Bishop-style constructive mathematics in type theory — a tutorial

Erik Palmgren
Stockholm University
www.math.su.se

Constructive Mathematics: Foundations and Practice
Niš, June 24-28, 2013

Introduction

"Bishop-style constructive mathematics is mathematics based on intuitionistic logic." (D.S. Bridges, F. Richman and others)

"The difference, then, between constructive mathematics and programming does not concern the primitive notions of the one or the other, because they are essentially the same, but lies in the programmer's insistence that his programs be written in a formal notation so that they can be read and executed by a machine, whereas, in constructive mathematics as practised by Bishop (1967), for example, the computational procedures (programs) are normally left implicit in the proofs, so that considerable further work is needed to bring them into a form which makes them fit for mechanical execution." (P. Martin-Löf, *Constructive Mathematics and Computer Programming*[1], 1979, pp. 156.)

---

[1] http://www.cs.tufts.edu/~nr/cs257/archive/per-martin-lof/constructive-math.pdf

Introduction - What is a set?

A naive notion of set (cf. Frege) is obtained by collecting all objects or other sets, according to some selection criterion $Q(x)$

$$\{x \mid Q(x)\}$$

Frege's "naive" set theory is inconsistent (Russell's paradox).

The iterative notion of set (G. Cantor 1890, E. Zermelo 1930) is to build up sets by stages in a cumulative hierarchy using transfinite iterations of power sets. The power set is not well understood from a constructive point of view.

Fortunately, it is possible to refine the iterative notion constructively by using well-founded trees (Aczel 1978) and to avoid power sets and ordinals.

Encoding of mathematical objects as iterative sets

All mathematical objects are built from the empty set (E. Zermelo 1930)

Natural numbers are for example usually encoded as

$$0 = \emptyset \quad 1 = 0 \cup \{0\} = \{\emptyset\} \quad 2 = 1 \cup \{1\} = \{\emptyset, \{\emptyset\}\} \quad \cdots \quad .$$

Pairs of elements can be encoded as $\langle a, b \rangle = \{\{a\}, \{a, b\}\}$.
Functions are certain sets of pairs objects ... etc.

Quotient structures are constructed by the method of equivalence classes — only one notion of equality is necessary.

(J.Myhill and P.Aczel (1970s): constructive versions of ZF set theory.)

What is a set? A more basic view

"A set is not an entity which has an ideal existence: a set exists only when it has been defined. To define a set we prescribe, at least implicitly, what we (the constructing intelligence) must do in order to construct an element of the set, and what we must do to show that two elements are equal" (Errett Bishop, Foundations of Constructive Analysis, 1967.)

Martin-Löf's constructive type theory conforms to this principle of defining sets.

Abstraction levels

One may disregard the particular representations of set-theoretic constructions, and describe their properties abstractly (in the spirit of Bourbaki, or even category theory).

For instance, the cartesian product of two sets A and B may be described as a set $A \times B$ together with two *projection* functions

$$\pi_1 : A \times B \longrightarrow A \qquad \pi_2 : A \times B \longrightarrow B,$$

such that for each $a \in A$ and each $b \in B$ there exists a unique element $c \in A \times B$ with $\pi_1(c) = a$ and $\pi_2(c) = b$. Thus π_k picks out the kth component of the abstract pair.

Reference to the particular encoding of pairs is avoided. This is a good principle in mathematics as well as in program construction.

Bishop's set theory

Errett Bishop introduced in his book *Foundations of Constructive Analysis* from 1967 a set theory which is of a more type-theoretic character as we shall see. *Bishop-style constructive mathematics* is mathematics done in the way of this book. Some prominent works:

- ▶ D.S. Bridges (1979). *Constructive Functional Analysis*. Pitman.
- ▶ E. Bishop and D.S. Bridges (1985). *Constructive Analysis.* Springer-Verlag.
- ▶ D.S. Bridges and F. Richman (1987). *Varieties of Constructive Mathematics.* London Mathematical Society Lecture Notes, Vol. 97. Cambridge University Press.
- ▶ R. Mines, F. Richman and W. Ruitenburg (1988). *A Course in Constructive Algebra.* Springer.

Plan of lectures

1. Introduction
2. Constructive interpretation of logic
3. Constructive type theory
4. Sets and equivalence relations
5. Choice sets and axiom of choice
6. Relations and subsets
7. Finite sets and relatives
8. Quotients
9. Universes and restricted power sets
10. Categories
11. Constructive set theory CZF as part of Bishop's set theory
12. Relation to categorical logic

2. Constructive interpretation of logic

The logical rules that can be accompanied by mental constructions constitute the so-called *intuitionistic logic.*

Here Heyting and Kolmogorov made crucial contributions towards its formalisation. The *Brouwer-Heyting-Kolmogorov* (BHK) interpretation of logic consists telling what constructions p *testifies*, *verifies* or *proves* a certain proposition A.

$$p : A$$

It is not precisely said what these constructions are, but they are general understood as being computable.

Kolmogorov calls A a *problem* and p a *solution*.

2.1 BHK-interpretation

We explain what it means that a construction p is a *witness* to the truth of the proposition A by induction on the form of A. This will be expressed more briefly as p *is a witness to* A, or that p *testifies* A.

- $\perp$ has no witnesses.
- p testifies $s = t$ iff $p = 0$ and s and t are computed to the same thing. (Here s and t are supposed to be natural numbers, or some similar finitely given mathematical objects.)
- p testifies $A \wedge B$ iff p is a pair $\langle a, b \rangle$ where a testifies A and b testifies B.
- p testifies $A \longrightarrow B$ iff p is a function which to each witness a to A gives a witness $p(a)$ to B.

- p testifies $A \vee B$ iff p has the form inl(a), in which case a testifies A, or p has the form inr(b), in which case b testifies B
- p testifies $(\forall x \in S)A(x)$ iff p is a function which to each element $d \in S$, provides a witness $p(d)$ to $A(d)$.
- p testifies $(\exists x \in S)A(x)$ iff p is a pair $\langle d, q \rangle$ consisting of $d \in S$ and a witness q to $A(d)$.

A proposition A is *valid under the BHK-interpretation,* or is *constructively true,* if there is a construction p such that p testifies A.

Note that the use of the word "witness" extends its usage in classical logic about existence statements. One may say that 2 is a witness to $(\exists x)\, x^2 = 4$ being true.

NB The established standard terminology is rather to say that p is a *proof* of A, and the construction p is called *proof-object.* (However we want to avoid possible confusion with formal derivations.)

Examples of BHK-interpretations

The lambda notation $\lambda x.a(x)$ is alternative notation for the function $x \mapsto a(x)$.

Examples

1. $p = \lambda x.x$ is a witness to the truth of $A \longrightarrow A$. This is clear, since $p(a) = (\lambda x.x)(a) = a$ and if a testifies A, then so does $p(a)$.

2. A witness to $A \wedge B \longrightarrow B \wedge A$ is given by the construction $f = \lambda x.\langle \pi_2 x, \pi_1 x \rangle$. π_k is the k-th projection.

3. Consider the proposition $\bot \longrightarrow A$. A witness to this is an arbitrary function f such as $f(x) = 42$: Suppose that a is a witness to $\bot$. But according to the BHK-interpretation $\bot$ has no witness, so we have a contradiction. By the absurdity law, anything follows, in particular that 42 is a witness to A.

Negation is *defined* as $\neg A =_{\mathrm{def}} (A \longrightarrow \bot)$. To prove $\neg A$ amounts to proving that A leads to a contradiction.

Example

The contraposition law $(A \longrightarrow B) \longrightarrow (\neg B \longrightarrow \neg A)$ is valid in under the BHK-interpretation. Suppose that f testifies $A \longrightarrow B$. We wish to find a witness to $(\neg B \longrightarrow \neg A)$, i.e. $(B \longrightarrow \bot) \longrightarrow (A \longrightarrow \bot)$. Suppose therefore that g testifies $\neg B$ and a testifies A. Thereby $f(a)$ is a witness to B, and hence $g(f(a))$ is a witness to $\bot$. The construction $\lambda a.g(f(a))$ thus testifies $\neg A$. Abstracting on g it is clear that $\lambda g.\lambda a.g(f(a))$ testifies $\neg B \longrightarrow \neg A$. The construction

$$\lambda f.\lambda g.\lambda a.g(f(a))$$

is finally the witness to the law of contraposition.

