J Autom Reasoning (2015) 55:199–210
DOI 10.1007/s10817-015-9340-6

Reconsidering Pairs and Functions as Sets

Chad E. Brown[1]

Received: 27 August 2013 / Accepted: 29 July 2015 / Published online: 18 September 2015

Abstract We give representations for ordered pairs and functions in set theory with the property that ordered pairs are functions from the finite ordinal 2. We conjecture that these representations are useful for formalized mathematics since certain isomorphic sets are identified. The definitions, theorems and proofs have been formalized in the proof assistant Coq using only the simply typed features of Coq. We describe the development within the context of an intuitionistic simply typed (higher-order) version of (well-founded) Zermelo-Fraenkel set theory without the axiom of infinity.

Keywords Set theory · Higher-order logic · Simple type theory · Higher-order theorem proving · Ordered pairs · Functions

1 Introduction

A foundation for mathematics must support the basic building blocks of the mathematical universe. Among these basic building blocks are numbers, pairs, sets and functions. A common foundation for mathematics is Zermelo-Fraenkel set theory (ZF) [23]. Sets are the only innate basic building blocks in ZF. However, there are well-known constructions for numbers, pairs and functions. The finite von Neumann ordinals in which $n = \{0, \ldots, n-1\}$ give a common representation for natural numbers. Kuratowski's representation of pairs (x, y) as $\{\{x\}, \{x, y\}\}$ is popular and it is common to identify functions with their graphs. For example, these are the representations used in the proof assistants Isabelle-ZF [16] and Mizar [3, 4, 7, 21].

Given sets X and Y, $X \times Y$ is notation for the set of pairs with components from X and Y and X^Y is notation for the set of functions from Y to X. Sometimes the notation for $X \times X$

✉ Chad E. Brown
idonotuseemail@mailinator.com

[1] Independent Researcher, mathgate.info/cebrown/contact.php

is simplified to be X^2. However, if 2 is the finite ordinal $\{0, 1\}$, then we have an ambiguity. Is X^2 notation for the set $X \times X$ of pairs or for the set $X^{\{0,1\}}$ of functions from $\{0, 1\}$ to X? Mathematically, this is not a serious ambiguity. There is an obvious isomorphism between the sets $X \times X$ and $X^{\{0,1\}}$ that respects the relevant structures. In particular, functions $f \in X^{\{0,1\}}$ can be mapped to pairs $(f0, f1)$ and pairs $(x, y) \in X \times X$ can be mapped to a function $f \in X^{\{0,1\}}$ such that $f0 = x$ and $f1 = y$. Mathematicians are justified in thinking of $X \times X$ and $X^{\{0,1\}}$ as being, essentially, the same sets.

When formalizing mathematics, there are drawbacks to having different isomorphic representations for such a basic operation as pairing. In Mizar's library, Kuratowski pairs are defined in the document giving the axioms of the set theory [21]. Later, finite sequences are defined leading to a different notion of pairing defined as a function from $\{1, 2\}$ [6]. Once one has these two definitions, there is always the question of *which* notion of pairing is appropriate in different situations. An expert will choose one or the other depending on the situation. For example, Definition 11 of the Mizar article [5] has the form $[0, \langle a, b\rangle]$ meaning the Kuratowski pair of 0 and the function mapping 1 to a and 2 to b.[1] This also means that theorems may be formulated using the different pairing operations. In order to apply the theorem when the pairing operations do not match, one would need to explicitly apply an isomorphism. The need to apply isomorphisms cannot be altogether avoided, but in cases involving such basic objects as pairs and functions it is worthwhile to consider representations chosen so that many such isomorphic sets will be equal.

We will describe representations for pairs and functions with the property that the set $X \times X$ will be the same as $X^{\{0,1\}}$. That is, pairs will be functions with domain $\{0, 1\}$. A pair u will be equal to $(u0, u1)$ where $u0$ is u applied to 0 and $u1$ is u applied to 1. We will continue to use the representation of natural numbers as finite ordinals. The representation of functions we will use is one due to Aczel [1]. The representation of pairs is similar to one considered by Morse [13] and can be informally described as being the disjoint union of the two sets. Morse ultimately used a different version of ordered pairs in [13].

Our constructions will work with both finite and infinite sets, but we will never need to make explicit use of infinite sets. Consequently, we work with a variant of ZF without the axiom of infinity. Instead of formulating the set theory in first order logic, we formulate it in simple type theory (higher-order logic). We include a description operator at the base type. Our simple type theory is intuitionistic. Since we are working in an intuitionistic setting, the set theory axioms must be chosen carefully. For example, the usual axiom of regularity implies excluded middle, and so it must be replaced by an $\in$-induction axiom. The axioms we choose are essentially those of $\mathtt{ZFI}_\mathtt{R}$ as described in [19] (a system first studied by Myhill [14]) except that infinity is omitted and the axioms are translated into the simply typed setting. The representation in simple type theory is similar to the version in Isabelle-ZF [16], but with higher-order aspects similar to those in [2, 15]. One benefit of using simple type theory is that we can quantify over predicates and (meta-level) functions. Consequently, the axioms of replacement, separation and $\in$-induction can each be stated with a single formula instead of using a schema of formulas. Another advantage of using

[1] Actually, in the pdf versions of the Mizar articles, the notation for Kuratowski pairs uses bold $\langle$ and $\rangle$.

simple type theory is that we can prove that we can define (meta-level) functions by $\in$-recursion before defining an object-level notion of pairing. In fact, $\in$-recursion will be used to define pairing.

In Section 2 we give the simply typed set theory. In Section 3 we give a few basic definitions and results, including an $\in$-recursion theorem. In Section 4 we specify what we require of our representation of pairs and functions. In Section 5 we define pairing and prove a number of results, and in Section 6 we do the same for functions. In Section 7 we define dependent sums and products (sets of pairs and functions).

The material described here has been formalized in the Coq theorem prover [12] using only the simply typed features of Coq.

2 Intuitionistic Simply Typed Set Theory

We briefly describe an intuitionistic form of Church's simple type theory [8, 9]. We start with two base types ι (the type of sets) and o (the type of propositions). Other (simple) types are function types: given two types σ and τ, the function type $\sigma\tau$ is a type (the type of functions from σ to τ). Let $\mathscr{T}$ be the set of types.

