ISSN: 1401-5617

Subsets and Quotients in Martin-Löf's Type Theory

Jesper Carlström

Research Reports in Mathematics
Number 8, 2002

Department of Mathematics
Stockholm University

Electronic versions of this document are available at
http://www.matematik.su.se/reports/2002/8

Date of publication: September 6, 2002
2000 Mathematics Subject Classification: Primary 03B15, Secondary 03B20.

Postal address:
Department of Mathematics
Stockholm University
S-106 91 Stockholm
Sweden

Electronic addresses:
http://www.matematik.su.se
info@matematik.su.se

Subsets and Quotients in Martin-Löf's Type Theory

Jesper Carlström*

September 2, 2002

Abstract

Subsets as propositional functions have been treated in type theory for a long time. For the development of algebra using this idea, one needs also the notion of an equivalence relation on a subset. Such relations are introduced in this paper. In order to get a satisfactory theory of partial functions, we propose also some changes to subset theory.

1 Introduction

We know already several ways of treating subsets together with equivalence relations in type theory, but they do not treat subsets as propositional functions, as we will do here. Most of these ways are described and compared in [2]. The most basic one is what we could call the "Σ-approach": subsets are treated as Σ-types in the following way.

Consider the pair $A', =_{A'}$, where A' is a set and $=_{A'}$ an equivalence relation on that set. Given a propositional function $A'' : (A')\,\mathrm{prop}$, we may construct a new pair

$$\Sigma(A', A''), =_\Sigma$$

where $(c =_\Sigma d) \stackrel{\mathrm{def}}{=} (\pi_\ell(c) =_{A'} \pi_\ell(d))$, and this new pair can be thought of as a "subset" in the sense which is common in category theory: there is a function

$$\pi_\ell(x) : A' \ (x : \Sigma(A', A''))$$

which is injective with respect to the equivalence relations in the obvious sense.

Any coarser equivalence $\equiv_\Sigma$ on $\Sigma(A', A'')$ defines naturally a "quotient"

$$\Sigma(A', A''), \equiv_\Sigma$$

and thus there is no problem in treating subsets and quotient sets together in this framework.

In some cases, however, this approach is not satisfactory. Consider the following example. Let A be a set with ring operations. Let $U : (A)\,\mathrm{prop}$ be the predicate saying that something is a *unit* or *invertible*,

$$U(a) \stackrel{\mathrm{def}}{=} (\exists y : A)(ay = 1 \wedge ya = 1)\,.$$

*mailto:jesper@matematik.su.se

In algebra, we often refer to "the group of units" of a ring. In taking the Σ-approach in formalizing this, one uses $\Sigma(A, U)$ with operations

$$\begin{aligned} 1 &\stackrel{\text{def}}{=} (1, p_1) \\ uv &\stackrel{\text{def}}{=} (\pi_\ell(u)\pi_\ell(v), p_\times(u, p, v, q)) \\ \text{inv}(u) &\stackrel{\text{def}}{=} (\pi_\ell\pi_r(u), (\pi_\ell(u), (\pi_r\pi_r(u), \pi_\ell\pi_r(u)))) \end{aligned}$$

where $p_1 : U(1)$ and $p_\times(u, p, v, q) : U(uv)$ $(u, v : A,\ p : U(u),\ q : U(v))$.

Consider now a practical situation, like the computation of $(a + b)^{-1} + c$, where $a, b, c : A$. Inversion is not defined on A, but on $\Sigma(A, U)$, so we need to "lift" $a + b$ to $\Sigma(A, U)$. Of course, we need a proof $p : U(a + b)$ to do this. So we use the function

$$(x, p) : \Sigma(A, U)\ (x : A,\ p : U(x))$$

of two variables to get an element of $\Sigma(A, U)$ out of the element of A we had, together with the proof needed.

We may then, using inv, invert $(a + b, p)$, but then we get an element of $\Sigma(A, U)$ rather than of A, so $\text{inv}((a+b, p)) + c$ does not make sense. So we have to apply π_ℓ to $\text{inv}((a+b, p))$ to get an element of A. Hence, the type-theoretical formalization of $(a + b)^{-1} + c$ becomes

$$\pi_\ell(\text{inv}((a + b, p))) + c\,.$$

Thus, what we really use for inverting elements is the function

$$d_p^{-1} \stackrel{\text{def}}{=} \pi_\ell(\text{inv}((d, p)))$$

which after unfolding of definitions becomes

$$d_p^{-1} \stackrel{\text{def}}{=} \pi_\ell((\pi_\ell\pi_r((d, p)), (\pi_\ell((d, p)), (\pi_r\pi_r((d, p)), \pi_\ell\pi_r((d, p))))))$$

and after reduction nothing but

$$d_p^{-1} \stackrel{\text{def}}{=} \pi_\ell(p)\,.$$

So someone formalizing rings with inversion in type theory would not benefit from mentioning $\Sigma(A, U)$, nor the function inv. It is much more straightforward to define $d_p^{-1} \stackrel{\text{def}}{=} \pi_\ell(p)$ directly. This is not a matter of taste, as was shown above, we are really forced to use this function. We may of course introduce $\Sigma(A, U)$, but we cannot avoid using the function d_p^{-1} in the end, simply because in order to use the function

$$\pi_\ell(\text{inv}(*)) : A\ (* : \Sigma(A, U))\,,$$

we need to have $*$, which we get using a function (d, p) of two variables: $d : A$ and $p : U(d)$, and so we use $\pi_\ell(\text{inv}((d, p)))$, which is definitionally equal to d_p^{-1}.

In formalizing the Fundamental Theorem of Calculus in `Coq`, Cruz-Filipe [3] discovered that he had no use for his 'subsetoid' (Σ-type with equivalence relation), precisely because of the facts mentioned above, and he also discovered that it was more efficient not mentioning subsetoids at all (private communication).

