Quotienting the Delay Monad by Weak Bisimilarity

James Chapman, Tarmo Uustalu, and Niccolò Veltri(✉)

Institute of Cybernetics, Tallinn University of Technology,
Akadeemia tee 21, 12618 Tallinn, Estonia
{james,tarmo,niccolo}@cs.ioc.ee

Abstract. The delay datatype was introduced by Capretta [3] as a means to deal with partial functions (as in computability theory) in Martin-Löf type theory. It is a monad and it constitutes a constructive alternative to the maybe monad. It is often desirable to consider two delayed computations equal, if they terminate with equal values, whenever one of them terminates. The equivalence relation underlying this identification is called weak bisimilarity. In type theory, one commonly replaces quotients with setoids. In this approach, the delay monad quotiented by weak bisimilarity is still a monad. In this paper, we consider Hofmann's alternative approach [6] of extending type theory with inductive-like quotient types. In this setting, it is difficult to define the intended monad multiplication for the quotiented datatype. We give a solution where we postulate some principles, crucially proposition extensionality and the (semi-classical) axiom of countable choice. We have fully formalized our results in the Agda dependently typed programming language.

1 Introduction

The delay datatype was introduced by Capretta [3] as a means to deal with partial functions (as in computability theory) in Martin-Löf type theory. It is used in this setting to cope with possible non-termination of computations (as, e.g., in the unbounded search of minimalization). Inhabitants of the delay datatype are delayed values, that we call computations throughout this paper. Crucially computations can be non-terminating and not return a value at all. The delay datatype constitutes a (strong) monad, which makes it possible to deal with possibly non-terminating computations just like any other flavor of effectful computations following Moggi's general monad-based method [12]. Often, one is only interested in termination of computations and not the exact computation time. Identifying computations that only differ by finite amounts of delay corresponds to quotienting the delay datatype by weak bisimilarity. The quotient datatype is used as a constructive alternative to the maybe datatype (see, e.g., [2]) and should also be a (strong) monad.

Martin-Löf type theory does not have built-in quotient types. The most common approach to compensate for this is to mimic them by working with setoids.

M. Leucker et al. (Eds.): ICTAC 2015, LNCS 9399, pp. 110–125, 2015.
DOI: 10.1007/978-3-319-25150-9_8

But this approach has some obvious shortcomings as well, for example, the concept of a function type is changed (every function has to come with a compatibility proof) etc. An alternative approach, which we pursue here, consists in extending the theory by postulating the existence of inductive-like quotient types à la Hofmann [6]. These quotient types are ordinary types rather than setoids.

In this paper, we ask the question: is the monad structure of the delay datatype preserved under quotienting by weak bisimilarity? Morally, this ought to be the case. In the setoid approach, this works out unproblematically indeed. But with inductive-like quotient types, one meets a difficulty when attempting to reproduce the monad structure on the quotiented datatype. Specifically, one cannot define the multiplication. The difficulty has to do with the interplay of the coinductive nature of the delay datatype, or more precisely the infinity involved, and quotient types. We discuss the general phenomenon behind this issue and provide a solution where we postulate some principles, the crucial ones being proposition extensionality (accepted in particular in homotopy type theory) and the (semi-classical) axiom of countable choice. It is very important here to be careful and not postulate too much: in the presence of proposition extensionality, the full axiom of choice implies the law of excluded middle.

As an aside, we also look at the (strong) arrow structure (in the sense of Hughes [7]) on the Kleisli function type for the delay datatype and ask whether this survives quotienting by pointwise weak bisimilarity. Curiously, here the answer is unconditionally positive also for inductive-like quotient types.