Let $(\mathscr{V}_\sigma)_{\sigma\in\mathscr{T}}$ be a disjoint family of infinite sets of variables. We use metavariables such as $x, y, z, \ldots$ to range over variables. We also have a set $\mathscr{C}_\sigma$ of constants of type σ. We will describe the specific constants in each $\mathscr{C}_\sigma$ shortly. We use the metavariable c to range over constants.

For each type σ there is a set Λ_σ of all terms of type σ. We use metavariables s, t and u to range over terms. This family of sets is defined inductively as follows: If $x \in \mathscr{V}_\sigma$, then $x \in \Lambda_\sigma$. If $c \in \mathscr{C}_\sigma$, then $c \in \Lambda_\sigma$. If $s \in \Lambda_{\sigma\tau}$ and $t \in \Lambda_\sigma$, then $(st) \in \Lambda_\tau$. If $x \in \mathscr{V}_\sigma$ and $t \in \Lambda_\tau$, then $(\lambda x.t) \in \Lambda_{\sigma\tau}$. If $s \in \Lambda_o$ and $t \in \Lambda_o$, then $(s \to t) \in \Lambda_o$. If $x \in \mathscr{V}_\sigma$ and $t \in \Lambda_o$, then $(\forall x.t) \in \Lambda_o$.

When $s \in \Lambda_\sigma$, we say s is a *term of type* σ. If s is a term of type o, we say s is a *formula*. When writing terms we omit parentheses whenever possible under the following conventions. Application associates to the left, so that stu means $((st)u)$. Implication associates to the right, so that $s \to t \to u$ means $(s \to (t \to u))$. Application binds more tightly than implication, so that $st \to su$ means $((st) \to (su))$. The scope of binders λ and $\forall$ is as far to the right as possible, so that $\lambda x.st$ means $(\lambda x.(st))$ and $\forall x.s \to t$ means $(\forall x.(s \to t))$. We write $\lambda x_1 \cdots x_n.s$ for $\lambda x_1.\cdots.\lambda x_n.s$ and $\forall x_1 \cdots x_n.s$ for $\forall x_1.\cdots.\forall x_n.s$. We will usually not be explicit about the types of variables if the intended type can be determined. Later we will introduce more notational conventions for terms when convenient.

We say two terms are α-equivalent if they are the same up to the names of bound variables. For example, $\lambda xy.xy$ is α-equivalent to $\lambda yx.yx$. We will treat α-equivalent terms as being equal. We assume the usual notion of free variables and let $\mathscr{F}s$ denote the (finite) set of variables that occur free in s. We denote the capture-avoiding substitution of t for the free occurrences of x in s by s_t^x. A *β-redex* is a term of the form $(\lambda x.s)t$. The *β-reduct* of $(\lambda x.s)t$ is s_t^x. An *η-redex* is a term of the form $\lambda x.sx$ where $x \notin \mathscr{F}s$. The *η-reduct* of $\lambda x.sx$ is s. We say s is a redex with reduct t if either s is a β-redex with β-reduct t or s is an η-redex with η-reduct t. We say s *one-step reduces* to t if there is a redex as a subterm of s and t is the result of replacing the redex with its reduct. *Convertibility* is simply the reflexive, symmetric, transitive closure of one-step reducibility. We write $s \approx t$ if s and t are convertible.

A *context* is a set of formulas. We use Γ to range over contexts. We define $\mathscr{F}\Gamma$ to be $\bigcup_{s\in\Gamma}\mathscr{F}s$. The natural deduction calculus given by the rules in Fig. 1 define when $\Gamma \vdash s$ holds for a context Γ and formula s.

What we have developed so far is a general intuitionistic simple type theory. Since our only interest is a specific instance corresponding to a set theory, we now fix the constants in the sets $\mathscr{C}_\sigma$. We will only have seven constants:

- d is a constant of type $(\iota o)\iota$. This will be used as a description operator.
- $\in$ is a constant of type $\iota\iota o$. This will be used to represent membership. We write formulas $\in\ s\ t$ using infix notation as $s \in t$. As infix notation, we assume $\in$ binds more tightly than implication but less tightly than application, so that $st \in tu \to st \in tu$ means $(((st) \in (tu)) \to ((st) \in (tu)))$.
- $\emptyset$ is a constant of type ι. This will be used as the empty set.
- $\bigcup$ and $\wp$ are constants of type $\iota\iota$. For $s \in \Lambda_\iota$, the term $\bigcup s$ will correspond to the union of the set s and the term $\wp s$ will correspond to the power set of s.
- s is a constant of type $\iota(\iota o)\iota$. This will correspond to sets formed by separation. We will use the notation $\{x \in s|t\}$ to represent the term $\mathsf{s}s(\lambda x.t)$.
- r is a constant of type $\iota(\iota\iota)\iota$. This will correspond to sets formed by replacement. We will use the notation $\{t|x \in s\}$ to represent the term $\mathsf{r}s(\lambda x.t)$.

We next define false, negation, conjunction, disjunction, equivalence, equality, existential and unique existential quantification. It is well-known that such operators can be defined in such a type theory. Russell indicated how to make some of the definitions [18], and most of the rest can be found in Prawitz [17]. Let $\bot$ be the formula $\forall q.q$. Let $\neg$ be the term $\lambda p.p \to \bot$ of type oo. Let $\wedge$ be the term $\lambda pq.\forall r.(p \to q \to r) \to r$ of type ooo. Let $\vee$ be the term $\lambda pq.\forall r.(p \to r) \to (q \to r) \to r$ of type ooo. Let $\equiv$ be the term $\lambda pq.\wedge(p \to q)(q \to p)$ of type ooo. We use infix notation for $\wedge$, $\vee$ and $\equiv$. We assume application and $\in$ bind more tightly than $\wedge$, $\wedge$ binds more tightly than $\vee$, $\vee$ binds more tightly than $\to$, and $\to$ binds more tightly than $\equiv$. We also write $s \notin t$ for $\neg(s \in t)$ and assume $\notin$ has the same binding strength as $\in$.