The Principle of Excluded Middle

The Principle of Excluded Middle (PEM)

$$A \vee \neg A$$

is not obviously valid under the BHK-interpretation, since we would need to find a method, which given the parameters in A, decides whether A is valid or not. If we restrict the possible constructions to computable functions, we may actually show that PEM is not constructively true. It is known that there is a primitive recursive function T such that $T(e, x, t) = 1$ in case t describes a terminating computation (t is, so to say, the complete "trace" of the computation) for the Turing machine e with input x, and having the value $T(e, x, t) = 0$ otherwise. By a suitable coding, the arguments to T may be regarded as natural numbers.

The halting problem for e and x may now be expressed by the formula

$$H(e, x) =_{\text{def}} (\exists t \in \mathbb{N})\ T(e, x, t) = 1.$$

According to PEM

$$(\forall e \in \mathbb{N})(\forall x \in \mathbb{N})\ H(e, x) \vee \neg H(e, x).$$

If this proposition were to have a computable witness, then we could decide the halting problem, contrary to Turing's well-known result that this is algorithmically undecidable.
The principle of indirect proof, *reductio ad absurdum* (RAA)

$$\neg\neg A \longrightarrow A$$

can be shown to be equivalent to PEM within intuitionistic logic, so it is not valid under the BHK-interpretation either.

2.2 Intuitionistic logic

Intuitionistic logic is best described by considering the derivation rules for natural deduction and then remove the RAA rule (principle of indirect proof):

Derivation rules:

$$\frac{A \quad B}{A \wedge B} \ (\wedge I) \qquad\qquad \frac{A \wedge B}{A} \ (\wedge E1) \qquad \frac{A \wedge B}{B} \ (\wedge E2)$$

$$\begin{array}{c} \overline{A}^{\,h} \\ \vdots \\ \dfrac{B}{A \longrightarrow B} \end{array} (\longrightarrow I, h) \qquad\qquad \frac{A \longrightarrow B \quad A}{B} \ (\longrightarrow E)$$

$$\frac{A}{A \vee B} \ (\vee I1) \qquad \frac{B}{A \vee B} \ (\vee I2) \qquad \frac{A \vee B \quad \overset{\overline{A}^{\,h_1}}{\underset{C}{\vdots}} \quad \overset{\overline{B}^{\,h_2}}{\underset{C}{\vdots}}}{C} \ (\vee E, h_1, h_2)$$

$$\frac{\bot}{A} \ (\bot E) \qquad \frac{\overset{\overline{\neg A}^{\,h}}{\underset{\bot}{\vdots}}}{A} \ (RAA, h)$$

$$\frac{A}{(\forall x)A} \ (\forall I) \qquad\qquad \frac{(\forall x)A}{A[t/x]} \ (\forall E)$$

$$\overline{A}^{\,h}$$
$$\vdots$$

$$\frac{A[t/x]}{(\exists x)A} \ (\exists I) \qquad \frac{(\exists x)A \quad C}{C} \ (\exists E, h)$$

The rules for the quantifiers have familiar restrictions.

We can verify the validity of rules under BHK:

$$\frac{a : A \qquad b : B}{\langle a, b \rangle : A \wedge B} \ (\wedge I) \qquad\qquad \frac{c : A \wedge B}{\pi_1 c : A} \ (\wedge E1) \qquad \frac{c : A \wedge B}{\pi_2 c : B} \ (\wedge E2)$$

$$\cfrac{\overline{x : A}^{\ h}}{\begin{array}{c} \vdots \\ b : B \end{array}}{\lambda x.b : A \longrightarrow B} (\longrightarrow I, h) \qquad\qquad \frac{c : A \longrightarrow B \qquad a : A}{c(a) : B} \ (\longrightarrow E)$$

$$\frac{a : A[t/x]}{\langle t, a \rangle : (\exists x)\, A}\ (\exists I)$$

$$\overline{y : A}^{\,h}$$
$$\vdots$$
$$\frac{c : (\exists x)A \quad d : C}{d[\pi_1 c, \pi_2 c/x, y] : C}\ (\exists E, h)$$

$$\frac{a : A}{\lambda x.a : (\forall x)\, A}\ (\forall I)$$

$$\frac{c : (\forall x)\, A}{c(t) : A[t/x]}\ (\forall E)$$

We have

Theorem The rules for intuitionistic logic are valid under the BHK-interpretation.

3. Constructive type theory

3.1 Curry-Howard correspondence

It was realized by H.B. Curry that there is a close correspondence between

(1) The implicational fragment of intuitionistic propositional logic.

(2) Simply typed combinatory logic.

(2') Simply typed λ-calculus.

The implicational fragment (1) is given by modus ponens rule and the axiom schemes

$A \Rightarrow B \Rightarrow A$,
$(A \Rightarrow B \Rightarrow C) \Rightarrow (A \Rightarrow B) \Rightarrow A \Rightarrow C$.

(2) The types of the combinators **K** and **S** are

K : $A \to B \to A$,
S : $(A \to B \to C) \to (A \to B) \to A \to C$.

The modus ponens rule

$$\frac{A \Rightarrow B \qquad A}{B}$$

coresponds to the type rule of application in combinatory logic: if
$M : A \longrightarrow B$ and $N : A$ then $MN : B$.

Indeed λ-expressions for the combinators

$\mathbf{K} = \lambda x.\lambda y.x$
$\mathbf{S} = \lambda x.\lambda y.\lambda z.x(z)(y(z))$
are literally witnesses for the BHK-truth of, respectively,
$A \Rightarrow B \Rightarrow A$ and
$(A \Rightarrow B \Rightarrow C) \Rightarrow (A \Rightarrow B) \Rightarrow A \Rightarrow C$.
[Check!]

There is a deeper connection:
Gentzen-Prawitz simplification rules in Natural Deduction

$$
\begin{array}{cc}
[A] \\
\vdots \\
\dfrac{B}{\dfrac{A \Rightarrow B \quad A}{B}} \quad \vdots \\
\end{array}
\qquad \text{simplifies to} \qquad
\begin{array}{c}
\vdots \\
A \\
\vdots \\
B
\end{array}
$$

β-rule of λ-calculus: $(\lambda x.b)(a) = b[a/x]$

$$
\begin{array}{cc}
[x : A] \\
\vdots \\
\dfrac{b : B}{\dfrac{\lambda x.b : A \Rightarrow B \quad a : A}{(\lambda x.b)(a) : B}} \quad \vdots \\
\end{array}
\qquad \text{simplifies to} \qquad
\begin{array}{c}
\vdots \\
a : A \\
\vdots \\
b[a/x] : B
\end{array}
$$

3.2 Propositions as Types

W. Howard (1969) extended the correspondence to full first order intuitionistic logic.
Thus the *Curry-Howard correspondence* or *Curry-Howard isomorphism.*

It is based on the very general

Propositions-as-types principle:

A proposition is the type of its proofs (witnesses)

To achieve propositions-as-types we must introduce types corresponding to propositions $A(x)$ depending on a parameter x which may range over some set (or type) S.

Thus we need *dependent types*.

$$T_x \qquad (x : S)$$

which we provisionally think of as families of sets.

(A fully formal treatment of dependent types is fairly complicated.)

The BHK-witnesses to $(\forall x : S)A(x)$ are constructions p so that $p(s)$ is a witness to $A(s)$ for all $s : S$.

The corresponding type is the *dependent product* or Π-type

$$\Pi_{x \in S} T_x = \{p : S \longrightarrow \cup_{x \in S} T_x \mid \text{for all } x : p(x) \in T_x\}.$$

The BHK-witnesses to $(\exists x : S)A(x)$ are constructions $\langle d, q \rangle$ so that q is a witness to $A(d)$.

The corresponding type is the *disjoint union* or Σ-type

$$\Sigma_{x \in S} T_x = \{(x, q) \mid x \in S, q \in T_x\}$$

(Common notation: $(\Pi x : S) T_x$ respectively $(\Sigma x : S) T_x$.)