This paper is organized as follows. In Sect. 2, we give an overview of the type theory we are working in. In Sect. 3, we introduce the delay datatype and weak bisimilarity. In Sect. 4, we extend type theory with quotients à la Hofmann. In Sect. 5, we analyze why a multiplication for the quotiented delay type is impossible to define. We notice that the problem is of a more general nature, and a larger class of types, namely non-wellfounded and non-finitely branching trees, suffer from it. In Sect. 6, we introduce the axiom of countable choice and derive some important consequences from postulating it. In Sect. 7, using the results of Sect. 6, we define multiplication for the delay type quotiented by weak bisimilarity (we omit the proof of the monad laws, which is the easy part—essentially the proofs for the unquotiented delay datatype carry over). In Sect. 8, we quotient the arrow corresponding to the monad by pointwise weak bisimilarity. Finally, in Sect. 9, we draw some conclusions and discuss future work.

We have fully formalized the results of this paper in the dependently typed programming language Agda [13]. The formalization is available at http://cs.ioc.ee/~niccolo/delay/.

2 The Type Theory Under Consideration

We consider Martin-Löf type theory with inductive and coinductive types and a cumulative hierarchy of universes $\mathcal{U}_k$. To define functions from inductive types or to coinductive types, we use guarded (co)recursion. The first universe is simply denoted $\mathcal{U}$ and when we write statements like "X is a type", we mean

$X : \mathcal{U}$ unless otherwise specified. We allow dependent functions to have implicit arguments and indicated implicit argument positions with curly brackets (as in Agda). We write $\equiv$ for propositional equality (identity types) and $=$ for judgmental (definitional) equality. Reflexivity, transitivity and substitutivity of $\equiv$ are named refl, trans and subst, respectively.

We assume the principle of *function extensionality*, expressing that pointwise equal functions are equal, i.e., the inhabitedness of

$$\mathsf{FunExt} = \prod_{\{X,Y:\mathcal{U}\}} \prod_{\{f_1,f_2:X\to Y\}} \left(\prod_{x:X} f_1\, x \equiv f_2\, x\right) \to f_1 \equiv f_2$$

Likewise we will assume analogous extensionality principles stating that strongly bisimilar coinductive data and proofs are equal for the relevant coinductive types and predicates, namely, the delay datatype and weak bisimilarity (check DExt, $\approx\mathsf{Ext}$ below in Sects. 3 and 4).

We also assume *uniqueness of identity proofs* for all types,[1] i.e., an inhabitant for

$$\mathsf{UIP} = \prod_{\{X:\mathcal{U}\}} \prod_{\{x_1,x_2:X\}} \prod_{p_1,p_2:x_1\equiv x_2} p_1 \equiv p_2.$$

A type X is said to be a *proposition*, if it has at most one inhabitant, i.e., if the type

$$\mathsf{isProp}\, X = \prod_{x_1,x_2:X} x_1 \equiv x_2$$

is inhabited.

For propositions, we postulate a further and less standard principle of *proposition extensionality*, stating that logically equivalent propositions are equal:[2]

$$\mathsf{PropExt} = \prod_{\{X,Y:\mathcal{U}\}} \mathsf{isProp}\, X \to \mathsf{isProp}\, Y \to X \leftrightarrow Y \to X \equiv Y$$

Here $X \leftrightarrow Y = (X \to Y) \times (Y \to X)$.

3 Delay Monad

For a given type X, each element of $\mathsf{D}\, X$ is a possibly infinite computation that returns a value of X, if it terminates. We define $\mathsf{D}\, X$ as a coinductive type by the rules

$$\frac{}{\mathsf{now}\, x : \mathsf{D}\, X} \qquad \frac{c : \mathsf{D}\, X}{\mathsf{later}\, c : \mathsf{D}\, X}$$

[1] Working in homotopy type theory [15], we would assume this principle only for 0-types, i.e., sets, and that would also be enough for our purposes.

[2] Propositions are (-1)-types and proposition extensionality is univalence for (-1)-types.