Equality can be defined at every type, but we will only need it at the base type ι. Let $=$ be the term $\lambda xy.\forall p.px \to py$ of type $\iota\iota o$. We use infix notation for $=$, and assume $=$ has the same binding strength as $\in$. We also write $s \neq t$ for $\neg(s = t)$.

Existential quantification can also be defined at every type. We will only need it at two types: ι and $\iota\iota$. Let $\mathbf{E}$ be the term $\lambda q.\forall p.(\forall x.qx \to p) \to p$ of type $(\iota o)o$. Let $\mathbf{E}^F$ be the term $\lambda Q.\forall p.(\forall F.QF \to p) \to p$ of type $((\iota\iota)o)o$. We write $\exists x.s$ for $\mathbf{E}(\lambda x.s)$ when $x \in \mathscr{V}_\iota$ and for $\mathbf{E}^F(\lambda x.s)$ when $x \in \mathscr{V}_{\iota\iota}$. We also define unique existential quantification at type ι. Let $\mathbf{E}^!$ be the term $\lambda q.(\exists x.qx) \wedge \forall xy.qx \to qy \to x = y$. We write $\exists!x.s$ for $\mathbf{E}^!(\lambda x.s)$.

Let $\subseteq$ be the term $\lambda XY.\forall x.x \in X \to x \in Y$ of type $\iota\iota o$. We use infix notation for $\subseteq$, and assume $\subseteq$ has the same binding strength as $\in$ and $=$.

We can now state the axioms of our intuitionistic simply typed set theory. We give the axioms as a context Γ_{a} consisting of the following formulas:

$$\frac{}{\Gamma \vdash s}\ s \in \Gamma \qquad \frac{\Gamma \vdash s}{\Gamma \vdash t}\ s \approx t \qquad \frac{\Gamma \cup \{s\} \vdash t}{\Gamma \vdash s \to t} \qquad \frac{\Gamma \vdash s \to t \quad \Gamma \vdash s}{\Gamma \vdash t}$$

$$\frac{\Gamma \vdash s_y^x}{\Gamma \vdash \forall x.s}\ x \in \mathscr{V}_\sigma, y \in \mathscr{V}_\sigma \setminus \mathscr{F}\Gamma \qquad \frac{\Gamma \vdash \forall x.s}{\Gamma \vdash s_t^x}\ x \in \mathscr{V}_\sigma, t \in \Lambda_\sigma$$

Fig. 1 Natural deduction calculus

- (Description) $\forall P.(\exists! x.Px) \to P(\mathsf{d}P)$
- (Extensionality) $\forall XY.X \subseteq Y \to Y \subseteq X \to X = Y$
- ($\in$-Induction) $\forall P.(\forall X.(\forall x.x \in X \to Px) \to PX) \to \forall X.PX$
- (Empty) $\neg\exists x.x \in \emptyset$
- (Union) $\forall Xx.x \in \bigcup X \equiv \exists Y.x \in Y \wedge Y \in X$
- (Power) $\forall XY.Y \in \wp X \equiv Y \subseteq X$
- (Separation) $\forall XPx.x \in \{z \in X | Pz\} \equiv x \in X \wedge Px$
- (Replacement) $\forall XFy.y \in \{Fz | z \in X\} \equiv \exists x.x \in X \wedge y = Fx$

We refer to the theory given by Γ_a as $\mathtt{IZF}_\omega^{-\infty}$. The subscript indicates the use of simple type theory (a form of higher-order logic). The superscript indicates that the axiom of infinity is omitted. We say a formula s is a *theorem of* $\mathtt{IZF}_\omega^{-\infty}$ if $\Gamma_a \vdash s$ holds.

From now on, we will only be concerned with theorems of $\mathtt{IZF}_\omega^{-\infty}$. We will state them as formulas, but the intended meta-theorem is that the given formula is a theorem of $\mathtt{IZF}_\omega^{-\infty}$. We will describe the interesting proofs informally, but all the proofs have been formalized in Coq in a way that corresponds to proofs in $\mathtt{IZF}_\omega^{-\infty}$.

The following lemma gives a few theorems which the reader may easily verify.

Lemma 1 *We have* $\forall X.\emptyset \subseteq X$, $\forall X.\emptyset \in \wp X$, $\forall X.X \in \wp X$, $\forall F.\{Fx | x \in \emptyset\} = \emptyset$ *and* $\forall XFG.(\forall x.x \in X \to Fx = Gx) \to \{Fx | x \in X\} = \{Gx | x \in X\}$.

3 Basic Definitions and Results

Now that we have fixed the set theory in question, we make a few basic definitions and indicate a few basic theorems which will be needed in the rest of the paper. In particular, we will define unordered pairs, singletons, binary unions, set difference and unions of families of sets. Also, we give some natural numbers as finite ordinals.

Zermelo included unordered pairs among the axioms of his original set theory [22]. However, once one adds Fraenkel's Replacement Axiom, unordered pairs can be defined. Zermelo points this out in [23]. Suppes gives the easy proof in [20] and Paulson formalized the proof in Isabelle/ZF [16]. The proof in classical ZF constructs the ordered pair $\{y, z\}$ by applying replacement with the two element set $\wp(\wp\emptyset)$ and a function mapping $\emptyset$ to y and $\wp\emptyset$ to z. The proof is not quite as easy in $\mathtt{IZF}_\omega^{-\infty}$, but is still within reach. First, let $\mathbf{T}$ be $\{X \in \wp(\wp\emptyset) | \emptyset \in X \vee \emptyset \notin X\}$. It is easy to prove both $\emptyset$ and $\wp\emptyset$ are elements of this set. Consider the term

$$t_X := \lambda w.\forall p.(\emptyset \notin X \to py) \to (\emptyset \in X \to pz) \to pw$$

of type ιo. If $\emptyset \notin X$, then y is the unique w such that $t_X w$. If $\emptyset \in X$, then z is the unique w such that $t_X w$. Let F be the term $\lambda X.\mathsf{d}t_X$ of type $\iota\iota$. By the description axiom, $\emptyset \notin X \to FX = y$ and $\emptyset \in X \to FX = z$. Consequently, $\{FX | X \in \mathbf{T}\}$ is a set that contains precisely y and z, as desired. Putting this together (and β-reducing), we define $\mathbf{U}$ to be the term $\lambda yz.\{\mathsf{d}(\lambda w.\forall p.(\emptyset \notin X \to py) \to (\emptyset \in X \to pz) \to pw) | X \in \mathbf{T}\}$ of type $\iota\iota$. We write $\{s, t\}$ for the term $\mathbf{U}st$. Formalizing the argument above, one can prove $\forall xyz.x \in \{y, z\} \equiv x = y \vee x = z$.