3.3. Martin-Löf Type Theory

Constructive Type Theory or Martin-Löf Type Theory (MLTT) is a formal system intended for the foundations of constructive mathematics.

It uses systematically the Propositions-as-types principle: there is *no difference between propositions and types.*

It exists in many variants, some of them have successfully been implemented on computers (c.f. Alf, Agda, Coq, Epigram) to exploit the possibility to execute *proofs as programs.*

There is a well-developed meaning theory for MLTT (see Martin-Löf 1984, 1985, 1996) based on proof-theoretical semantics, which makes it possible to see the correctness and computational content in the rules and to further extend the theory.

A general principle (Martin-Löf 1984) is that to define a type A, one must say

- what it means to be a canonical element of the type
- when two canonical elements are equal.

Example

Type of natural numbers: canonical elements

$$0 : \mathbb{N} \qquad \frac{b : \mathbb{N}}{s(b) : \mathbb{N}} \qquad 0 = 0 : \mathbb{N} \qquad \frac{b = c : \mathbb{N}}{s(b) = s(c) : \mathbb{N}}$$

• Two types A and B are equal ($A = B$) iff they have the same canonical elements and the same equalities holds between canonical elements.

• Meaning of $b : A$ (b is an element of A):

An b element of type A is a method (or program) which when executed gives a canonical element of type A.

Example
$s(0) + s(0)$ is a non-canonical element of $\mathbb{N}$ which executes to the canonical $s(s(0))$. Thus $s(0) + s(0) : \mathbb{N}$.

• Meaning of $b = c : A$ (b and c are equal elements of A):

when both executed, b and c give equal canonical elements.

This leads to the four basic *judgement forms* of MLTT

$$A \text{ type} \qquad b : A \qquad b = c : A \qquad A = B$$

Each of judgement forms $\mathcal{J}$ comes in a hypothetical form

$$\mathcal{J} \qquad (x_1 : A_1, x_2 : A_2(x_1), \ldots, x_n : A_n(x_1, \ldots, x_{n-1}))$$

• For $n = 1$ it means that

(1) $\mathcal{J}[d/x_1]$ whenever $d : A_1$,

(2) $A[d/x_1] = A[e/x_1]$ whenever $d = e : A_1$, in case $\mathcal{J}$ is "A type",

and $b[d/x_1] = b[e/x_1] : A$ whenever $d = e : A_1$, in case $\mathcal{J}$ is "$b : A$".

Π-types. Let B be a type depending on $x : A$. The introduction rule for Π is this:

$$\frac{\begin{array}{c} (x : A) \\ \vdots \\ b : B \end{array}}{\lambda x.b : (\Pi x : A)B}(\Pi I)$$

The elimination rule is

$$\frac{f : (\Pi x : A)B \qquad a : A}{f(a) : B[a/x]} \; (\Pi E)$$

The associated computation rule is $(\lambda x.b)(a) = b[a/x] : B[a/x]$, often called the β-rule.

The Π-types covers both universal quantifiers and implication:

- $(\forall x : S)B(x) = (\Pi x : S)B(x)$
- $B \to C = (\Pi p : B)C$

Σ-types. Let B be a type depending on $x : A$. The introduction rule for Σ is

$$\frac{a : A \qquad b : B[a/x]}{\langle a, b \rangle : (\Sigma x : A)B} \ (\Sigma I)$$

If A is regarded as the quantification domain and B is regarded as a proposition, we see that (ΣI) may be read as the rule $(\exists I)$.

In case B is independent of x, so that $B[a/x] = B$, we see that the rule (ΣI) has the same shape as $(\wedge I)$. (Set-theoretically it holds in this case that: $(\Sigma x \in A)B = A \times B$.)

Σ-types

The elimination rule for Σ is the following, supposing that C type $(z : (\Sigma x : A)B)$

$$\frac{\begin{array}{c} (x : A, y : B) \\ \vdots \quad \vdots \\ c : (\Sigma x : A)B \quad d : C[\langle x, y \rangle / z] \end{array}}{\mathsf{split}(c, \lambda x.\lambda y.d) : C[c/z]} \quad (\Sigma E)$$

The associated computation rule is

$\mathsf{split}(\langle a, b \rangle, g) = g(a)(b) : C[\langle a, b \rangle / z].$

Binary sums

Introduction rule:

$$\frac{a : A}{\text{inl}(a) : A + B} \qquad \frac{b : B}{\text{inr}(b) : A + B}$$

For C type $(z : A + B)$ we have the elimination rule

$$\frac{c : A + B \quad d : C[\text{inl}(x)] \quad e : C[\text{inr}(y)]}{D(\lambda x.d, \lambda y.e, c) : C[c/z]}$$

Computation rules:

$$D(\lambda x.d, \lambda y.e, \text{inl}(a)) = d[a/x] \qquad D(\lambda x.d, \lambda y.e, \text{inr}(b)) = e[b/y]$$

Empty type and unit type

The empty type N_0 has no introduction rules.
For C type $(z : N_0)$ there is the elimination rule

$$\frac{c : N_0}{R_0(c) : C[c/z]}$$

The unit type N_1 has one introduction rule: $\star : N_1$. For C type
$(z : N_1)$ there is the elimination rule

$$\frac{c : N_1 \quad d : C[\star/z]}{R_1(c, d) : C[c/z]}$$

Inductive types: natural numbers

$$0 : \mathbb{N} \qquad \frac{a : \mathbb{N}}{s(a) : \mathbb{N}}$$

$$\frac{c : \mathbb{N} \quad d : C[0/z] \quad \overset{(x : \mathbb{N}, y : C[x/z])}{\underset{\vdots}{e : C[s(x)/z]}}}{R(c, d, \lambda x.\lambda y.e) : C[c/z]}$$

$$R(0, d, \lambda x.\lambda y.e) = d$$
$$R(s(a), d, \lambda x.\lambda y.e) = e[a, R(a, d, \lambda x.\lambda y.e)/x, y].$$

NB: Induction scheme is the type of the recursion operator

Identity types

The equality judgement

$$a = b : A$$

is not a type (i.e. proposition) it self and can not occur in compound expression. Type theory therefore has a special type, the **identity type**:

$$I(A, a, b)$$

to make equality a proposition. Two different axiomatisations:
Extensional identity types:

$$\frac{a = b : A}{r : I(A, a, b)} \qquad \frac{r : I(A, a, b)}{a = b : A} \qquad \frac{c : I(A, a, b)}{c = r : I(A, a, b).}$$

System not strongly normalising. Type checking undecidable.
Intensional identity types: type checking properties decidable.
$I(A \to B, f, g)$ is not extensional equality.

Translation Table

A proposition may be regarded as a type according to the following translation scheme

$$
\begin{array}{ll}
(\forall x : A)P\, x & (\Pi x : A)P\, x \\
(\exists x : A)P\, x & (\Sigma x : A)P\, x \\
P \wedge Q & P \times Q \\
P \vee Q & P + Q \\
P \Rightarrow Q & P \longrightarrow Q \\
\top & \mathrm{N}_1 \\
\bot & \mathrm{N}_0 \\
\neg P \quad (= P \Rightarrow \bot) & P \longrightarrow \mathrm{N}_0
\end{array}
$$

The judgement

A is true

means that there is some p so that $p : A$.

3.4 Coq - an implementation of type theory

Coq[2] is a proof assistant intended for building formal mathematical proof and enabling the extraction of algorithms from constructive proofs. It is based on the *Calculus of Inductive Construction* which in turn uses the principles for Martin-Löf type theory:

- ▶ Types are either built as *generalized function types* or as *inductive types.*
- ▶ The elimination rules are generated by the introduction rules.

However in Coq equality judgments are implicit when working with the system and are embedded in the computation rules.