Thus, there is a gap between abstract Σ-based foundation of algebra on the one hand, and concrete formalization on the other. If the formalizer chooses to

avoid Σ-types, which is sound in our example because there is nothing gained by having them, then the Σ-based foundation fails to say that the inversion is defined on a subset — indeed, it is not defined on a subset in the sense of a Σ-type. But there is no doubt that there is, in an intuitive sense, a subset involved in this example. The idea of subsets as propositional functions allows to say that the inversion d_p is defined on a subset. But one needs also the notion of an equivalence relation on a subset in order to treat algebraic structures. In the example considered, it means that we want to be able to express such things as the units forming a group. Instead of saying that the subset of units is a certain Σ-type, we say that this subset is nothing but the propositional function U itself. We get, in the formalism we are about to develop, that the given equivalence on A satisfies the criterion of being an equivalence on U; and likewise for the multiplication and the constant 1. Finally, the operation d_p^{-1} is an extensional function $U \to U$. If we phrase the group axioms with quantifiers restricted to U, they are satisfied. So indeed we are justified in saying that U is a group with the operations mentioned.

2 Related Work

Subsets as propositional functions were first proposed by Martin-Löf [5, p. 64]. Nordström *et al.* [6] took a different approach by introducing new types for subsets, but they also gave an interpretation in pure Martin-Löf type theory where a subset A was interpreted as a pair A', A'', with $A' : \text{set}$, $A'' : (A')\,\text{prop}$. This paper is inspired by their ideas, but we do not introduce new types, but stay in pure type theory using the interpretation directly. To do like that was also the idea used by Sambin and Valentini in their so called 'toolbox' for subsets [10], which is maybe the most important source of inspiration for the present paper. In fact, we would have used their treatment of subsets if it had not been the case that it turned out to be necessary to change one detail, with the consequence that we have to rework almost everything.

The problem appears on the last page [10, p. 243], where partial functions are introduced. A partial function from a subset to a set B is a function $f(a, p) : B$ (a an element of the underlying set of the subset, p a proof that a belongs to the subset). According to the definition used in the toolbox, one requires $\mathsf{Id}(B, f(a, p), f(a, q))$ true for all proofs p, q of the right type. But this requirement is too strong for many purposes. One example was given in the introduction: it would be good to have inverse functions of rings accepted as partial functions. But the problem arises already at a very basic level: we would like to establish an isomorphism between a subset of A' as a propositional function A'' on the one hand, and the type $\Sigma(A', A'')$ on the other. We have of course the left projection $\pi_\ell(x) : A'$ $(x : \Sigma(A', A''))$ in one direction, but in the other direction, the obvious candidate is

$$(x, p) : \Sigma(A', A'')\ (x : A',\ p : A''(x))\,,$$

and hence we want this one to be accepted as a partial function from the subset to the Σ-type, but $\mathsf{Id}(\Sigma(A', A''), (a, p), (a, q))$ is of course not in general true.

Thus, the notion of partial function used in the toolbox is too restrictive: we have to drop the requirement $\mathsf{Id}(B, f(a, p), f(a, q))$ true. But when this is done, we naturally get propositions of the form $P(f(a, p))$, which depend on p as well

as on a. When such propositions occur in the scope of quantifiers, we have to accept that we sometimes need to quantify also over proofs, for instance, we may need to say

$$(\exists x : A', p : A''(x))P(f(x,p))\,.$$

Propositions like this one were not considered in earlier works on subsets.

So, when accepting partial functions that depend on proofs, we need to rework many details of subset theory. This paper is such an attempt. It would be a hopeless task to mention the origin of every idea properly, instead, please note that most ideas are taken directly from [6, 10], sometimes with small but crucial modifications.

There is an increasing interest in algebraic structures with carrier types not being sets. For instance, it is natural to treat the type of propositions as a Heyting algebra, or frame, with the equivalence relation being logical equivalence. Also, in formal topology (see e.g. [8, 9]), the formal points do not form sets, but they may have algebraic structure. Therefore, we will as much as possible try to avoid the assumption that we have sets. As a consequence, we have to use types rather than propositions in several definitions: for instance, instead of saying that a function is injective if a certain proposition is true, we have to say that it is injective if a certain type is inhabited.

3 Subsets

We adopt the notation A for the pair A', A'', as was done in [6], with the only difference that we do not require A' : set, but that A' is a type and $A'' : (A')\,\mathrm{prop}$. The pair is written "$\{A' \mid A''\}$" or "$\{x : A' \mid A''(x)\}$". Here A' is said to be the 'underlying type' and A'' is said to be a 'class' of objects of this type. We say that a class is a 'subset' when the underlying type is a set.

We need also express that $a : A'$ and $p : A''(a)$ build an 'element of A', which we do by saying "$a_p \,\varepsilon\, A$". The subscript notation here is *not* some kind of indexing, we could as well have written "a, p", but we prefer the subscript notation because it reminds of the fact that p is often considered not to be important in informal mathematics. When it is inessential to know what proof p we have, we may simply hide it. The subscript notation makes it possible to erase proof symbols without getting unreadable formulas.

$$\begin{aligned} A &\stackrel{\mathrm{def}}{=} \{A' \mid A''\} \stackrel{\mathrm{def}}{=} A', A'' \\ a_p &\stackrel{\mathrm{def}}{=} a, p \\ a_p \,\varepsilon\, A &\stackrel{\mathrm{def}}{=} a : A',\ p : A''(a)\,. \end{aligned}$$

Hiding p, we may write

$$a \,\varepsilon\, A \stackrel{\mathrm{def}}{=} a : A',\ A''(a)\,\mathrm{true}\ .$$

We also get the following notation for partial functions:

$$f(a_p) \stackrel{\mathrm{def}}{=} f(a,p)\,.$$

We almost automatically get quantification over subsets defined (not for arbitrary classes because we quantify over A', which therefore has to be a set), simply by spelling out what has been said about the notation "$a_p \,\varepsilon\, A$":

$$(\forall x_p \,\varepsilon\, A)P(x_p) \stackrel{\text{def}}{=} (\forall x : A',\, p : A''(x))P(x_p)$$
$$(\exists x_p \,\varepsilon\, A)P(x_p) \stackrel{\text{def}}{=} (\exists x : A',\, p : A''(x))P(x_p)\,.$$

Observe that the scope may depend on the proof p. When this is not the case, we simply get, by the type-theoretical definitions of $\to$ and $\wedge$,

$$(\forall x : A')(A''(x) \to P(x))$$
$$(\exists x : A')(A''(x) \wedge P(x))\,,$$

which are the standard definitions of restricted quantifiers, appearing also in [10, 6].