Once one has unordered pairs, singletons $\{s\}$ can be taken to mean $\{s, s\}$. Unordered pairs also allow us to define binary unions. We take $s \cup t$ to mean $\bigcup\{s, t\}$, and assume $\cup$ binds more tightly than $\in$. Set difference is definable from separation. Let $\mathbf{M}$ be $\lambda XY.\{x \in X | x \notin Y\}$. We write $s \setminus t$ for $\mathbf{M}st$ and assume $\setminus$ binds as tightly as $\cup$.

We can also describe a union of a family of sets. Let $\mathbf{F}$ be $\lambda XF.\bigcup\{Fx|x \in X\}$ of type $\iota(\iota\iota)\iota$. We write $\bigcup_{x\in s} t$ for terms of the form $\mathbf{F}s(\lambda x.t)$ and treat this notation as a binder that binds x and whose scope is as far to the right as possible. Let $y \in \mathscr{V}_\iota$. It is easy to prove the theorem $\forall XFy.y \in (\bigcup_{x\in X} Fx) \equiv \exists x.x \in X \wedge y \in Fx$.

We now describe a few finite ordinals. We take 0 to be $\emptyset$, as usual. The ordinal successor of a set X is taken to be $X \cup \{X\}$. To this end, let s^+ be notation for the term $s \cup \{s\}$. We assume the postfix $^+$ notation binds more tightly than all other notation. We take 1 to be 0^+ and take 2 to be 1^+. One can easily prove $1 = \{0\}$ and $2 = \{0, 1\}$, as expected.

We next describe definitions by $\in$-recursion. Definitions by $\in$-recursion are justified by the axiom of $\in$-induction. Suppose Φ is of type $\iota(\iota\iota)\iota$. Let $\mathbf{C}_\Phi$ be the formula $\forall XFG.(\forall x.x \in X \to Fx = Gx) \to \Phi XF = \Phi XG$. If $\mathbf{C}_\Phi$, then the value ΦXF depends only on X and the values Fx for $x \in X$. Under this condition, Φ can be used to define a (meta-level) function $\mathbf{R}\Phi$ such that $\forall X.\mathbf{R}\Phi X = \Phi X(\lambda x.\mathbf{R}\Phi x)$. Since we are working in a higher-order logic, we can define such an operator $\mathbf{R}$ of type $(\iota(\iota\iota)\iota)\iota\iota$ without too much trouble. First let $\mathbf{G}$ of type $(\iota(\iota\iota)\iota)\iota\iota o$ be

$$\lambda\Phi XY.\forall R.(\forall XF.(\forall x.x \in X \to Rx(Fx)) \to RX(\Phi XF)) \to RXY.$$

The term $\mathbf{G}\Phi$ corresponds to the least relation R such that if $\forall x.x \in X \to Rx(Fx)$, then $RX(\Phi XF)$.[2] We will prove that $\mathbf{G}\Phi$ is the graph of the function $\mathbf{R}\Phi$ we want to define. This justifies defining $\mathbf{R}$ to be the term $\lambda\Phi X.\mathsf{d}(\mathbf{G}\Phi X)$.

Lemma 2 *We have the following.*

1. $\forall\Phi XF.(\forall x.x \in X \to \mathbf{G}\Phi x(Fx)) \to \mathbf{G}\Phi X(\Phi XF)$
2. $\forall\Phi R.(\forall XF.(\forall x.x \in X \to \mathbf{G}\Phi x(Fx) \wedge Rx(Fx)) \to RX(\Phi XF)) \to$
   $\forall XY.\mathbf{G}\Phi XY \to RXY$
3. $\forall\Phi XY.\mathbf{G}\Phi XY \to \exists F.(\forall x.x \in X \to \mathbf{G}\Phi x(Fx)) \wedge Y = \Phi XF$
4. $\forall\Phi.\mathbf{C}_\Phi \to \forall XYZ.\mathbf{G}\Phi XY \to \mathbf{G}\Phi XZ \to Y = Z$
5. $\forall\Phi.\mathbf{C}_\Phi \to \forall X.\mathbf{G}\Phi X(\mathbf{R}\Phi X)$
6. $\forall\Phi.\mathbf{C}_\Phi \to \forall X.\mathbf{G}\Phi X(\Phi X(\mathbf{R}\Phi))$

Proof The first part follows easily from the definition of $\mathbf{G}$. Part 2 is an induction principle which is proven using the relation $\lambda XY.\mathbf{G}\Phi XY \wedge RXY$ with the definition of $\mathbf{G}$ and using Part 1. Part 3 is an inversion principle which follows from Part 2 using the relation $\lambda XY.\exists F.(\forall x.x \in X \to \mathbf{G}\Phi x(Fx)) \wedge Y = \Phi XF$. Part 4 is proven by $\in$-induction using Part 3. Part 5 is proven by $\in$-induction using the description axiom and Parts 1 and 4. Part 6 follows easily from Parts 1 and 5. □

Theorem 1 $\forall\Phi.\mathbf{C}_\Phi \to \forall X.\mathbf{R}\Phi X = \Phi X(\mathbf{R}\Phi)$.

Proof This follows from Parts 4, 5 and 6 of Lemma 2. □

One could alternatively define $\mathbf{G}\Phi$ via the Knaster-Tarski Fixed Point Theorem with the monotone operator $\lambda RXY.\exists F.(\forall x.x \in X \to Rx(Fx)) \wedge Y = \Phi XF$.

[2]The formalization of $\in$-recursion could be simplified in Coq by defining $\mathbf{G}$ as an inductive predicate since Coq automatically generates and proves induction principles. We use the definition here to remain within simple type theory.