---

[2]Available from: coq.inria.fr

Basic Martin-Löf type theory can be defined very succinctly in Coq:

```
Definition Pi (A: Type)(B: A -> Type) :=
    (forall x: A, B x).

Inductive Sigma (A: Type) (B: A -> Type) :=
    pair { prj1: A; prj2: B prj1 }.

Inductive BinSum (A: Type)(B: Type) : Type :=
    inl (a: A) | inr (b: B).

Inductive Id (A: Type)(x: A) : A -> Type :=
    refl : Id A x x.

Inductive Nat: Type := zero | S (n:Nat).

Inductive Empty: Type :=.
```

Example Proof script for decidability of equality on Nat

```
Definition P (x:Nat) :=
∀ y , (Id Nat x y) ∨ ¬ (Id Nat x y).

Theorem DecideEquality: ∀ x , P x.
Proof.
induction x. unfold P. induction y. apply inl. apply refl.
apply inr. apply Peano4a.
unfold P. induction y. apply inr. apply Peano4b.
assert (Id Nat x y ∨ ¬ Id Nat x y).
apply IHx. elim H. intro K. induction K. apply inl. apply
refl.
intro K. apply inr. intro L. assert (Id Nat x y) as M.
apply Peano3. apply L. apply K. apply M.
Defined.
```

Decide whether 1 is equal to 2 using the proof.

```
Eval compute in (DecideEquality (S zero) (S (S zero))).

  = inr (Id Nat (S zero) (S (S zero))) (¬ Id Nat (S zero)
(S (S zero)))
    (fun L : Id Nat (S zero) (S (S zero)) =>
    Peano4a zero (Peano3 zero (S zero) L))

  : Id Nat (S zero) (S (S zero)) ∨ ¬ Id Nat (S zero) (S
(S zero))
```

4. Sets and equivalence relations

Relations and predicates on types:

A *predicate P on a type X* is a family of propositions $P\,x$ $(x : X)$.

A *relation R between types X and Y* is a family of propositions $R\,x\,y$ $(x : X, y : Y)$. If $X = Y$, we say that R is a *binary relation* on X.

A binary relation R on X is an *equivalence relation* if there are functions *ref*, *sym* and *tra* with

$ref\,a : R\,a\,a$ $(a : X)$,

$sym\,a\,b\,p : R\,b\,a$ $(a : X, b : X, p : R\,a\,b)$,

$tra\,a\,b\,c\,p\,q : R\,a\,c$ $(a, b, c : X, p : R\,a\,b, q : R\,b\,c)$.

We may suppress the proof objects and simply write, for instance in the last line

$$R \; a \; c \; \text{true} \qquad (a, b, c : X, R \; a \; b \; \text{true}, R \; b \; c \; \text{true}),$$

which is equivalent to

$$(\forall a : X)(\forall b : X)(\forall c : X)(R \; a \; b \wedge R \; b \; c \Rightarrow R \; a \; c) \; \text{true}.$$

Sets

Definition A *set X* is a type X together with an equivalence relation $=_X$ on X. Write this as

$$X = (X, =_X).$$

We shall also write $x \in X$ for $x : X$.

Remark
In Greeneleaf (1981) X is called a *preset,* rather than a type. A *completely presented set* seems to be the terminology of Bishop.

In the type theory community $X = (X, =_X)$ is often known as a *setoid*.

Examples Let N be the type of natural numbers. Define equivalence relations

$x =_{\mathbb{N}} y$ iff $\mathrm{Tr}\,(\mathrm{eq}_{\mathrm{N}}\, x\, y)$

(Here $\mathrm{eq}_N : N \longrightarrow N \longrightarrow \mathrm{Bool}$ is the equality tester for N and $\mathrm{Tr}\,\mathrm{tt} = \top$ and $\mathrm{Tr}\,\mathrm{ff} = \bot$)

$x =_n y$ iff $x - y$ is divisble by n

Then

- $\mathbb{N} = (\mathrm{N}, =_{\mathbb{N}})$ is the *set of natural numbers*
- $\mathbb{Z}_n = (\mathrm{N}, =_n)$ is the *set of integers modulo n.*

Examples (cont.) $N \times N$ denotes the type of pairs of natural numbers. Define an equivalence relation

$(x, y) =_{\mathbb{Z}} (x', y')$ iff $x + y' =_{\mathbb{N}} x' + y$

Then $\mathbb{N} = (N \times N, =_{\mathbb{Z}})$ is the *set of integers.*

$N \times N \times N$ denotes the type of triples of natural numbers. Define an equivalence relation

$(x, y, z) =_{\mathbb{Q}} (x', y', z')$ iff $(z' + 1)(x + y') =_{\mathbb{N}} (z + 1)(x' + y)$

Then $\mathbb{Q} = (N \times N \times N, =_{\mathbb{Q}})$ is the *set of rational numbers.*

These are direct constructions. It is possible to construct them more algebraically using abstract constructions.

Functions vs operations

What is usually called functions in type theory, we call here *operations*.

Definition. A *function* f from the set X to the set Y is a pair (f, ext_f) where $f : X \longrightarrow Y$ is an operation so that

$$(\mathrm{ext}_f \ a \ b \ p) : f \ a =_Y f \ b \quad (a, b : X, p : a =_X b).$$

To conform with usual mathematical notation, function application will be written

$$f(a) =_{\mathrm{def}} f \ a$$

Two functions $f, g : X \longrightarrow Y$ are *extensionally equal*, $f =_{[X \longrightarrow Y]} g$, if there is e with

$$e \ a : f(a) =_Y g(a) \quad (a \in X).$$

Set constructions

The product of sets A and B is a set $P = (P, =_P)$ where $P = A \times B$ (cartesian product as types) and the equality is defined by

$(x, y) =_P (u, v)$ iff $x =_A u$ and $y =_B v$.

Standard notation for this P is $A \times B$. Projection function are $\pi_1(x, y) = x$ and $\pi_1(x, y) = y$. This construction can be verified to satisfy the abstract property (page 5). (It can as well be expressed by the categorical universal property for products.)

The disjoint union $A \mathbin{\dot\cup} B$ (or $A + B$) is definied by considering the corresponding type construction.

The functions from A to B form a set B^A defined to be the type

$$(\Sigma f : A \longrightarrow B)(\forall x, y : A)[x =_A y \longrightarrow f\,x =_B f\,y],$$

together with the equivalence relation

$$(f, p) =_{B^A} (g, q) \Longleftrightarrow_{\mathrm{def}} (\forall x : A)\, f\,x =_B g\,x.$$

The evaluation function $\mathrm{ev}_{A,B} : B^A \times A \longrightarrow B$ is given by

$$\mathrm{ev}_{A,B}((f, p), a) = f\,a$$

Proposition. Let A, B and X be sets. For every function $h : X \times A \longrightarrow B$ there is a unique function $\hat{h} : X \longrightarrow B^A$ with

$$\mathrm{ev}_{A,B}(\hat{h}(x), y) = h(x, y) \qquad (x \in X, y \in A).$$

A set X is called *discrete,* if for all $x, y \in X$

$$(x =_X y) \ \vee \ \neg \, (x =_X y).$$

In classical set theory all sets are discrete. This is not so constructively, but we have

Proposition. The unit set $\mathbf{1}$ and the set of natural numbers $\mathbb{N}$ are both discrete. If X and Y are discrete sets, then $X \times Y$ and $X + Y$ are discrete too.

However, the assumption that $\mathbb{N}^{\mathbb{N}}$ is discrete implies a nonconstructive principle (WLPO):

$$(\forall n \in \mathbb{N}) f(n) = 0 \ \vee \ \neg (\forall n \in \mathbb{N}) f(n) = 0$$

Coarser and finer equivalences

An equivalence relation $\sim$ is *finer* than another equivalence relation $\approx$ on a type A if for all $x, y : A$

$$x \sim y \implies x \approx y.$$

It is easy to prove by induction that $=_\mathbb{N}$ is the finest equivalence relation on $\mathbb{N}$.

If there is a finest equivalence relation $=_A$ on a type A, the set $A = (A, =_A)$ has the *substitutivity property*

$$x =_A y \implies (P\,x \Leftrightarrow P\,y)$$

for any predicate P on the type A.

Sets are rarely substitutive, and the notion is not preseved by isomorphisms. $\mathbb{Z}_n$ as constructed above is not substitutive; an isomorphic construction yields substitutivity.