We choose to have the relation ε on the judgemental level, rather than on the propositional level as is proposed in the toolbox. However, just as we need the equality $\mathsf{Id}(A', a, b)$ as a propositional counterpart of the judgemental equality $a = b : A'$, it is often convenient to have counterparts also for the ε-relation and similar concepts. We therefore introduce such counterparts.

Judgement	Proposition
$a \,\varepsilon\, A$ $\stackrel{\text{def}}{=}$ $a : A',\ A''(a)\,\text{true}$	$a \,\varepsilon\, A$ $\stackrel{\text{def}}{=}$ $A''(a)$
$A \subseteq B$ $\stackrel{\text{def}}{=}$ $A' = B',\ x \,\varepsilon\, B\,\text{true}\,(x \,\varepsilon\, A)$	$A \subseteq B$ $\stackrel{\text{def}}{=}$ $(\forall x : A')(A''(x) \to B''(x))$ $\stackrel{\text{def}}{=}(\forall x : A')(x \,\varepsilon\, A \to x \,\varepsilon\, B)$ $\stackrel{\text{def}}{=}(\forall x \,\varepsilon\, A)(x \,\varepsilon\, B)$

The propositional version of $A \subseteq B$ makes sense only when $A' : \text{set}$ and $A' = B'$. It is safe to have the same notation for judgements and propositions, since it is always clear from the context whether something is a judgement or a proposition.

> *Remark.* Our propositional relation $a \,\varepsilon\, A$ differs from the one in the toolbox, where it is defined as $A''(a) \wedge \mathsf{Id}(A', a, a)$. As a consequence, our definition will not have the type-checking properties required in the toolbox, but this seems not to be a difficulty, since our judgemental form of ε has this property, and this seems indeed to be sufficient. See also the remark at the end of the next section.

4 Abuse of Language

It is common in mathematics to identify a set with the improper subset of it. In this spirit, it is natural when $A : \text{set}$ — and more generally, when A is a type — to use the notation "A" also for $\{x : A \mid \top\}$. It has to be checked manually that the context is clear enough for this abuse of language to be secure.

For instance, we may introduce a new symbol "U" by writing

"suppose $U \subseteq A$".

For this to make sense, we have to interpret it as $U \subseteq \{x : A \,|\, \top\}$ and assume that "U" denotes a pair $\{U' \,|\, U''\}$ with U' a type and $U'' : (U')\,\mathrm{prop}$. Then "$U \subseteq A$" means

$$U' = A \qquad \top \text{ true } (x \,\varepsilon\, U)\,.$$

The right judgement is trivially correct, so the expression "$U \subseteq A$" contributes precisely with the information that U is a pair $\{A \,|\, U''\}$ with U'' a class of objects of type A.

A second way of abusing language in a convenient way is when the underlying type is kept fixed but many classes are considered: say we consider $\{A \,|\, U\}$, $\{A \,|\, V\}$, and so on. Then it is natural to write simply "U" for $\{A \,|\, U\}$, "V" for $\{A \,|\, V\}$ etc. For instance, it is natural to write "$(\forall x \,\varepsilon\, U)P(x)$" (as proposed in the toolbox) instead of "$(\forall x \,\varepsilon \{x : A \,|\, U(x)\})P(x)$". In the toolbox, "$\{x : A \,|\, U(x)\}$" was just another notation for U, which is in agreement with our second abuse of language.

With A a type, this second abuse of language gives another interpretation of the expression "suppose $U \subseteq A$": clearly, "A" must still be interpreted as $\{x : A \,|\, \top\}$, because the second interpretation presupposes that A is a propositional function, which is not the case, because it was given that A is a type. But "U" can be interpreted as $\{A \,|\, U\}$, so that "$U \subseteq A$" is shorthand for "$\{A \,|\, U\} \subseteq \{x : A \,|\, \top\}$". Spelling out what this means, we get

$$U : (A)\,\mathrm{prop} \qquad A = A \qquad \top \text{ true } (x \,\varepsilon\, \{A \,|\, U\})\,.$$

Hence, the expression "$U \subseteq A$" contributes in this case precisely by saying that U is a propositional function on A, i.e., a subset of A, or a class of objects of type A.

To sum up, assuming A is a type, the expression "suppose $U \subseteq A$" can be interpreted in two different ways, either as the supposition that U is a propositional function on A (a subset or class) or as the supposition that U is a pair $\{A \,|\, U''\}$ with U'' a propositional function on A.

> *Remark.* When A is a type, the abuse of language gives that the *judgement* "$a \,\varepsilon\, A$" is $a : A$, $\top$ true, thus essentially $a : A$, while the *proposition* "$a \,\varepsilon\, A$" in this case is nothing but the proposition $\top$.

5 Equivalence Relations on Subsets

In introducing equivalence relations on subsets, we want to establish a correspondence with equivalence relations on Σ-types. We have already stated that the correspondence should be given by the pair

$$\pi_\ell(x) : A' \;\; (x : \Sigma(A', A''))$$
$$(x, p) : \Sigma(A', A'') \;\; (x : A', p : A''(x))$$

and so it remains only to see what notion of equivalence is induced on A by this pair of functions.