4 Specification of Pairs and Functions

We are now in a position to precisely state what we would like an implementation of pairs and functions to satisfy. For $s, t \in \Lambda_\iota$, we need a term (s, t) of type ι. This can be provided by a term $\mathbf{P}$ of type $\iota\iota$ which constructs pairs (s, t) as $\mathbf{P}st$. Similarly, given a term $s \in \Lambda_\iota$ (corresponding to a set) and a term $t \in \Lambda_{\iota\iota}$ (corresponding to a function from sets to sets), we would like to have a term $\mathbf{L}st$ of type ι that encodes the function t when restricted to the set s. This can be given by a term $\mathbf{L}$ of type $\iota(\iota\iota)\iota$. We use the λ-binder to have the binder notation $\lambda x \in s.t$ for terms of the form $\mathbf{L}s(\lambda x.t)$. One can distinguish the set theory level λ from the type theory level λ by the presence or absence of $\in$ after the bound variable. The basic correctness property for pairing is $\forall xywz.(x, y) = (w, z) \equiv x = w \wedge y = z$. Similarly, the basic correctness property for $\mathbf{L}$ is $\forall XFG.(\forall x.x \in X \to Fx = Gx) \equiv (\lambda x \in X.Fx) = \lambda x \in X.Gx$. We must choose $\mathbf{P}$ and $\mathbf{L}$ so that these formulas will be theorems.

Note that we will now have two kinds of functions: functions at the level of the type theory are of type $\sigma\tau$ and functions at the level of the set theory are of type ι. Because of this distinction in the types, no confusion should arise. We will exclusively use F and G to range over variables of type $\iota\iota$ and f to range over variables of type ι. Note that the operator $\mathbf{L}$ takes a set X and a type theory level function F and returns the set theory level function $\lambda x \in X.Fx$.

In principle this is enough to say we have an encoding of pairs and functions. However, typically we also want to consider *sets* of pairs and functions. We specify this for the dependent case. We want terms $\mathbf{Q}^\Sigma$ and $\mathbf{Q}^\Pi$ of type $\iota(\iota\iota)\iota$. We use binder notation $\Sigma x \in s.t$ for terms of the form $\mathbf{Q}^\Sigma s(\lambda x.t)$ and use binder notation $\Pi x \in s.t$ for terms of the form $\mathbf{Q}^\Pi s(\lambda x.t)$. Intuitively, $\Sigma x \in s.t$ should be the set of pairs (x, y) where $x \in s$ and $y \in t$ (and t may depend on x). Likewise, $\Pi x \in s.t$ should be the set of functions f taking each $x \in s$ to an element of t (and where the intended domain of f is the set s). We require $\forall XYz.z \in (\Sigma x \in X.Yx) \equiv \exists x.x \in X \wedge \exists y.y \in Yx \wedge z = (x, y)$ and $\forall XYf.f \in (\Pi x \in X.Yx) \equiv \exists F.(\forall x.x \in X \to Fx \in Yx) \wedge f = \lambda x \in X.Fx$ to be theorems. When $x \notin \mathscr{F}t$, we write $s \times t$ for $\Sigma x \in s.t$ and we write t^s for $\Pi x \in s.t$.

A practical implementation of functions should include an operator for applying a function to an argument. This will be a term $\mathbf{A}$ of type $\iota\iota\iota$. A term $\mathbf{A}st$ corresponds to applying the object-level function to argument t. (For now, let us assume s corresponds to an object-level function and t corresponds to a member of its intended domain.) As usual, we would like to have an infix notation for $\mathbf{A}st$. Since s has type ι (and hence does not have a function type), there is no ambiguity in writing st for $\mathbf{A}st$. We have two basic correctness criteria for application. The first corresponds to β-reduction: $\forall XFx.x \in X \to (\lambda x \in X.Fx)x = Fx$. The second has the form of a common typing rule in dependent type theory: $\forall XYfx.f \in (\Pi x \in X.Yx) \to x \in X \to fx \in Yx$.

We now extend the specification to include extra properties of pairs and functions. Since the intention is that pairs (s, t) are actually functions with domain 2, we require $\forall F.(\lambda x \in 2.Fx) = (F0, F1)$. In addition, we will fix the behavior of application when used outside the domain of an object-level function by using 0 as a default value and requiring $\forall XFx.x \notin X \to (\lambda x \in X.Fx)x = 0$.

Finally, we will also require that the set $\wp 1$ satisfies certain closure properties. These properties were, in fact, the original motiviation for considering alternative representations for pairs and functions. A common way to give a set theoretic (proof irrelevant) semantics for a type theory with an impredicative universe `Prop` of propositions is to interpret `Prop`

$$\mathbf{P} \in \Lambda_{\iota\iota\iota} \qquad \mathbf{L} \in \Lambda_{\iota(\iota\iota)\iota} \qquad \mathbf{A} \in \Lambda_{\iota\iota\iota} \qquad \mathbf{Q}^{\Sigma} \in \Lambda_{\iota(\iota\iota)\iota} \qquad \mathbf{Q}^{\Pi} \in \Lambda_{\iota(\iota\iota)\iota} \qquad (s,t) := \mathbf{P}st$$

$$\lambda x \in s.t := \mathbf{L}s(\lambda x.t) \qquad st := \mathbf{A}st \qquad \Sigma x \in s.t := \mathbf{Q}^{\Sigma}s(\lambda x.t) \qquad s \times t := \Sigma x \in s.t \ (\text{if } x \notin \mathscr{F}t)$$

$$\Pi x \in s.t := \mathbf{Q}^{\Pi}s(\lambda x.t) \qquad t^s := \Pi x \in s.t \ (\text{if } x \notin \mathscr{F}t) \qquad \forall xywz.(x,y) = (w,z) \equiv x = w \wedge y = z$$

$$\forall XFG.(\forall x.x \in X \to Fx = Gx) \equiv (\lambda x \in X.Fx) = \lambda x \in X.Gx$$

$$\forall XYz.z \in (\Sigma x \in X.Yx) \equiv \exists x.x \in X \wedge \exists y.y \in Yx \wedge z = (x,y)$$

$$\forall XYf.f \in (\Pi x \in X.Yx) \equiv \exists F.(\forall x.x \in X \to Fx \in Yx) \wedge f = \lambda x \in X.Fx$$

$$\forall XFx.x \in X \to (\lambda x \in X.Fx)x = Fx \qquad \forall XYfx.f \in (\Pi x \in X.Yx) \to x \in X \to fx \in Yx$$

$$\forall F.(\lambda x \in 2.Fx) = (F0, F1) \qquad \forall XFx.x \notin X \to (\lambda x \in X.Fx)x = 0$$

$$\forall XY.(\forall x.x \in X \to Yx \in \wp 1) \to (\Pi x \in X.Yx) \in \wp 1$$

$$\forall X.X \in \wp 1 \to \forall Y.(\forall x.x \in X \to Yx \in \wp 1) \to (\Sigma x \in X.Yx) \in \wp 1$$