Theorem. To any type A, the identity type construction Id assigns a finest equivalence $\mathrm{Id}\ A$. The resulting set, also denoted A, is substitutive.

Remark. Substitutive sets are however very convenient for direct formalisation in e.g. Agda or Coq, as extensionality proofs can be avoided.

5. Choice sets and axiom of choice

A set S is a *choice set,* if for any surjective function $f : X \longrightarrow S$, there is right inverse $g : S \longrightarrow X$, i.e.

$$f(g(s)) = s \qquad (s \in S).$$

Theorem. Every substitutive set is a choice set.

(Zermelo's) Axiom of Choice may be phrased thus:
Every set is a choice set.

Theorem. Zermelo's AC implies the law of excluded middle.

Though Zermelo's AC is incompatible with constructivism, there is related axiom (theorem of type theory) freely used in Bishop constructivism.

Theorem. For any set A there is a choice set A and surjective function $p : A \longrightarrow A$. (In categorical logic often referred to as "existence of enough projectives".)
As a consequence, Dependent Choice is valid (see notes, p. 76).

Theorem. If A and B are choice sets, then so are $A \times B$ and $A + B$.

6. Relations and subsets

Definition A *(extensional) property* P of the set X is a family of propositions $P\,x$ $(x \in X)$ with

$$x =_X y, P\,x \Longrightarrow P\,y.$$

We also say that P is a *predicate* on X.

A *relation* R between sets X and Y is a family of propositions $R\,x\,y$ $(x \in X, y \in Y)$ such that

$$x =_X x', y =_Y y', R\,x\,y \Longrightarrow R\,x'\,y'.$$

The relation is *univalent* if $y =_Y y'$, whenever $R\,x\,y$ and $R\,x\,y'$.

Write $P(x)$, $R(x, y)$ etc. in the extensional situation.

Restatement of choice principles for relations.

The following is the theorem of *unique choice.*

Thm. Let R be a univalent relation between the sets X and Y. It is total if, and only if, there exists a function $f : X \longrightarrow Y$, called a *selection function,* such that

$$R(x, f(x)) \qquad (x \in X).$$

(This function is necessarily unique if it exists.)

An alternative characterisation of choice sets is

Thm. A set X is a choice set iff for every set Y, each total relation R between X and Y has a selection function $g : X \longrightarrow Y$ so that

$$R(x, g(x)) \qquad (x \in X).$$

Dependent choice

Dependent choice. Let A be a set which is the surjective image of a choice set. Let R be a binary relation on A such that

$$(\forall x \in A)(\exists y \in A)R(x, y).$$

Then for each $a \in A$, there exists a function $f : \mathbb{N} \longrightarrow A$ with $f(0) = a$ and

$$R(f(n), f(n+1)) \qquad (n \in \mathbb{N}).$$

Proof. Let $p : P \longrightarrow A$ be surjective, where P is a choice set. By surjectivity, we have

$$(\forall u \in P)(\exists v \in P)R(p(u), p(v)).$$

Since P is a choice set we find $h : P \longrightarrow P$ with $R(p(u), p(h(u)))$ for all $u \in P$.

For $a \in A$, there is $b_0 \in P$ with $a = p(b_0)$.

Define by recursion $g(0) = b_0$ and $g(n+1) = h(g(n))$, and let $f(n) = p(g(n))$. Thus $R(p(g(n)), p(g(n+1)))$, so f is indeed the desired choice function. $\qquad \square$

Remark Thus we have proved the general dependent choice theorem in type theory with identity types. We also get another proof of countable choice, without requiring a particular subsititutive construction of natural numbers.

Subsets as injective functions

Let X be a set. A *subset* of X is a pair $S = (\partial S, \iota_S)$ where ∂S is a set and $\iota_S : \partial S \longrightarrow X$ is an injective function.

An element $a \in X$ is a *member of S* (written $a \in_X S$) if there exists $d \in \partial S$ with $a =_X \iota_S(d)$.

Inclusion $\subseteq_X$ and equality $\equiv_X$ of subsets of X can be defined in the usual logical way.

Prop. For subsets A and B of X, the inclusion $A \subseteq_X B$ holds iff there is a function $f : \partial A \longrightarrow \partial B$ with $\iota_B \circ f = \iota_A$. (Such f are unique and injective.)

The subsets are equal iff f is a bijection.

Separation of subsets

For a property P on a set X, the subset

$$\{x \in X \mid P(x)\} = \Big(\{x \in X : P(x)\}, \iota\Big)$$

is defined by the data:

$$\{x \in X : P(x)\} =_{\text{def}} (\Sigma x \in X)P(x)$$

and

$$\langle x, p \rangle =_{\{x \in X : P(x)\}} \langle y, q \rangle \Longleftrightarrow_{\text{def}} x =_X y$$

and $\iota(\langle x, p \rangle) =_{\text{def}} x$.

(Note the careful syntactic distinction of ":" and "|".)

Note that

$$
\begin{aligned}
a \in_X \{x \in X \mid P(x)\} &\Leftrightarrow (\exists d \in \{x \in X \,:\, P(x)\})\, a = \iota(d) \\
&\Leftrightarrow (\exists x \in X)(\exists p : P\, x)\, a = \iota(\langle x, p \rangle) \\
&\Leftrightarrow P\, a \\
&\Leftrightarrow P(a)
\end{aligned}
$$

The usual set-theoretic operations $\cap$, $\cup$, $\overline{(\ )}$ can now be defined "logically" for subsets.

A subset S of X is *decidable*, or *detachable*, if for all $a \in X$

$$
a \in_X S \ \lor \ \neg(a \in_X S).
$$

Union of subsets: logical definition.

Let $A = (\partial A, \iota_A)$ and $B = (\partial B, \iota_B)$ be subsets of X.
Their union is the following subset of X

$$A \cup B = \{z \in X \mid z \in_X A \text{ or } z \in_X B\}.$$

Taking $U = A \cup B$ apart as $U = (\partial U, \iota_U)$ we see that ∂U is

$$(\Sigma z : X)(z \in_X A \text{ or } z \in_X B) \;\; = \;\; (\Sigma z : X)((z \in_X A) + (z \in_X B)).$$

whereas $\iota_U(z, p) = z$.

Complement

The complement of the subset A of X is defined as

$$\overline{A} = \{z \in X \mid \neg z \in_X A\}.$$

For $\overline{A} = (\partial C, \iota_C)$ we have

$$\partial C = (\Sigma z : X)((z \in_X A) \longrightarrow \perp).$$

That A is a decidable subset of X can be expressed as $A \cup \overline{A} = X$.

The decidable subsets form a boolean algebra.

Partial functions

A *partial function* f from A to B consists of a subset $(\mathrm{D}_f, \mathrm{d}_f)$ of A, its *domain of definition* (denoted $\mathrm{dom}\, f$) and a function $\mathrm{m}_f : \mathrm{D}_f \longrightarrow B$. We write this with a special arrow symbol as $f : A \rightharpoonup B$.

Such $f : A \rightharpoonup B$ is *total* if its domain of definition equals A as a subset, or equivalently, if d_f is an isomorphism.

Another partial function $g : A \rightharpoonup B$ *extends* f, writing $f \subseteq g : A \rightharpoonup B$, if for each $s \in \mathrm{D}_f$ there exists $t \in \mathrm{D}_g$ with $\mathrm{d}_f(s) = \mathrm{d}_g(t)$ and $\mathrm{m}_f(s) = \mathrm{m}_g(t)$. If both $f \subseteq g$ and $g \subseteq f$, we define f and g to be equal as partial functions.

Example. Let $F = (F, \cdot, +, 0, 1)$ be a field, and let

$$U = \{x \in F \mid (\exists y \in F) x \cdot y = 1\}$$

be the subset of invertible elements. Define a function $\mathrm{m}_r : \partial U \longrightarrow F$ to be $\mathrm{m}_r(x) = y$, where y is unique such that $x \cdot y = 1$. Thus the reciprocal is a partial function $r = (\cdot)^{-1} : F \rightarrow F$.