Let R be an equivalence relation on a type X. The relation lifts to an equivalence relation $\sim_R$ on $\mathsf{D}\,X$ that we call *strong R-bisimilarity*. The relation is coinductively defined by the rules

$$\frac{p : x_1 R\, x_2}{\mathsf{now}_\sim\, p : \mathsf{now}\, x_1 \sim_R \mathsf{now}\, x_2} \qquad \frac{p : c_1 \sim_R c_2}{\mathsf{later}_\sim\, p : \mathsf{later}\, c_1 \sim_R \mathsf{later}\, c_2}$$

We alternatively denote the relation $\sim_R$ with $\mathsf{D}\,R$, since strong R-bisimilarity is the functorial lifting of the relation R to $\mathsf{D}\,X$. Strong $\equiv$-bisimilarity is simply called strong bisimilarity and denoted $\sim$. While it ought to be the case morally, one cannot prove that strongly bisimilar computations are equal in Martin-Löf type theory. Therefore we postulate an inhabitant for

$$\mathsf{DExt} = \prod_{\{X:\mathcal{U}\}} \prod_{\{c_1,c_2:\mathsf{D}\,X\}} c_1 \sim c_2 \to c_1 \equiv c_2$$

We take into account another equivalence relation $\approx_R$ on $\mathsf{D}\,X$ called *weak R-bisimilarity*, which is in turn defined in terms of *convergence*. The latter is a binary relation between $\mathsf{D}\,X$ and X relating terminating computations to their values. It is inductively defined by the rules

$$\frac{p : x_1 \equiv x_2}{\mathsf{now}_\downarrow\, p : \mathsf{now}\, x_1 \downarrow x_2} \qquad \frac{p : c \downarrow x}{\mathsf{later}_\downarrow\, p : \mathsf{later}\, c \downarrow x}$$

Two computations are considered weakly R-bisimilar, if they differ by a finite number of applications of the constructor later (from where it follows classically that they either converge to R-related values or diverge). Weak R-bisimilarity is defined coinductively by the rules

$$\frac{p_1 : c_1 \downarrow x_1 \;\; p_2 : x_1 R\, x_2 \;\; p_3 : c_2 \downarrow x_2}{\downarrow_\approx p_1\, p_2\, p_3 : c_1 \approx_R c_2} \qquad \frac{p : c_1 \approx_R c_2}{\mathsf{later}_\approx\, p : \mathsf{later}\, c_1 \approx_R \mathsf{later}\, c_2}$$

Weak $\equiv$-bisimilarity is called just weak bisimilarity and denoted $\approx$. In this case, we modify the first constructor for simplicity:

$$\frac{p_1 : c_1 \downarrow x \;\; p_2 : c_2 \downarrow x}{\downarrow_\approx p_1\, p_2 : c_1 \approx c_2}$$

The delay datatype D is a (strong) monad. The unit η is the constructor now while the multiplication μ is "concatenation" of laters:

$$\begin{aligned} &\mu : \mathsf{D}\,(\mathsf{D}\,X) \to \mathsf{D}\,X \\ &\mu\,(\mathsf{now}\, c) = c \\ &\mu\,(\mathsf{later}\, c) = \mathsf{later}\,(\mu\, c) \end{aligned}$$

In the quotients-as-setoids approach, it is trivial to define the corresponding (strong) monad structure on the quotient of D by $\approx$. The role of the quotiented datatype is played by the setoid functor $\hat{\mathsf{D}}$, defined by $\hat{\mathsf{D}}\,(X, R) = (\mathsf{D}\,X, \approx_R)$. The unit $\hat{\eta}$ and multiplication $\hat{\mu}$ are just η and μ together with proofs of that the

appropriate equivalences are preserved. The unit $\hat{\eta}$ is a setoid morphism from (X, R) to $(D\,X, \approx_R)$, as $x_1\,R\,x_2 \to \mathsf{now}\,x_1 \approx_R \mathsf{now}\,x_2$ by definition of $\approx_R$. The multiplication $\hat{\mu}$ is a setoid morphism from $(\mathsf{D}\,(\mathsf{D}\,X), \approx_{\approx_R})$ to $(\mathsf{D}\,X, \approx_R)$, since $c_1 \approx_{\approx_R} c_2 \to \mu\,c_1 \approx_R \mu\,c_2$ for all $c_1, c_2 : \mathsf{D}\,(\mathsf{D}\,X)$. The monad laws hold up to $\approx_R$, since they hold up to $\sim_R$.