Suppose first that an equivalence $\equiv_\Sigma$ is defined on $\Sigma(A', A'')$ as in Sect. 1. We may then define

$$(a_p \equiv_A b_q) \stackrel{\text{def}}{=} ((a, p) \equiv_\Sigma (b, q))\,,$$

getting a relation $\equiv_A$ of *four* variables out of the relation $\equiv_\Sigma$ of *two* variables.

A motivation for having the proofs in subscript notation comes from the fact that we have a certain amount of "proof irrelevance":

$$x_p \equiv_A x_q \,\mathrm{true}\ (x_p, x_q \,\varepsilon\, A)\,,$$

which is proved using

$$(a_p \equiv_A a_q) \stackrel{\mathrm{def}}{=} ((a,p) \equiv_\Sigma (a,q)) \leftarrow ((a,p) =_\Sigma (a,q)) \stackrel{\mathrm{def}}{=} (a =_{A'} a)\,,$$

with $a =_{A'} a$ proved by reflexivity of $=_{A'}$.

This leads us to the following axioms for equivalence on a class:

$$x_p =_A y_q : \mathrm{prop}\ (x_p, y_q \,\varepsilon\, A)$$

such that

$$\begin{array}{rl}
(\mathsf{refl}) & x_p =_A x_q \,\mathrm{true}\ (x : A',\ p, q : x \,\varepsilon\, A) \\
(\mathsf{sym}) & x_p =_A y_q \,\mathrm{true}\ (x_p, y_q \,\varepsilon\, A,\ y_q =_A x_p \,\mathrm{true}) \\
(\mathsf{tr}) & x_p =_A z_r \,\mathrm{true}\ (x_p, y_p, z_r \,\varepsilon\, A,\ x_p =_A y_q \,\mathrm{true},\ y_q =_A z_r \,\mathrm{true})\,.
\end{array}$$

Note that, in the reflexivity axiom, we do not require the proofs to be the same on both sides. This is a result of the idea that $=_A$ be coarser than some totally defined equivalence relation, as in the introductory example. In fact, our reflexivity axiom is equivalent with this requirement in the case A' : set, because

$$\mathsf{Id}(A', x, y) \to (x_p =_A y_q) \,\mathrm{true}\ (x_p, y_q \,\varepsilon\, A)$$

follows from refl. This is not as straightforward to prove as one might think: first, let $P(z) \stackrel{\mathrm{def}}{=} (\forall r : z \,\varepsilon\, A)(x_p =_A z_r)$. Using $\lambda r.\, \mathsf{refl}_{=_A}(x, p, r) : P(x)$ and a proof of $\mathsf{Id}(A', x, y)$, we get a proof of $P(y)$. Applying it to q, we get a proof of $x_p =_A y_q$. Thus, $\lambda w.\, \mathsf{ap}(\mathsf{subst}(w, \lambda r.\, \mathsf{refl}_{=_A}(x, p, r)), q)$ is a proof of $\mathsf{Id}(A', x, y) \to (x_p =_A y_q)$.

The notion of equivalence proposed here seems to be a heavy one because of all proofs it may depend on. But a feature of type theory is that this kind of dependence is always optional. In many applications, one gets dependence only on a few proofs. In particular, ordinary equivalence relations are equivalence relations in our sense. We just proved above that the Id relation is the finest of all equivalence relations.

A pair $A \stackrel{\mathrm{def}}{=} \{A' \,|\, A''\}$ together with an equivalence $=_A$ will be denoted by "$A/{=_A}$". We use the same notation when A is a type and $=_A$ a total equivalence relation on it. According to what we have said about abuse of language in Sect. 4, "$A/{=_A}$" should mean $\{A \,|\, (x)\top\}/{=_A}$ in that case, but clearly it is safe and more natural to dispense with the propositional function $(x)\top$.

Of course, if $C \stackrel{\mathrm{def}}{=} A/{=_A}$, we allow ourselves to write "$x_p \,\varepsilon\, C$" when we actually mean $x_p \,\varepsilon\, A$, etc.

6 Partial Functions

A *property* defined on A is a propositional function $P : (x_p \,\varepsilon\, A)\,\mathrm{prop}$ of two variables, with

$$P(x_q)\,\mathrm{true}\ (x_p, x_q \,\varepsilon\, A,\ P(x_p)\,\mathrm{true})\,.$$

Being equal to a distinguished element is a property: Let $c_r \,\varepsilon\, A$ and $P(a_p) \stackrel{\mathrm{def}}{=} (a_p =_A c_r)$, then

$$\mathsf{tr}_{=_A}(x_q, x_p, c_r, \mathsf{refl}_{=_A}(x, q, p), s) : P(x_q)\ (x_p, x_q \,\varepsilon\, A,\ s : P(x_p))\,.$$

Properties are examples of *partial functions*. A partial function from A to B is a function $f : (x_p \,\varepsilon\, A)B'$ with $f(x_p) \,\varepsilon\, B\,\mathrm{true}\ (x_p \,\varepsilon\, A)$ (we often say only 'function' instead of 'partial function').

When B is a type, we may, by the abuse of language mentioned in Sect. 4, write "B" also for $\{x : B \mid \top\}$, and so a partial function from A to B is an element of $(x_p \,\varepsilon\, A)B$ with $\top\,\mathrm{true}\ (x_p \,\varepsilon\, A)$. Since the latter judgement is trivially correct, a partial function from A to B is in this case precisely an element of $(x_p \,\varepsilon\, A)B$.

When A is a type, we get, by the same abuse of language, that a partial function from A to B is a function $f : (A, \top)B'$ with $f(x_p) \,\varepsilon\, B\,\mathrm{true}\ (x_p \,\varepsilon\, A)$. It is essentially a function of type $(A)B'$ in the sense that it is extensionally equal to such a function, namely to $(x)f(x_{\mathsf{tt}}) : (A)B'$, because we have

$$\mathsf{Id}(B', f(x_p), f(x_{\mathsf{tt}}))\,\mathrm{true}\ (x_p \,\varepsilon\, A)\,.$$

However, on the intensional level, we have to distinguish between *functions* from A to B and *partial functions* from A to B when A is a type. We will not always make that distinction explicit, the correct interpretation will be evident from the context.

