boolean, i.e. a partial function into {true,false}. The
rules for substitution into expressions involving Ξ therefore depend upon an understanding of
what substitutions are reasonable in the argument to a partial function.
For this reason I develop the definition of a partial function in the combinatory logic prior to sup-
plying the rules for substitution.
I have also taken the view, contrary to my previous work, that reduction of the number of primi-
tives is not desirable, since it results in a less abstract system and may increase the risk of incon-
sistency. Reductions, where they are possible, will appear in the semantics.
The resulting theory has in some respects a flavour of constructive type theories. This is entirely
superficial and largely accidental.
The semi-formal notation is marked by a vertical bar in the left margin. The formal notation is in
the language Miranda and occurs between marks ||< and ||> in the text. To make the Miranda
more readable special symbols have been used where less readable alphanumeric constructors are
necessary. Special translation facilities are used to enable the text to be processed by the Miranda
compiler.

2. SYNTAX
The abstract syntax of terms is that of the lambda calculus. A concrete syntax provides more read-
able forms for important constructs. The formal definition is provided in terms of the abstract
syntax, the informal in terms of the concrete syntax. There is therefore not a very close match
between the formal and informal accounts.
2.1. Formal Abstract Syntax

A term is a constant, a variable, an application or an abstraction.

\[
\begin{aligned}
\text{term} &::= \text{V [char]} \mid \text{C [char]} \mid \text{term } \text{term} \mid \lambda \text{[char]} \text{term} \\
\end{aligned}
\]

The formal system describes how to prove "sequents".

A sequent is a list of terms followed by "" followed by a list of terms and should be read "if each term on the left is true then so is each term on the right".

2.2. Concrete Syntax

The concrete syntax for the core language is described here. Extensions will be described informally as new constants are introduced.

\[
\begin{aligned}
\text{term} &::= \text{constant} \mid \text{variable} \mid \text{term } \text{term} \mid \lambda \text{variable} \text{term} \\
\text{sequent} &::= [\text{term}] [\text{term}] \\
\end{aligned}
\]

2.3. Proofs

The syntax of proofs is also given in an abstract and a concrete form.

The abstract form consists of a Miranda abstract data type which enables the computation of a subset of the sequents known as theorems.

The concrete form consists of axiom schemata, showing a set of sequents which may be accepted as theorems without proof, and a collection of rules indicating how a sequent may be proven to be a theorem given suitable premises which are themselves (shown to be) theorems.

The axiom and rule schemata employ free syntactic variables to make clear which sequents are proper instances of the schemas. These variables each range over specific syntactic categories as follows:

x, x2, y, z range over variables.

u, u', v, v', w are metavariables ranging over terms.

Φ, Γ, Θ, Δ are metavariables ranging over lists of terms.

Φ ⊆ Γ should be read "every term in Φ is also in Γ".

Informal further restrictions on the permitted instantiation of these metavariables may be specified against the schema. Ambiguities should be resolved by reference to the formal abstract specification.
The primitive constants are:

\[=\] equality (infix)

\[\lor\] logical OR (infix)

\[\forall\] restricted quantification

\[\downarrow\] choice function

3. THE BASIC SYSTEM

4. DOMAINS

5. TYPES