In this paper, our goal is to establish that the delay datatype quotiented by weak bisimilarity is a monad also in Hofmann's setting [6], where the quotient type of a given type has its propositional equality given by the equivalence relation. We discuss such quotient types in the next section.

4 Inductive-Like Quotients

In this section, we describe quotient types as particular inductive-like types introduced by M. Hofmann in his PhD thesis [6]. Let X be a type and R an equivalence relation on X. For any type Y and function $f : X \to Y$, we say that f is *R-compatible* (or simply *compatible*, when the intended equivalence relation is clear from the context), if the type

$$\mathsf{compat}\,f = \prod_{\{x_1,x_2:X\}} x_1 R x_2 \to f\,x_1 \equiv f\,x_2$$

is inhabited. The quotient of X by the relation R is described by the following data:

(i) a carrier type X/R;
(ii) a constructor $[\_] : X \to X/R$ together with a proof $\mathsf{sound} : \mathsf{compat}\,[\_]$;
(iii) a dependent eliminator: for every family of types $Y : X/R \to \mathcal{U}_k$ and function $f : \prod_{x:X} Y\,[x]$ with $p : \mathsf{dcompat}\,f$, there exists a function $\mathsf{lift}\,f\,p : \prod_{q:X/R} Y\,q$ together with a computation rule

$$\mathsf{lift}_\beta\,f\,p\,x : \mathsf{lift}\,f\,p\,[x] \equiv f\,x$$

for all $x : X$.

The predicate $\mathsf{dcompat}$ is compatibility for dependent functions $f : \prod_{x:X} Y\,[x]$:

$$\mathsf{dcompat}\,f = \prod_{\{x_1,x_2:X\}} \prod_{r:x_1 R x_2} \mathsf{subst}\,Y\,(\mathsf{sound}\,r)\,(f\,x_1) \equiv f\,x_2.$$

We postulate the existence of data (i)–(iii) for all types X and equivalence relations R on X. Notice that the predicate $\mathsf{dcompat}$ depends of the availability of sound. Also notice that, in (iii), we allow elimination on every universe $\mathcal{U}_k$. In our development, we actually eliminate only on $\mathcal{U}$ and once on $\mathcal{U}_1$ (Proposition 2).

The *propositional truncation* (or *squash*) $\|X\|$ of a type X is the quotient of X by the total relation $\lambda\,x_1\,x_2.\,\top$. We write $|\_|$ instead of $[\_]$ for the constructor of $\|X\|$. The non-dependent version of the elimination principle of $\|X\|$ is employed several times in this paper, so we spell it out: in order to construct a function of type $\|X\| \to Y$, one has to construct a constant function of type $X \to Y$.

Informally an inhabitant of $\|X\|$ corresponds to an "uninformative" proof of inhabitedness of X. For example, an inhabitant of $\|\sum_{x:X} P\,x\|$ can be thought of as a proof of there existing an element of X that satisfies P that has forgotten the information of which element satisfies the predicate P. Propositional truncation and other notions of weak or anonymous existence have been thoroughly studied in type theory [9].

We call a function $f : X \to Y$ *surjective*, if the type $\prod_{y:Y} \|\sum_{x:X} f\,x \equiv y\|$ is inhabited, and a *split epimorphism*, if the type $\|\sum_{g:Y\to X} \prod_{y:Y} f\,(g\,y) \equiv y\|$ is inhabited. We say that f is a *retraction*, if the type $\sum_{g:Y\to X} \prod_{y:Y} f\,(g\,y) \equiv y$ is inhabited. Every retraction is a split epimorphism, and every split epimorphism is surjective.