In the case with both A and B types, we get that a partial function from A to B is an element of the type $(A, \top)B$, thus essentially an element of $(A)B$, as we wish it to be.

An *extensional* function $A/{=_A} \to B/{=_B}$ is a function f from A to B such that

$$f(x_p)_\alpha =_B f(y_q)_\beta\,\mathrm{true}\ (x_p, y_q \,\varepsilon\, A,\ x_p =_A y_q\,\mathrm{true},\ \alpha : f(x_p) \,\varepsilon\, B,\ \beta : f(y_q) \,\varepsilon\, B)\,.$$

This definition is independent of the choice of $f'' : (x_p \,\varepsilon\, A)(f(x_p) \,\varepsilon\, B)$. However, given such an f'', it is sufficient to prove

$$f(x_p)_{f''(x_p)} =_B f(y_q)_{f''(y_q)}\,\mathrm{true}\ (x_p, y_q \,\varepsilon\, A,\ x_p =_A y_q\,\mathrm{true})$$

because of the form of the reflexivity axiom for $=_B$.

Being a partial function in the sense of the toolbox is equivalent with being, in our language, an extensional function $A/\mathsf{Id} \to B/\mathsf{Id}$, with $A', B : \mathrm{set}$. The proof of this equivalence goes very much like the proof of $\mathsf{Id}(A', x, y) \to (x_p =_A y_q)$ in the previous section.

By taking $\mathrm{prop}/{\Leftrightarrow}$ for $B/{=_B}$, we get a notion of *extensional propositional function*:

$$P(x_p) \Leftrightarrow P(y_q)\,\mathrm{true}\ (x_p, y_q \,\varepsilon\, A,\ x_p =_A y_q\,\mathrm{true})\,.$$

An extensional propositional function is clearly a property, and therefore the notion of property will not be important when equivalence relations are present, it rather belongs to pure subset theory.

7 Partial Graphs

A *partial graph* from a type A to a type B with equivalence relation $=_B$ is a relation $R(x, y)$: prop $(x : A, y : B)$ such that $y =_B y'$ true $(x : A, y, y' : B, R(x, y)$ true, $R(x, y')$ true). If R is such a relation and B : set, we may define the *domain* of R by $D(a) \stackrel{\text{def}}{=} (\exists y : B)R(a, y)$ and we get a partial function from D to B by

$$f(a_p) \stackrel{\text{def}}{=} \pi_\ell(p)\,.$$

If, on the other hand, f is a partial function from U to B, we get a partial graph by

$$R(a, b) \stackrel{\text{def}}{=} (\exists p : a \,\varepsilon\, U)(f(a_p) =_B b)\,.$$

The correspondence indicated is an equivalence when extensionality is assumed.

8 Quotients of Quotients

We put

$$(A/=_A)/\equiv_A \;\stackrel{\text{def}}{=}\; A/\equiv_A\,.$$

This is known as the 'Second Isomorphism Theorem' in the literature but in a setting like ours, where we do not use equivalence classes, it can be given as a definition.

It is natural to require that $\equiv_A$ be coarser than $=_A$, because this is what makes the projection $A/=_A \,\to\, A/\equiv_A$ extensional. Without this requirement, we can interpret the sign "/" as nothing but a change of equivalence relations.

9 Subsets of Subsets

A subclass of A is given by a property $P(x_p)$: prop $(x_p \,\varepsilon\, A)$. Thus, to know that an element a belongs to it, we need both a proof $p : a \,\varepsilon\, A$ to make $P(a_p)$ make sense, and then a proof $q : P(a_p)$. Forming a pair of these proofs, we get precisely a canonical proof of the proposition $(\exists p : a \,\varepsilon\, A)P(a_p)$ and hence we define

$$\{A \mid P\} \stackrel{\text{def}}{=} \{x : A' \mid (\exists p : x \,\varepsilon\, A)P(x_p)\}\,,$$

allowing also the notation $\{x_p \,\varepsilon\, A \mid P(x_p)\}$.

When $P(a_p)$ does not depend on p, we get, by the type-theoretical definition of $\wedge$, simply

$$\{x \,\varepsilon\, A \mid P(x)\} \stackrel{\text{def}}{=} \{x : A' \mid A''(x) \wedge P(x)\}\,,$$

as in [6].

When A is a type, "$\{x \,\varepsilon\, A \mid P(x)\}$" is, by the abuse of language, interpreted as $\{x \,\varepsilon \{x : A \mid \top\} \mid P(x)\}$, i.e. $\{x : A \mid \top \wedge P(x)\}$. So $\{x \,\varepsilon\, A \mid P(x)\}$ and $\{x : A \mid P(x)\}$ are extensionally equal, though the latter one is intensionally more natural.

10 Subsets of Quotients

With $=_{A'}$ a total equivalence relation on A', we let

$$\{A'/=_{A'} \mid P\} \stackrel{\text{def}}{=} \{A' \mid P\}/=_{A'} .$$

In the general case $A \stackrel{\text{def}}{=} \{A' \mid A''\}$ with $=_A$ not necessarily total, we need to restrict it to $\{A \mid P\}$, so we put

$$\{A/=_A \mid P\} \stackrel{\text{def}}{=} \{A \mid P\}/=_r ,$$

with $=_r$ the restriction as defined in Sect. 12.

It is natural to require that a subset of a quotient should respect the equivalence, i.e., that the propositional function be extensional. Surprisingly, it seems that this requirement is not as important as one might think. Therefore, we do not include it in the definition. Instead, it can be added when convenient.