Fig. 2 Specification of pairs and functions

as $\wp 1$.[3] Since $\wp 1$ is not closed under function spaces when representing functions as graphs, Aczel [1] introduced an alternative representation so that $\wp 1$ is closed under function spaces. We will require $\wp 1$ to be closed under function spaces (where the codomain is in $\wp 1$) and closed under sets of pairs (where the sets containing both components are in $\wp 1$).

All the requirements of the specification are summarized in Fig. 2. The last two formulas are the formal versions of closure requirements for $\wp 1$.

If we assume the properties of the specification, then a number of theorems are provable. For example, $\forall X.X \times X = X^2$, $\forall xy.(x, y)0 = x$ and $\forall xy.(x, y)1 = y$ are provable from the specification.

5 Pairs as Disjoint Unions

We will define the pair (X, Y) so that it equals $\{(0, x)|x \in X\} \cup \{(1, y)|y \in Y\}$. That is, we will define pairing via disjoint union. Of course, $\{(0, x)|x \in X\} \cup \{(1, y)|y \in Y\}$ already makes use of pairing. To avoid circularity, we use $\in$-recursion to define a function $\mathbf{I}_1$. We will later prove (after defining pairing) that $\mathbf{I}_1 y = (1, y)$. From $\mathbf{I}_1$ we can easily define a function $\mathbf{I}_0$ which will later have the property $\mathbf{I}_0 x = (0, x)$. Once we have $\mathbf{I}_1$ and $\mathbf{I}_0$, we can define pairing as $\{\mathbf{I}_0 x|x \in X\} \cup \{\mathbf{I}_1 y|y \in Y\}$.

We define $\mathbf{I}_1$ by $\in$-recursion as on operator that recursively adjoins 0. Let $\mathbf{I}_1$ be the term $\mathbf{R}(\lambda XF.\{0\} \cup \{Fx|x \in X\})$ of type $\iota\iota$.

Lemma 3 $\forall X.\mathbf{I}_1 X = \{0\} \cup \{\mathbf{I}_1 x|x \in X\}$

Proof This follows from Theorem 1 and Lemma 1. □

Let $\mathbf{I}_0$ be the term $\lambda X.\{\mathbf{I}_1 x|x \in X\}$ of type $\iota\iota$.

We will need to know $\mathbf{I}_0$ and $\mathbf{I}_1$ are injective. To this end, we define a one-sided inverse $\mathbf{I}^-$ by $\in$-recursion. The function $\mathbf{I}^-$ will be a one-sided inverse to both $\mathbf{I}_0$ and $\mathbf{I}_1$ simultaneously. Let $\mathbf{I}^-$ be the term $\mathbf{R}(\lambda XF.\{Fx|x \in X \setminus \{0\}\})$.

[3] Note that under classical assumptions, $\wp 1$ is simply 2, in which case 2 has two "propositions" – 0 (false) and 1 (true).

Lemma 4 $\forall X.\mathbf{I}^{-}X = \{\mathbf{I}^{-}x | x \in X \setminus \{0\}\}$

Proof This follows from Theorem 1 and Lemma 1. □

Lemma 5 *We have* $\forall X.\mathbf{I}^{-}(\mathbf{I}_0 X) = X$ *and* $\forall X.\mathbf{I}^{-}(\mathbf{I}_1 X) = X$.

Proof The proof of $\forall X.\mathbf{I}^{-}(\mathbf{I}_1 X) = X$ is by $\in$-induction using Lemmas 3 and 4. The proof of $\forall X.\mathbf{I}^{-}(\mathbf{I}_0 X) = X$ uses $\forall X.\mathbf{I}^{-}(\mathbf{I}_1 X) = X$, the definition of $\mathbf{I}_0$ and Lemma 4. □

From Lemma 5 we can conclude that $\mathbf{I}_0$ and $\mathbf{I}_1$ are injective functions, as desired. We also want to prove that $\mathbf{I}_0$ and $\mathbf{I}_1$ have disjoint images.

Lemma 6 $\forall XY.\mathbf{I}_0 X \neq \mathbf{I}_1 Y$

Proof It is easy to prove $0 \in \mathbf{I}_1 Y$ and $0 \notin \mathbf{I}_0 X$. □

Before moving on to pairs, we establish the following simple result.

Lemma 7 $\mathbf{I}_0 0 = 0$

Proof This follows immediately from Lemma 1 and the definition of $\mathbf{I}_0$. □

We now define pairing. Let $\mathbf{P}$ be the term $\lambda XY.\{\mathbf{I}_0 x | x \in X\} \cup \{\mathbf{I}_1 y | y \in Y\}$ of type $\iota\iota\iota$. We write (s, t) as notation for the term $\mathbf{P}st$. A number of results follow easily from the lemma above and the definition of $\mathbf{P}$.

Lemma 8 $(0, 0) = 0$, $\forall x.\mathbf{I}_0 x = (0, x)$ *and* $\forall x.\mathbf{I}_1 x = (1, x)$.

Due to the equations in Lemma 8 we no longer need to consider the functions $\mathbf{I}_0$ and $\mathbf{I}_1$. (This is why $\mathbf{I}_0$ and $\mathbf{I}_1$ were not included in the specification in Section 4.) In fact, we can characterize the set encoding the pair as originally intended.

Lemma 9 $\forall XYz.z \in (X, Y) \equiv (\exists x.x \in X \wedge z = (0, x)) \vee (\exists y.y \in Y \wedge z = (1, y))$.

Using Lemma 9 it is easy to prove the following.