In fact, for any univalent relation R between sets X and Y there is partial function $f_R = (D, d, m)$ given by

$$\partial D = \{u \in X \times Y : R(\pi_1(u), \pi_2(u))\}$$

$d = \pi_1 \circ \iota_D$ and $m = \pi_2 \circ \iota_D$.

Example For any pair of subsets A and B of X that are disjoint $A \cap B = \emptyset$, we may define a partial characteristic function

$$\chi : X \rightharpoonup \{0, 1\}$$

satisfying

$\chi(z) = 0$ iff $z \in_X A$,

$\chi(z) = 1$ iff $z \in_X B$,

by considering the univalent relation $R(z, n)$:

$$(z \in_X A \wedge n = 0) \vee (z \in_X B \wedge n = 1).$$

Caution The collection of partial functions $A \rightharpoonup B$ does in general not form a set in constructive type/set theory, even in the case $A = B = \{0\}$.

Partial functions are composed in the following manner: if $f : A \rightharpoonup B$ and $g : B \rightharpoonup C$, define the composition $h = g \circ f : A \rightharpoonup C$ by

$$\mathrm{D}_h = \{(s, t) \in \mathrm{D}_f \times \mathrm{D}_g : \mathrm{m}_f(s) = \mathrm{d}_g(t)\}$$

The function $\mathrm{d}_h : \mathrm{D}_h \longrightarrow A$ given by composing the projection to D_f with d_d is injective. The function $\mathrm{m}_h : \mathrm{D}_h \longrightarrow C$ is defined by the composition of the projection to D_g and d_g.

Example (Bishop and Bridges 1985, p. 222)
A real-valued *integrable function* f on $[0, 1]$ is a partial function
$[0, 1] \rightarrow \mathbb{R}$ which is strongly extensional, i.e. $f(x) \neq f(y)$ implies
$x \neq y$, and for which there exists a sequence $f_n : [0, 1] \longrightarrow \mathbb{R}$ of
uniformly continuous functions where

$$\sum_{n=1}^{\infty} \int_0^1 |f_n(x)| dx$$

converges and where

$$f(x) = \sum_{n=1}^{\infty} f_n(x)$$

whenever $\sum_{n=1}^{\infty} |f_n(x)|$ converges.

$A \subseteq [0, 1]$ is *full* if is included in domain of an integrable function as
above. (Constructive counterpart of the complement of a null set.)

7. Finite sets and their relatives

The *canonical n-element set* is

$$\mathbb{N}_n = \{k \in \mathbb{N} : k < n\} \hookrightarrow \mathbb{N}.$$

Any set X isomorphic to such a set is called *finite.* It may be written

$$\{x_0, \ldots, x_{n-1}\}$$

where $k \mapsto x_k : \mathbb{N}_n \longrightarrow X$ is the isomorphism.

Since $x_j = x_k$ iff $j = k$, we can always decide whether two elements of a finite set are equal by comparison of indices.

A related notion is more liberal:

A set X is called *subfinite,* or *finitely enumerable,* if there is, for some $n \in \mathbb{N}$, a surjection $x : \mathbb{N}_n \longrightarrow X$.

Here we are only required to enumerate the elements, not tell them apart.

We can always tell whether a subfinite set is empty by checking if $n = 0$.

Remark. A subset of a finite set need not be finite, or even subfinite. Consider

$$\{0 \in \mathbb{N}_1 : P\}$$

where P is some undecided proposition.

Some basic properties

Let X and Y be sets. Then:

(i) X finite $\iff$ X subfinite and discrete

(ii) X subfinite, $f : X \longrightarrow Y$ surjective $\implies Y$ subfinite

(iii) Y discrete, $f : X \longrightarrow Y$ injective $\implies X$ discrete

(iv) Y discrete, $X \hookrightarrow Y \implies X$ discrete

(v) Y finite, $X \hookrightarrow Y$ decidable $\implies X$ finite.

8. Quotients

Let $X = (X, =_X)$ be a set and let $\sim$ be a relation on this set. Then by the extensionality of the relation

$$x =_X y \Longrightarrow x \sim y. \tag{1}$$

Thus if $\sim$ is an equivalence relation on X

$$X/\!\sim \; = (X, \sim)$$

is a set, and $q : X \longrightarrow X/\!\sim$ defined by $q(x) = x$ is a surjective function.

We have the following extension property. If $f : X \longrightarrow Y$ is a function with

$$x \sim y \implies f(x) =_Y f(y), \qquad (2)$$

then there is a unique function $\overline{f} : X/\!\!\sim \; \longrightarrow Y$ (up to extensional equality) with

$$\overline{f}(i(x)) =_Y f(x) \qquad (x \in X).$$

We have constructed *the quotient of X with respect to* $\sim$:
$q : X \longrightarrow X/\!\!\sim$

Remark. Every set is a quotient of a choice set. Namely, X is the quotient of X w.r.t. $=_X$.

Proposition. A set is subfinite iff it is the quotient of a finite set.

9. Universes and restricted powersets

A general problem with (or feature of) predicative theories like Martin-Löf type theory is their inability to define a set of *all* subsets of a given set. It is, though, often sufficient to consider certain restricted classes of subsets in a certain situation.

A *set-indexed family $\mathcal{F} = (F, I)$ of subsets* of a given set X consists of an *index set $I = (I, =_I)$* and a subset F_i of X for each $i : I$, which are such that if $i =_I j$ then F_i and F_j are equal as subsets of X.

A subset S of X *belongs to the family $\mathcal{F}$*, written $S \in \mathcal{F}$, if $S = F_i$ (as subsets of X) for some $i \in I$.

Consider any family of types $\mathcal{U} = (T, U)$, where $T\, i$ is a type for each $i : U$. It represents a collection of sets, the $\mathcal{U}$-sets, as follows.

First, a *$\mathcal{U}$-representation* of a set is a pair $r = (i_0, e)$ where $i_0 : I$ and $e : T\, i_0 \times T\, i_0 \longrightarrow U$ is an operation so that

$$a =_r b \Leftrightarrow_{\mathrm{def}} T(e\, a\, b)$$

defines an equivalence relation on the type $T\, i_0$. Then this is a set

$$\hat{r} = (T\, i_0, =_r).$$

A set X is *$\mathcal{U}$-representable,* or simply a *$\mathcal{U}$-set*, if it is in bijection with $\hat{r}$ for some $\mathcal{U}$-representation r. The $\mathcal{U}$-sets defines, in fact, a full subcategory of the category of sets, equivalent to a small category.

Example For $U = \mathrm{N}$ and $T\, n = \mathrm{N}_n$, the $(\mathrm{N}, \mathrm{N}_{(-)})$-sets are the finite sets.

Restricted power sets

For any set X and any family of types $\mathcal{U}$, define the family $\mathcal{R}_{\mathcal{U}}(X)$ of subsets of X as follows.

- Its index set I consists of triples (r, m, p) where r is a $\mathcal{U}$-representation, $m : \hat{r} \longrightarrow X$ is a function and p is a proof that m is injective.

- Two such triples (r, m, p) and (s, n, q) are equivalent, if $(\hat{r}, m)$ and $(\hat{s}, n)$ are equal as subsets.

- For index $(r, m, p) \in I$, the corresponding subset of X is $F_{(r,m,p)} = (\hat{r}, m)$.

Proposition A subset $S = (\partial S, \iota_S)$ of X belongs to $\mathcal{R}_{\mathcal{U}}(X)$ iff ∂S is a $\mathcal{U}$-set.

Unless $\mathcal{U}$ has some closure properties, $\mathcal{R}_{\mathcal{U}}(X)$ will not be closed under usual set-theoretic operations. We review some common such properties below. Suppose that $\mathcal{U}$ is a type-theoretic universe.

- If $\mathcal{U}$ is closed under Σ, then $\mathcal{R}_{\mathcal{U}}(X)$ is closed under binary $\cap$, and $\bigcup_{i \in I}$ indexed by $\mathcal{U}$-sets I.
- If $\mathcal{U}$ is closed under Π, then $\mathcal{R}_{\mathcal{U}}(X)$ is closed under $\bigcap_{i \in I}$ indexed by $\mathcal{U}$-sets I, and the binary set operation

$$(A \Rightarrow B) = \{x \in X : x \in A \Rightarrow x \in B\}.$$

- If $\mathcal{U}$ is closed under $+$, then $\mathcal{R}_{\mathcal{U}}(X)$ is closed under binary $\cup$.
- If $\mathcal{U}$ contains an empty type, then $\mathcal{R}_{\mathcal{U}}(X)$ contains $\emptyset$.