11 Some Examples

There are many applications for which it is important that we do not require that the underlying type is a set:

1. The class
$$\{X : \text{prop} \mid X\}$$
of true propositions, or the class
$$\{X : \text{prop} \mid A \to X\}$$
of propositions that follow from a fixed proposition A.

2. Powerset and families of subsets. For A : set, let $\mathcal{P}(A) \stackrel{\text{def}}{=} ((A)\,\text{prop})/=_{\text{ext}}$, where
$$X =_{\text{ext}} Y \stackrel{\text{def}}{=} (\forall x : A)(x \,\varepsilon\, X \iff x \,\varepsilon\, Y) .$$
An $I/=_I$-indexed extensional family of subsets of a set A is an extensional function $I/=_I \to \mathcal{P}(A)$. This makes sense both when I is a type and when $I \stackrel{\text{def}}{=} \{I' \mid I''\}$. In the former case, this definition amounts to saying that a family of subsets is of type $(I, A)\,\text{prop}$, thus by definition a binary relation (satisfying also an extensionality condition). In the case with $I \stackrel{\text{def}}{=} \{I' \mid I''\}$, a family of subsets is of type $(i : I,\ p : i \,\varepsilon\, I,\ A)\,\text{prop}$, i.e., a ternary relation with the type of the second argument depending on the first argument.
With the notation "U_{i_p}" instead of "$U(i_p)$" the extensionality condition reads
$$U_{i_p} =_{\text{ext}} U_{j_q}\,\text{true}\ (i_p, j_q \,\varepsilon\, I,\ i_p =_I j_q\,\text{true}) .$$

3. Suppose $=_A$ is a total equivalence relation on a set A. For $a : A$, let $[a]$ be the corresponding equivalence class $(x)(x =_A a)$. The class
$$EC_{A/=_A} \stackrel{\text{def}}{=} \{X : \mathcal{P}(A) \mid (\exists x : A)(X =_{\text{ext}} [x])\}$$

of equivalence classes is isomorphic with $A/{=_A}$ (this notion of isomorphism will be made precise in Sect. 13): $(x)[x]$ is an extensional function $A/{=_A} \to EC_{A/=_A}$ with extensional inverse being $(X_p)\pi_\ell(p)$.

$EC_{A/=_A}$ consists of extensional subsets only. The class of extensional subsets with extensional equality is

$$\mathcal{P}(A/{=_A}) \stackrel{\text{def}}{=} \{X : \mathcal{P}(A) \mid (\forall x\, \varepsilon\, X,\, y\, \varepsilon [x])(y\, \varepsilon\, X)\}$$

and it may be formally checked that $EC_{A/=_A} \subseteq \mathcal{P}(A/{=_A})$.

4. 'Lifting', as in Domain Theory. With classical logic, it can be described as the addition of a single 'bottom' element $\bot$, but there are also well known (classically equivalent) definitions which work constructively, for instance the collection of subsets with at most one element [4]. We show that this construction can be carried out and naturally expressed in our language; so it is not, as is often thought, impredicative in itself, what is impredicative is to accept the resulting class as a set (see however [7] for a way of solving this using a universe).

   Let B : set with an equivalence relation $=_B$. The lifting of $B/{=_B}$ is

   $$(B/{=_B})_\bot \stackrel{\text{def}}{=} \{X\, \varepsilon\, \mathcal{P}(B/{=_B}) \mid (\forall x, y\, \varepsilon\, X)(x =_B y)\}\,.$$

   We have $EC_{B/=_B} \subseteq (B/{=_B})_\bot$ but they are not equal since the empty subset is an element of the lifting. On the other hand, any inhabited element of the lifting is an element of $EC_{B/=_B}$.

   Now, let A' : set with equivalence relation $=_A$. Let $A \stackrel{\text{def}}{=} \{A' \mid A''\}$ for some propositional function A''. For an extensional $f : A/{=_A} \to B/{=_B}$, let

   $$f_\bot(a) \stackrel{\text{def}}{=} \{y : B \mid (\exists p : a\, \varepsilon\, A)(y =_B f(a_p))\}\,.$$

   It is straightforward to check that $f_\bot$ is an extensional function $A'/{=_A} \to (B/{=_B})_\bot$ and that it extends f in the sense that if $a_p\, \varepsilon\, A$, then

   $$f_\bot(a) =_{\text{ext}} [f(a_p)]\,\text{true}\,.$$

5. In a formal topology (see e.g. [8, 9]), let $U \lhd\rhd V$ mean that U and V cover each other. Let

   $$\Omega(S, \lhd) \stackrel{\text{def}}{=} \mathcal{P}(S)/{\lhd\rhd}\,.$$

   The class of opens that cover $a : S$ is given by

   $$\{U : \Omega(S, \lhd) \mid a \lhd U\}$$

   which is definitionally equal to

   $$\{U : (S)\,\text{prop} \mid a \lhd U\}/{\lhd\rhd}\,.$$

12 Kernels and Images

For the definition of ker and im of $f : A \to B$, we need a distinguished

$$f'' : (x_p \,\varepsilon\, A)(f(x_p) \,\varepsilon\, B)$$

for them to depend on. To be precise, we would have to write "$\ker_{f,f''}$", but for readability, we do not spell out the functions.

Suppose that $=_B$ is defined on B (we do not need any equivalence defined on A).

The kernel is defined by

$$a_p \ker b_q \stackrel{\text{def}}{=} f(a_p)_{f''(a_p)} =_B f(b_q)_{f''(b_q)} .$$

Note that f'' appears in the subscript only. Therefore, if $=_B$ does not depend on proofs, ker will not depend on f'', and thus we need not specify it. We then get simply

$$a_p \ker b_q \stackrel{\text{def}}{=} f(a_p) =_B f(b_q)$$

and when f does not depend on the proof, simply

$$a \ker b \stackrel{\text{def}}{=} f(a) =_B f(b) .$$

That $f : A/{=_A} \to B/{=_B}$ is extensional is precisely that ker is coarser than $=_A$.