Lemma 10 *We have* $\forall xy.(0, x) = (0, y) \rightarrow x = y$, $\forall xy.(1, x) = (1, y) \rightarrow x = y$ *and* $\forall xy.(0, x) \neq (1, y)$.

Using Lemmas 9 and 10 we obtain Lemma 11.

Lemma 11 *We have* $\forall XYx.(0, x) \in (X, Y) \rightarrow x \in X$, $\forall XYy.(1, y) \in (X, Y) \rightarrow y \in Y$, $\forall xywz.(x, y) \subseteq (w, z) \rightarrow x \subseteq w$ *and* $\forall xywz.(x, y) \subseteq (w, z) \rightarrow y \subseteq z$.

Proof We prove the first statement. The other proofs are similar. Assume $(0, x) \in (X, Y)$. By Lemma 9 and the last part of Lemma 10 there is some $x' \in X$ such that $(0, x) = (0, x')$. By the first part of Lemma 10 $x = x'$ and so $x \in X$. □

From Lemma 11 we can easily prove the following theorem.

Theorem 2 $\forall xywz.(x, y) = (w, z) \equiv x = w \wedge y = z.$

Note that Theorem 2 corresponds to the first required property in Fig. 2.

6 Aczel Representation of Functions

We now turn to the representation of functions. Given X of type ι and a function F of type $\iota\iota$, we will define a set $\lambda x \in X.Fx$ of type ι which will represent the corresponding set theory level function. As a set, $\lambda x \in X.Fx$ will contain precisely the pairs (x, y) where $y \in Fx$. We define $\mathbf{L}$ to be the term $\lambda XF.\bigcup_{x \in X}\{(x, y)|y \in Fx\}$ of type $\iota(\iota\iota)\iota$. We write $\lambda x \in s.t$ as notation for the term $\mathbf{L}s(\lambda x.t)$. The following lemma is clear from the definition.

Lemma 12 $\forall XFz.z \in (\lambda x \in X.Fx) \equiv \exists x.x \in X \wedge \exists y.y \in Fx \wedge z = (x, y)$

Using Lemma 12 and Theorem 2 we have the following.

Lemma 13 $\forall XFxy.(x, y) \in (\lambda x \in X.Fx) \equiv x \in X \wedge y \in Fx$

We next define an application operator $\mathbf{A}$. Given an object level function f and a potential argument x, $\mathbf{A}fx$ should be the set of all y such that $(x, y) \in f$. Let $\mathbf{A}$ be the term $\lambda fx.\{\mathsf{d}(\lambda y.z = (x, y))|z \in \{z \in f|\exists y.z = (x, y)\}\}$ of type $\iota\iota$. We write st as notation for $\mathbf{A}st$ when s and t are terms of type ι.

Lemma 14 $\forall fxy.y \in fx \equiv (x, y) \in f.$

Proof Note that fx is notation for $\mathbf{A}fx$. Let f and x be given. By the axiom of description and Theorem 2, we know $\forall y.(x, y) = (x, \mathsf{d}(\lambda w.(x, y) = (x, w)))$ and so $\forall y.\mathsf{d}(\lambda w.(x, y) = (x, w)) = y$. Given this, we can prove $\forall y.y \in fx \equiv (x, y) \in f$ using the axioms of separation and replacement. □

We now have the infrastructure to prove more properties from Fig. 2.

Theorem 3 *We have* $\forall XFx.x \in X \rightarrow (\lambda x \in X.Fx)x = Fx$. *Furthermore, we have* $\forall XFx.x \notin X \rightarrow (\lambda x \in X.Fx)x = 0$.

Proof By Lemmas 13 and 14, $y \in (\lambda x \in X.Fx)x$ if and only if $(x, y) \in (\lambda x \in X.Fx)$ if and only if $x \in X \wedge y \in Fx$. This suffices to prove both results. □

Theorem 4 $\forall XFG.(\forall x.x \in X \rightarrow Fx = Gx) \equiv (\lambda x \in X.Fx) = \lambda x \in X.Gx.$

Proof Lemma 12 implies one direction. Theorem 3 implies the other. □

Theorem 5 $\forall F.(\lambda z \in 2.Fz) = (F0, F1).$

Proof Assume $u \in (\lambda z \in 2.Fz)$. By Lemma 12 there exist $z \in 2$ and $y \in Fz$ such that $u = (z, y)$. Either $z = 0$ or $z = 1$. In either case $u \in (F0, F1)$ by Lemma 9. Assume $u \in (F0, F1)$. By Lemma 9, either $u = (0, x)$ for some $x \in F0$ or $u = (1, y)$ for some $y \in F1$. In either case $u \in (\lambda z \in 2.Fz)$ by Lemma 12. □

We also prove application to 0 and 1 operate as projections on pairs as expected.

Theorem 6 $\forall xy.(x, y)0 = x$, $\forall xy.(x, y)1 = y$ *and* $\forall xyi.i \notin 2 \to (x, y)i = 0$.

Proof The first two results follow from Lemmas 9, 11 and 14. For the last result, assume $z \in (x, y)i$. By Lemma 14 $(i, z) \in (x, y)$ and so $i \in 2$ by Lemma 9. □

7 Sums and Products

Defining dependent sums (sets of pairs) is trivial. Let $\mathbf{Q}^\Sigma$ be $\mathbf{L}$. We write $\Sigma x \in s.t$ as notation for $\Sigma s(\lambda x.t)$. Note that $\Sigma x \in s.t$ is the same term as $\lambda x \in s.t$. Lemma 12 can now be written as $\forall XFz.z \in (\Sigma x \in X.Fx) \equiv \exists x.x \in X \wedge \exists y.y \in Fx \wedge z = (x, y)$ justifying the property of $\mathbf{Q}^\Sigma$ specified in Fig. 2. Lemma 8 implies $\wp 1$ is closed under Σ: $\forall X.X \in \wp 1 \to \forall Y.(\forall x.x \in X \to Yx \in \wp 1) \to (\Sigma x \in X.Yx) \in \wp 1$.