Standard Martin-Löf type universes $\mathcal{U}$ (see Martin-Löf 1984) satisfies indeed the conditions above.

$$
\begin{array}{rclcrcl}
\hat{N} & : & U & & T\,\hat{N} & = & N \\
\widehat{N_0} & : & U & & T\,\widehat{N_0} & = & N_0 \\
\widehat{N_1} & : & U & & T\,\widehat{N_1} & = & N_1 \\
(\hat{+}) & : & U \longrightarrow U \longrightarrow U & & T\,(a\,\hat{+}\,b) & = & T\,a + T\,b \\
\hat{\Sigma} & : & & & T\,(\hat{\Sigma}\,a\,b) & = & \Sigma\,(T\,a)\,(\lambda x.T\,(bx)) \\
\hat{\Pi} & : & & & T\,(\hat{\Pi}\,a\,b) & = & \Pi\,(T\,a)\,(\lambda x.T\,(bx)) \\
& \vdots & & & & \vdots &
\end{array}
$$

10. Categories

We first use a definition of category where no equality relation between objects is assumed, as introduced in type theory by P. Aczel 1993, P. Dybjer and V. Gaspes 1993. Such categories are adequate for developing large parts of elementary category theory inside type theory (Huet and Saibi 2000).

A *small E-category* **C** consists of a type Ob of *objects* (no equivalence relation between objects is assumed) and for all $A, B : \mathrm{Ob}$ there is a set $\mathrm{Hom}(A, B)$ of *morphisms from A to B*. There is a identity morphism $\mathrm{id}_A \in \mathrm{Hom}(A, A)$ for each $A : \mathrm{Ob}$. There is a composition function $\circ : \mathrm{Hom}(B, C) \times \mathrm{Hom}(A, B) \longrightarrow \mathrm{Hom}(A, C)$. These data satisfy the equations $\mathrm{id} \circ f = f$, $g \circ \mathrm{id} = g$ and $f \circ (g \circ h) = (f \circ g) \circ h$.

For a *locally small E-category* we allow Ob to be a sort.

Example. *The category of sets,* **Sets**, has as objects sets. The set of functions from A to B is denoted $\mathrm{Hom}(A, B)$. The category **Sets** is locally small, but not small.

Example. The *discrete category given by a set $A = (A, =_A)$.* The objects of the category are the elements of A. Define $\mathrm{Hom}(a, b)$ as the type (of proofs of) $a =_A b$. Any two elements of this type are considered equal. (The proofs of reflexivity and transitivity provide id and $\circ$ respectively. Also the proof of symmetry, gives that two objects a and b are isomorphic if, and only if, $a =_A b$.) Denote the discrete category by $A^{\#}$. This is a small category.

10.2 Categories with equality on objects

Families of sets have more structure than in set theory.

A *family F of sets indexed by a set I* is a functor $F : I^{\#} \longrightarrow \textbf{Sets}$.

Explication:
For each element a of I, $F(a)$ is a set.

For any proof object $p : a =_I b$, $F(p)$ is function from $F(a)$ to $F(b)$, a so-called *transporter function*.

Moreover, since any two morphisms p and q from a to b in $I^{\#}$ are identified, we have $F(p) = F(q)$. The functoriality conditions thus degenerate to the following:

(a) $F(p) = \mathrm{id}_{F(a)}$ for any $p : a =_I a$.

(b) $F(q) \circ F(p) = F(r)$ for all $p : a =_I b$, $q : b =_I c$, $r : a =_I c$.

Note that each $F(p)$ is indeed an isomorphism, and that $F(q)$ is the inverse of $F(p)$ as soon as $p : a =_I b$ and $q : b =_I a$.

Remark. If each set in the family F is a subset of a fixed set X, i.e. $i_a : F(a) \hookrightarrow X$ and so that $i_a \circ F(r) = i_b$ for $r : a =_I b$, then $(F(a), i_a) = (F(b), i_b)$ as subsets of X, if $a =_I b$.

Remark. Families of sets are treated in essentially this way in (Bishop and Bridges 1985, Exercise 3.2).

Two constructions of categories

A family F of sets over a set I gives rise to a category of sets $\mathcal{C} = \mathcal{C}(I, F)$ as follows. The objects are given by the index set $\mathcal{C}_0 = I$, and are thus equipped with equality, and the set of arrows $\mathcal{C}_1$ is

$$((\Sigma i, j : |I|)\mathrm{Ext}(F(i), F(j)), \sim)$$

which, thus, consists of triples (i, j, f) where $f : F(i) \longrightarrow F(j)$ is an extensional function, and where two arrows are equal $(i, j, f) \sim (i', j', f')$ if, and only if, there are proof objects $p : i =_I i'$ and $q : j =_I j'$ such that the diagram

$$
\begin{array}{ccc}
F(i) & \xrightarrow{\ f\ } & F(j) \\
{\scriptstyle F(p)}\big\downarrow & & \big\downarrow{\scriptstyle F(q)} \\
F(i') & \xrightarrow[\ f'\ ]{} & F(j')
\end{array}
\tag{3}
$$

commutes.

The second construction is as follows. Define a category $\mathcal{S}(I, F)$ whose set of objects is I, and whose arrows[3] are triples (i, j, R) where R is functional binary relation on $S = \Sigma(I, F)$ with $\mathrm{dom}(R) \doteq F(i)$ and $\mathrm{ran}(R) \subseteqdot F(j)$. Two arrows (i, j, R) and (i', j', R') are equal when $i =_I i'$, $j =_I j'$ and $R \doteq R'$. The domain and codomain of (i, j, R) are i and j respectively. The composition of (i, j, R) and (j', k, Q) is $(i, k, Q \circ R)$ when $j =_I j'$. Here $Q \circ R$ denotes the relational composition.

Theorem. (P. 2013) $\mathcal{S}(I, F) \cong \mathcal{C}(I, F)$ $\square$

---

[3]The triples actually form a set since they can be represented by graphs of functions, as the isomorphism theorem shows later.

11. Constructive set theory CZF as part of Bishop's set theory

Aczel's standard model of CZF: for a universe U, $T(\cdot)$, the set-theoretic universe V is inductively defined by the rules

$$\frac{a : U \quad f : T(a) \longrightarrow V}{\sup(a, f) : V}.$$

The equality $=_V$ is the smallest relation satisfying the two rules

$$\frac{\forall x : T(a).\exists y : T(b).f(x) =_V g(y) \quad \forall y : T(b).\exists x : T(a).f(x) =_V g(y)}{\sup(a, f) =_V \sup(b, g)}$$

Now $V = (V, =_V)$ is a set (setoid!)
The membership relation is defined by

$$u \in_V \sup(a, f) \Longleftrightarrow \exists x : T(a).u =_V f(x)$$

For $u : V$ define the set

$$B(u) = (|B(u)|, =_{B(u)})$$

of elements of V belonging to u by letting

$$|B(u)| = \Sigma z : V . z \in_V u$$

and

$$(z, p) =_{B(u)} (z', p') \iff z =_V z'. \tag{4}$$

Note that for a set $u = \sup(a, f)$, it holds that

$$B(\sup(a, f)) \cong (T(a), \sim_f)$$

where

$$x \sim_f x' \iff f(x) =_V f(x').$$

We define therefore

$$R(\sup(a, f)) = (T(a), \sim_f).$$

A set A is *V-representable* iff there is some $u : V$ and a bijection $\phi : A \cong R(u)$.