An application of the kernel is in the order of classes. We have defined

$$A \subseteq B \quad \stackrel{\text{def}}{=} \quad A' = B',\ x \,\varepsilon\, B \,\mathrm{true}\,(x \,\varepsilon\, A)$$

which says precisely that the "inclusion" $(x_p)x$ is a function $A \to B$. With a distinguished $i'' : (x_p \,\varepsilon\, A)(x \,\varepsilon\, B)$, we get a kernel

$$a_p \ker b_q \stackrel{\text{def}}{=} a_{i''(a_p)} =_B b_{i''(b_q)} .$$

When $=_B$ does not depend on proofs, this amounts to

$$a \ker b \stackrel{\text{def}}{=} a =_B b .$$

Thus, if $=_B$ is defined on B, we get an equivalence defined on A: the 'restriction to A'. For instance, the restriction of $=_A$ to $\{A \mid P\}$ is the kernel $\ker_{(x_p)x,\,\pi_\ell}$ of the inclusion $\{A \mid P\} \to A$.

When $=_B$ does not depend on proofs, the restriction will be the same as $=_B$.

> *Remark.* It is worth noting that $\lambda x.\lambda p.i''(x_p)$ is a proof of the *proposition* $A \subseteq B$. In the toolbox, $A'' \subseteq_{A'} B''$ is defined to be
>
> $$(\forall x : A')((A''(x) \wedge \mathsf{Id}(A', x, x)) \to (B''(x) \wedge \mathsf{Id}(A', x, x))) .$$
>
> A proof of this proposition is $\lambda x.\lambda q.(i''(x_{\pi_\ell(q)}), \pi_r(q))$.

The definition of image is straightforward, but it uses quantification over A' and thus needs A' : set to make sense:

$$\mathrm{im} \stackrel{\text{def}}{=} \{y_q \,\varepsilon\, B \mid (\exists x_p \,\varepsilon\, A)(f(x_p)_{f''(x_p)} =_B y_q)\} .$$

13 Injectivity, Surjectivity, and Bijectivity

It is easy to state what injectivity and surjectivity should mean using quantifiers, but since we want these concepts to make sense also for types that are not sets, we avoid quantifiers.

To begin with, we state injectivity for a (not necessarily extensional) partial function $f : A/{=_A} \to B/{=_B}$ by the judgement

$$x_p =_A y_q \,\mathrm{true}\ (x_p, y_p \,\varepsilon\, A,\ \alpha : f(x_p) \,\varepsilon\, B,\ \beta : f(y_q) \,\varepsilon\, B,\ f(x_p)_\alpha =_B f(y_q)_\beta \,\mathrm{true})\,.$$

When $A', B' : \mathrm{set}$ and $f'' : (x_p \,\varepsilon\, A)(f(x_p) \,\varepsilon\, B)$, this judgement is equivalent with the truth of the proposition

$$(\forall x_p, y_q \,\varepsilon\, A)(f(x_p)_{f''(x_p)} =_B f(y_q)_{f''(y_q)} \to x_p =_A y_q)$$

because of the form of the reflexivity axiom for $=_B$.

It is tempting to define surjectivity by $B \subseteq \mathrm{im}$ or something similar, but the definition of im uses quantifiers, so we avoid it. Instead, we define surjectivity as Bishop [1, p.15] did by "having (not necessarily extensional) right inverse".

If $=_B$ is a total equivalence relation on B, f is an extensional function $A/\mathsf{Id} \to B/{=_B}$, and g is a function $B \to A$ (not necessarily extensional), we state that g is right inverse of f (and f left inverse of g) by the judgement

$$f(g(y_q)_r)_s =_B y_q \,\mathrm{true}\ (y_q \,\varepsilon\, B,\ r : g(y_q) \,\varepsilon\, A,\ s : f(g(y_q)) \,\varepsilon\, B)\,.$$

This judgement is independent of the choices of $f'' : (x_p \,\varepsilon\, A)(f(x_p) \,\varepsilon\, B)$ and $g'' : (y_q \,\varepsilon\, B)(g(y_q) \,\varepsilon\, A)$. With such choices made, we get the equivalent — and more familiarly looking — judgement

$$f(g(y_q)_{g''(y_q)})_{f''(g(y_q)_{g''(y_q)})} =_B y_q \,\mathrm{true}\ (y_q \,\varepsilon\, B)\,.$$

The equivalence of these judgements relies on the extensionality of $f : A/\mathsf{Id} \to B/{=_B}$.

The surjectivity of f is expressed by the pair of judgements $g : B \to A$, 'g is right inverse of f', but with g hidden.

When f is extensional $A/{=_A} \to B/{=_B}$ and $A' : \mathrm{set}$, so that $\mathrm{im}_{f,f''}$ makes sense, f is surjective to im. The corresponding right inverse g is extensional precisely when f is injective.

If $f : A/{=_A} \to B/{=_B}$ and $g : B/{=_B} \to A/{=_A}$ are both extensional and each-others inverses on both sides, we have the notion of *bijective correspondence* or *isomorphism*. In that case, f is both injective and surjective. When we hide g, we get the judgement that f is bijective.

On the other hand, suppose f is both injective and surjective. Let $g : B \to A$ be the right inverse which exists by the surjectivity. We prove that f is bijective by proving extensionality of g and that g is also left inverse of f. The extensionality of g is obtained immediately from the injectivity of f, and

$$g(f(x_p)_q)_r =_A x_p$$

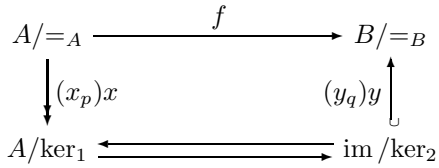
follows, also by injectivity of f, from

$$f(g(f(x_p)_q)_r)_s =_B f(x_p)_q\,,$$

which is easily proved using that g is right inverse of f.