Defining dependent products (sets of functions) is not quite as easy, but does not require new ideas. Let $\mathbf{Q}^\Pi$ be $\lambda XY.\{f \in \wp(\Sigma x \in X.\bigcup(Yx)) | \forall x.x \in X \to fx \in Yx\}$ in $\Lambda_{\iota(\iota)\iota}$. We write $\Pi x \in s.t$ for $\mathbf{Q}^\Pi s(\lambda x.t)$, or t^s when $x \notin \mathscr{F}t$. It is straightforward to verify the remaining properties in Fig. 2. We also have $\forall X.X \times X = X^2$, where $X \times X$ is $\Sigma x : X.X$, using Theorem 5.

A number of monotonocity results are provable, including the following:

$$\forall XY.X \subseteq Y \to (\forall ZW.(\forall x.x \in X \to Zx \subseteq Wx) \to (\Sigma x \in X.Zx) \subseteq \Sigma y \in Y.Wy$$

$$\forall XYAB.(\forall x.x \in X \to Ax \subseteq Bx) \to X \subseteq Y \to (\forall y.y \in Y \to y \in X \vee y \notin X)$$
$$\to (\forall y.y \in Y \to y \notin X \to 0 \in By) \to (\Pi x \in X.Ax) \subseteq \Pi y \in Y.By$$

A consequence of the monotonicity result for Π is that $A^m \subseteq A^n$ whenever $0 \in A$ and m and n are finite ordinals with $m \in n$.

8 Conclusion

We have shown how one can define pairs and functions so that pairs are functions from 2 and the equation $X \times X = X^{\{0,1\}}$ holds. We conjecture that these representations of pairs and functions are more convenient in the context of formalized mathematics than the usual convention of taking pairs to be Kuratowski pairs and functions to be their graphs. For example, if u is a pair, it is common to write u_0 and u_1 for the two components. With the representation here, this subscript notation can simply be formalized as application of the function u to 0 or 1.

Acknowledgments Most of this work was done while part of Professor Gert Smolka's Programming Systems Lab at Saarland University. Thanks to Professor Gert Smolka for his support and stimulating conversations. Jonas Kaiser formalized Kuratowski pairs and Aczel functions using Tarski-Grothendieck set theory axiomatized in Coq as part of his Master's Thesis [11]. The discussions we had during this period improved my understanding of the Aczel representation of functions.

References

1. Aczel, P.: On relating type theories and set theories. In: Altenkirch, T., Naraschewski, W., Reus, B. (eds.) TYPES, *Lecture notes in computer science*, vol. 1657, pp. 1–18. Springer (1998)

2. Agerholm, S., Gordon, M.: Experiments with ZF set theory in HOL and Isabelle. In: Proceedings of the 8th International Workshop on Higher Order Logic Theorem Proving and its Applications, LNCS, pp. 32–45. Springer (1995)
3. Bancerek, G.: The ordinal numbers. Formalized Mathematics **1**(1), 91–96 (1990). http://fm.mizar.org/1990-1/pdf1-1/ordinal1.pdf
4. Bancerek, G.: Sequences of ordinal numbers. Formalized Mathematics **1**(2), 281–290 (1990). http://fm.mizar.org/1990-1/pdf1-2/ordinal2.pdf
5. Bancerek, G.: Algebra of morphisms. Formalized Mathematics **6**(2), 303–310 (1997). http://fm.mizar.org/1997-6/pdf6-2/catalg_1.pdf
6. Bancerek, G., Hryniewiecki, K.: Segments of natural numbers and finite sequences. Formalized Mathematics **1**(1), 107–114 (1990). http://fm.mizar.org/1990-1/pdf1-1/finseq_1.pdf
7. Byliński, C.: Functions and their basic properties. Formalized Mathematics **1**(1), 55–65 (1990). http://fm.mizar.org/1990-1/pdf1-1/funct_1.pdf
8. Church, A.: A formulation of the simple theory of types. J. Symb. Log. **5**, 56–68 (1940)
9. DeMarco, M., Lipton, J.: Completeness and cut-elimination in the intuitionistic theory of types. J. Log. Comput. **15**(6), 821–854 (2005)
10. van Heijenoort, J.: From Frege to Gödel. A source book in mathematical logic 1879–1931. Harvard University Press, Cambridge (1967)
11. Kaiser, J.: Formal Construction of a Set Theory in Coq. Master's thesis, Universität des Saarlandes (2012)
12. The Coq development team: The Coq proof assistant reference manual. LogiCal Project (2012). http://coq.inria.fr. Version 8.4
13. Morse, A.P.: A theory of sets. Academic Press (1965)
14. Myhill, J.: Some properties of intuitionistic Zermelo-Fraenkel set theory. In: Mathias, A., Rogers, H. (eds.) 1971 Cambridge summer school in mathematical logic, *Lecture Notes in Mathematics*, vol. 337, pp. 206–231. Springer, Berlin (1973)
15. Obua, S.: Partizan games in Isabelle/HOLZF. In: Barkaoui, K., Cavalcanti, A., Cerone, A. (eds.) ICTAC, *Lecture Notes in Computer Science*, vol. 4281, pp. 272–286. Springer (2006)
16. Paulson, L.C.: Set theory for verification: I. from foundations to functions. J. Autom. Reason. **11**, 353–389 (1993)
17. Prawitz, D.: Natural deduction: a proof-theoretical study. Dover (2006)
18. Russell, B.: The principles of mathematics. Cambridge University Press (1903)
19. Sĉêdrov, A.: Intuitionistic set theory. In: Harrington, A.S.L.A., Morley, M.D., Simpson, S. (eds.) Harvey Friedman's research on the foundations of mathematics, *Studies in Logic and the Foundations of Mathematics*, vol. 117, pp. 257–284. Elsevier (1985)
20. Suppes, P.: Axiomatic set theory. Dover (1972)
21. Trybulec, A.: Tarski Grothendieck set theory. Formalized Mathematics **1**(1), 9–11 (1990). http://fm.mizar.org/1990-1/pdf1-1/tarski.pdf
22. Zermelo, E.: Untersuchungen über die Grundlagen der Mengenlehre I. Mathematische Annalen **65**, 261–281 (1908). English translation, "Investigations in the foundations of set theory" in [10], pages 199–215
23. Zermelo, E.: Über Grenzzahlen und Mengenbereiche. Fundamenta Mathematicae **16**, 29–47 (1930)