Proposition 1. *The constructor* $[\_]$ *is surjective for all quotients.*

Proof. Given a type X and an equivalence relation R on X, we define:

$$[\_]\mathsf{surj} : \prod_{q:X/R} \left\| \sum_{x:X} [x] \equiv q \right\|$$
$$[\_]\mathsf{surj} = \mathsf{lift}\,(\lambda\, x.\, |x, \mathsf{refl}|)\,p$$

The compatibility proof p is trivial, since $|x_1, \mathsf{refl}| \equiv |x_2, \mathsf{refl}|$ for all $x_1, x_2 : X$. □

A quotient X/R is said to be *effective*, if the type $\prod_{x_1,x_2:X} [x_1] \equiv [x_2] \to x_1\,R\,x_2$ is inhabited. In general, effectiveness does not hold for all quotients. But we can prove that all quotients satisfy a weaker property. We say that a quotient X/R is *weakly effective*, if the type $\prod_{x_1,x_2:X} [x_1] \equiv [x_2] \to \|x_1\,R\,x_2\|$ is inhabited.

Proposition 2. *All quotients are weakly effective.*

Proof. Let X be a type, R an equivalence relation on X and $x : X$. Consider the function $\|x\,R\,\_\| : X \to \mathcal{U}$, $\|x\,R\,\_\| = \lambda\, x'.\, \|x\,R\,x'\|$. We show that $\|x\,R\,\_\|$ is R-compatible. Let $x_1, x_2 : X$ with $x_1\,R\,x_2$. We have $x\,R\,x_1 \leftrightarrow x\,R\,x_2$ and therefore $\|x\;R\;x_1\| \leftrightarrow \|x\;R\;x_2\|$. Since propositional truncations are propositions, using proposition extensionality, we conclude $\|xRx_1\| \equiv \|xRx_2\|$. We have constructed a term $p_x : \mathsf{compat}\,\|x\,R\,\_\|$, and therefore a function $\mathsf{lift}\,\|x\,R\,\_\|\,p_x : X/R \to \mathcal{U}$ (large elimination is fundamental in order to apply lift, since $\|x\,R\,\_\| : X \to \mathcal{U}$ and $X \to \mathcal{U} : \mathcal{U}_1$). Moreover, $\mathsf{lift}\,\|x\,R\,\_\|\,p_x\,[y] \equiv \|x\,R\,y\|$ by its computation rule.

Let $[x_1] \equiv [x_2]$ for some $x_1, x_2 : X$. We have:

$$\|x_1\;R\;x_2\| \equiv \mathsf{lift}\,\|x_1 R\,\_\|\,p_{x_1}\,[x_2] \equiv \mathsf{lift}\,\|x_1\;R\;\_\|\,p_{x_1}\,[x_1] \equiv \|x_1\;R\;x_1\|$$

and $x_1\,R\,x_1$ holds, since R is reflexive. □

Notice that the constructor $[\_]$ is not a split epimorphism for all quotients. The existence of a choice of representative for each equivalence class is a non-constructive principle, since it implies the *law of excluded middle*, i.e., the inhabitedness of the following type:

$$\mathsf{LEM} = \prod_{\{X:\mathcal{U}\}} \mathsf{isProp}\,X \to X + \neg X$$

where $\neg X = X \to \bot$.