14 The First Isomorphism Theorem

It is now an easy exercise to show that any extensional function f, with domain inside a set, can be factored into a projection onto a quotient, followed by an isomorphism, followed by an inclusion.



The relation $\ker_1$ is the kernel of (f, f''), while $\ker_2$ is the kernel of the inclusion $\mathrm{im} \to B$.

The proof is immediate when all definitions have been expanded. We give the details because they serve as instructive examples. We have

$$\begin{aligned}
\mathrm{im} &\stackrel{\mathrm{def}}{=} \{y_q \,\varepsilon\, B \mid (\exists x_p \,\varepsilon\, A) f(x_p)_{f''(x_p)} =_B y_q\} \\
&\stackrel{\mathrm{def}}{=} \{y : B' \mid (\exists q : y \,\varepsilon\, B,\, x_p \,\varepsilon\, A) f(x_p)_{f''(x_p)} =_B y_q\} \\
(a_p \,\ker_1\, b_q) &\stackrel{\mathrm{def}}{=} (f(a_p)_{f''(a_p)} =_B f(b_q)_{f''(b_q)}) \\
(a_p \,\ker_2\, b_q) &\stackrel{\mathrm{def}}{=} (a_{\pi_\ell(p)} =_B b_{\pi_\ell(q)})\,.
\end{aligned}$$

The projection is extensional because f is. The inclusion is extensional by definition of $\ker_2$. The asserted map $A/\ker_1 \to \mathrm{im}\,/\ker_2$ is nothing but f itself, because we do not use equivalence classes. The proof that f is indeed $A \to \mathrm{im}$ is

$$F(a_p) \stackrel{\mathrm{def}}{=} (f''(a_p), (a, (p, \mathsf{refl}_B(f(a_p), f''(a_p), f''(a_p)))))\,.$$

We prove that f is extensional and injective $A/\ker_1 \to \mathrm{im}\,/\ker_2$ by proving

$$a_p \,\ker_1\, b_q \iff f(a_p)_{F(a_p)} \,\ker_2\, f(b_q)_{F(b_q)}\,,$$

but this is obvious because by expanding the definitions, we get definitionally equal propositions on both sides.

According to the previous section, it now suffices to find a right inverse, which then automatically is extensional and an inverse to f. We claim that $g(a_q) \stackrel{\mathrm{def}}{=} \pi_\ell \pi_r(q)$ is such a right inverse. It is a function $\mathrm{im} \to A$ because

$$\pi_\ell \pi_r \pi_r(q) : g(y_q) \,\varepsilon\, A \;\; (y_q \,\varepsilon\, \mathrm{im})\,.$$

To prove that g is a right inverse is to prove the judgement

$$f(g(y_q)_{\pi_\ell \pi_r \pi_r(q)})_{F(g(y_q)_{\pi_\ell \pi_r \pi_r \pi_r(q)})} \,\ker_2\, y_q \,\mathrm{true}\;\; (y_q \,\varepsilon\, \mathrm{im})\,,$$

which, by definition of F and $\ker_2$, amounts to

$$f(g(y_q)_{\pi_\ell \pi_r \pi_r(q)})_{f''(g(y_q)_{\pi_\ell \pi_r \pi_r \pi_r(q)})} =_B y_{\pi_\ell(q)} \,\mathrm{true}\;\; (y_q \,\varepsilon\, \mathrm{im})\,.$$

So we assume $y_q \,\varepsilon\, \mathrm{im}$, i.e. $y : B', q : (\exists r : y \,\varepsilon\, B,\, x_p \,\varepsilon\, A)(f(x_p)_{f''(x_p)} =_B y_r)$, and prove $f(g(y_q)_{\pi_\ell \pi_r \pi_r(q)})_{f''(g(y_q)_{\pi_\ell \pi_r \pi_r \pi_r(q)})} =_B y_{\pi_\ell(q)}$. But that is easily done: $\pi_r \pi_r \pi_r(q)$ is a such a proof.

15 Conclusion

We have given tools for handling equivalence relations together with subsets as propositional functions. We hope that these tools will be useful in formalizing mathematics. Whether this is indeed the case has to be judged by experience.

16 Acknowledgements

I would like to thank Per Martin-Löf for many discussions and ideas on this subject, as well as for reading and commenting on an early version of the manuscript. Also Giovanni Curi gave me valuable comments.

References

[1] E. Bishop. *Foundations of Constructive Analysis*. McGraw-Hill, 1967.

[2] V. Capretta. *Abstraction and Computation*, Chap. 5: Setoids. PhD thesis, Katholieke Universiteit Nijmegen, 2002.

[3] L. Cruz-Filipe. Formalizing real calculus in Coq. In V. Carreño, C. Muñoz, and S. Tahar, eds., *Theorem Proving in Higher Order Logics*, NASA Conference Proceedings, 2002.

[4] A. Kock. The constructive lift monad. Technical Report RS-95-20, BRICS, 1995. `http://www.brics.dk/RS/95/20/`.

[5] P. Martin-Löf. *Intuitionistic Type Theory*. Bibliopolis, Naples, 1984.

[6] B. Nordström, K. Petersson, and J. Smith. *Programming in Martin-Löf's Type Theory*. Oxford University Press, 1990. `http://www.cs.chalmers.se/Cs/Research/Logic/book/`.

[7] E. Palmgren. Partial algebras in type theory. Unpublished note. `http://www.math.uu.se/~palmgren/`, 2002.

[8] G. Sambin. Intuitionistic formal spaces — a first communication. In D. Skordev, ed., *Mathematical Logic and its Applications*, pp. 187–204, 1987.

[9] G. Sambin. Some points in formal topology. In *Topology in Computer Science — Dagstuhl meeting in June 2000*, to appear.

[10] G. Sambin and S. Valentini. Building up a toolbox for Martin-Löf's type theory: subset theory. In G. Sambin and J. Smith, eds., *Twenty-Five Years of Constructive Type Theory*, pp. 221–244. Oxford University Press, 1995.