Let $u = \sup(a, f)$ and $v = \sup(b, g)$. If we examine

$$\mathrm{Ext}(R(u), R(v)),$$

the standard construction of the set of functions from $R(u)$ to $R(v)$, it has the underlying type

$$\Sigma h : T(a) \longrightarrow T(b).(\forall x, y : T(a)(fx =_V fy \Rightarrow h(gx) =_V h(gy))) \tag{5}$$

and equality $\sim$ defined by

$$(h, p) \sim (h', p') \text{ iff } \forall x : T(a).h(gx) =_V h'(gx).$$

Let $F_{u,v}$ denote the type in (5). Define

$$\gamma(h, p) = \sup(a, \lambda x.\langle fx, h(gx)\rangle)$$

which gives the graph of the function h, when $(h, p) : F_{u,v}$. Suppose that the type $F_{u,v}$ has a code $\varphi_{u,v}$ in U so that $F_{u,v} = T(\varphi_{u,v})$.

Now we can form
$$v^u = \sup(\varphi_{u,v}, \gamma),$$
which is the set all of functions from u to v. Indeed we have

$z \in_V v^u$ iff z is a total and functional relation from u to v,

where the latter can be formally expressed as the conjunction of the following statements

$$(\forall t \in V)(t \in_V z \Rightarrow (\exists x, y \in V)(x \in_V u \wedge y \in_V v \wedge t =_V \langle x, y \rangle)),$$

$$(\forall x \in V)(x \in_V u \Rightarrow (\exists y \in V)(y \in_V v \wedge \langle x, y \rangle \in_V z)),$$

$$(\forall x, y, y' \in V)(\langle x, y \rangle \in_V z \wedge \langle x, y' \rangle \in_V z \Rightarrow y =_V y').$$

We have the following bijective correspondence
For any $u = \sup(a, f)$, $v = \sup(b, g) \in V$, there is a bijection

$$\psi : R(v^u) \longrightarrow \mathrm{Ext}(R(u), R(v))$$

given by $\psi(h, p) = (h, p)$. $\square$

The internal category of sets in V may be described as follows. Define the category $\mathcal{V}$ to have as objects $\mathcal{V}_0$ the setoid $V = (V, =_V)$. The arrows $\mathcal{V}_1$ has as underlying type

$$\Sigma u \in V.\mathrm{Isarrow}(u)$$

where $\mathrm{Isarrow}(u)$ is the predicate

$$\exists a, b, f \in V.u =_V \langle\langle a, b\rangle, f\rangle \wedge f \text{ is a total and functional relation from } a$$

Equality $(u, p) =_{\mathcal{V}_1} (u', p')$ is defined to be $u =_V u'$. The setoid $\mathcal{V}_2$ of composable arrows has for underlying type

$$\Sigma w \in V.\Sigma u, v \in \mathcal{V}_1.w =_V \langle \pi_1(u), \pi_1(v)\rangle \wedge \mathrm{cod}\, u =_v \mathrm{dom}\, v$$

and its equality is given by $(w, p) \sim (w', p')$ iff $w =_V w'$.
Composition cmp of arrows is obtained by composition of relations in the usual set-theoretic way.

Theorem
$\mathcal{V}$ *is a category.* $\quad\square$

A different category is constructed using the method above. We extend $R(\cdot)$ to a family of sets $\bar{R}$ over the set $V = (V, =_V)$.

Lemma $\bar{R}$ is a family of sets over $(V, =_V)$.

From the family $(V, \bar{R})$, we may construct the category $\mathcal{C} = \mathcal{C}(V, \bar{R})$, and, then compare it to the category $\mathcal{V}$ above. The objects of the two categories are give by the same set. Let $F_0 : \mathcal{C}_0 \longrightarrow \mathcal{V}_0$ be the identity map. There is a bijection $\mathcal{C}_1 \longrightarrow \mathcal{V}_1$ given by

$$(a, b, f) \mapsto \langle \langle a, b \rangle, \gamma(|f|, \mathrm{ext}_f) \rangle.$$

Further, this yields a bijection $F_2 : \mathcal{C}_2 \longrightarrow \mathcal{V}_2$ by letting F_1 act on the two component arrows. It is then straightforward to verify that F_0, F_1 and F_2 form a functor which is an isomorphism. We have

Theorem
The categories $\mathcal{C}(V, \bar{R})$ and $\mathcal{V}$ are isomorphic. $\square$

Conclusion:

1) V-representable sets constitute a good category.

2) We may use proofs in CZF+REA+RDC via the model V to prove results about V-representable sets in Coq.

3) There are predicatively acceptable axioms such sREA which seems to require stronger type theories than Coq.

12. Relation to categorical logic

Category theory provides an abstract way of defining the essential mathematical properties of sets, in terms of universal constructions.

An *elementary topos* is a category with properties similar to the sets, though neither classical logic (discreteness of sets), or axioms of choice are assumed among these properties.

C. McLarty: *Elementary Categories, Elementary Toposes*. Oxford University Press 1992.

J. Lambek and P.J. Scott: *Introduction to Higher-Order Categorical Logic*. Cambridge University Press 1986.

Also predicative versions of toposes have been developed

I. Moerdijk and E. Palmgren: Type Theories, Toposes and Constructive Set Theory, *Annals of Pure and Applied Logic* 114(2002).

A constructive, predicative version of Lawvere's ETCS is available:

E. Palmgren: Constructivist and Structuralist Foundations: Bishop's and Lawvere's Theories of Sets. *Annals of Pure and Applied Logic* 163(2012). (Preprint version: arXiv:1201.6272v1).

References

📄 P. Aczel and M. Rathjen (2001). *Notes on Constructive Set Theory*, Report No. 40, Stockholm: Institut Mittag-Leffler, Royal Swedish Academy of Sciences.

📄 P. Aczel and M. Rathjen (forthcoming). *Constructive Set Theory*.

📄 Y. Bertot, *Coq in a Hurry*. http://cel.archives-ouvertes.fr/inria-00001173

📄 E. Bishop and D.S. Bridges (1985). *Constructive Analysis*. Springer.

📄 E. Bishop (1967). *Foundations of Constructive Analysis*. McGraw-Hill.

📄 D.S. Bridges and F. Richman (1987). *Varieties of Constructive Mathematics*. London Mathematical Society Lecture Notes, Vol. 97. Cambridge University Press.

📄 The Coq Proof Assistant. http://coq.inria.fr

📄 T. Coquand, P. Dybjer, E. Palmgren and A. Setzer. *Type-theoretic foundation of constructive mathematics.* Notes distributed at TYPES Summer School, Göteborg, August 2005.

📄 N. Greenleaf (1981). Liberal Constructive Set Theory. In: F. Richman (ed.), *Constructive Mathematics, Proceedings of the Conference Held at Las Cruces, New Mexico, August 11–15, 1980,* Lecture Notes in Mathematics, vol. 873, Springer.

📄 H. Lombardi and C. Quitté (2011). *Algèbre Commutative. Méthodes constructives.* Calvage et Mounet.

📄 M.E. Maietti. Modular correspondence between dependent type theories and categories including pretopoi and topoi. *Mathematical Structures in Computer Science* 15(2005), 1089–1149.

📄 P. Martin-Löf (1984). *Intuitionistic Type Theory.* Notes by Giovanni Sambin of a series of lectures given in Padua, June 1980. Bibliopolis.

📄 P. Martin-Löf (2006). 100 years of Zermelo's axiom of choice: what was the problem with it? *The Computer Journal* (2006) 49 (3): 345–350.

📄 R. Mines, F. Richman, and W. Ruitenburg (1988), *A Course in Constructive Algebra,* Universitext, Heidelberg: Springer Verlag.

📄 J. Myhill (1973). Some Properties of Intuitionistic Zermelo-Fraenkel Set Theory, in *Cambridge Summer School in Mathematical Logic,* A. Mathias and H. Rogers (eds.), Lecture Notes in Mathematics, 337, Heidelberg: Springer Verlag, 206–231.

📄 B. Nordström, K. Peterson and J.M. Smith (1990). *Programming in Martin-Löf 's Type Theory.* Oxford University Press. (Also available at URL: www.cse.chalmers.se/research/group/logic/book/)

📄 E. Palmgren and O. Wilander (2013). Constructing categories and setoids of setoids in type theory. Preprint. http://people.su.se/~epalm/

📄 E. Palmgren (2013).Yet another category of setoids with equality on objects. Preprint. http://people.su.se/~epalm/