Proposition 3. *Suppose that* $[\_]$ *is a split epimorphism for all quotients. Then* LEM *is inhabited.*

Proof. Let X be a type together with a proof of $\mathsf{isProp}\,X$. We consider the equivalence relation R on Bool, $x_1\,R\,x_2 \;=\; x_1 \equiv x_2 \;+\; X$. By hypothesis we obtain $\| \sum_{\mathsf{rep}:\mathsf{Bool}/R\to\mathsf{Bool}} \prod_{q:\mathsf{Bool}/R} [\mathsf{rep}\,q] \equiv q \|$. Using the elimination principle of propositional truncation, it is sufficient to construct a constant function of type:

$$\sum_{\mathsf{rep}:\mathsf{Bool}/R\to\mathsf{Bool}} \prod_{q:\mathsf{Bool}/R} [\mathsf{rep}\,q] \equiv q \;\to\; X + \neg X$$

Let $\mathsf{rep} : \mathsf{Bool}/R \to \mathsf{Bool}$ with $[\mathsf{rep}\,q] \equiv q$ for all $q : \mathsf{Bool}/R$. We have $[\mathsf{rep}\,[x]] \equiv [x]$ for all $x : \mathsf{Bool}$, which by Proposition 2 implies $\|\mathsf{rep}\,[x]\,R\,x\|$.

Note now that the following implication (a particular form of axiom of choice on Bool) holds:

$$\mathsf{acBool} : \prod_{x:\mathsf{Bool}} \|\mathsf{rep}\,[x]\,R\,x\| \to \left\| \prod_{x:\mathsf{Bool}} \mathsf{rep}\,[x]\,R\,x \right\|$$
$$\mathsf{acBool}\,r = \mathsf{lift}_2\,(\lambda\,r_1\,r_2.\,|d\,r_1\,r_2|)\,p\,(r\,\mathsf{true})\,(r\,\mathsf{false})$$

where $d\,r_1\,r_2\,\mathsf{true} = r_1$ and $d\,r_1\,r_2\,\mathsf{false} = r_2$, and lift_2 is the two-argument version of lift. The compatibility proof p is immediate, since the return type is a proposition.

We now construct a function of type $\|\prod_{x:\mathsf{Bool}} \mathsf{rep}\,[x]\,R\,x\| \to X + \neg X$. It is sufficient to define a function $\prod_{x:\mathsf{Bool}} \mathsf{rep}\,[x]\,R\,x \to X + \neg X$ (it will be constant, since the type $X + \neg X$ is a proposition, if X is a proposition), so we suppose $\mathsf{rep}\,[x]\;R\;x$ for all $x : \mathsf{Bool}$. We analyze the (decidable) equality $\mathsf{rep}\,[\mathsf{true}] \equiv \mathsf{rep}\,[\mathsf{false}]$ on Bool. If it holds, then we have $\mathsf{true}\,R\,\mathsf{false}$ and therefore an inhabitant of X. If it does not hold, we have an inhabitant of $\neg X$: let $x : X$, therefore $\mathsf{true}\;R\;\mathsf{false}$, and this implies $\mathsf{rep}\,[\mathsf{true}] \equiv \mathsf{rep}\,[\mathsf{false}]$ holds, which contradicts the hypothesis. □

We already noted that not all quotients are effective. In fact, postulating effectiveness for all quotients implies LEM [10]. But the quotient we are considering in this paper, namely $\mathsf{D}\,X/{\approx}$ for a type X, is indeed effective. Notice that, by Proposition 2, it suffices to prove that $\|c_1 \approx c_2\| \to c_1 \approx c_2$ for all $c_1, c_2 : \mathsf{D}\,X$.

Lemma 1. *For all types* X *and* $c_1, c_2 : \mathsf{D}\,X$*, there exists a constant endofunction on* $c_1 \approx c_2$*. Therefore, the type* $\|c_1 \approx c_2\| \to c_1 \approx c_2$ *is inhabited.*

Proof. Let X be a type and $c_1, c_2 : \mathsf{D}\,X$. We consider the following function.

$$\begin{aligned}
&\mathsf{canon}{\approx} : c_1 \approx c_2 \to c_2 \approx c_2\\
&\mathsf{canon}{\approx}\,(\downarrow_{\approx}(\mathsf{now}_{\downarrow}\,p_1)\,p_2) = \downarrow_{\approx}(\mathsf{now}_{\downarrow}\,p_1)\,p_2\\
&\mathsf{canon}{\approx}\,(\downarrow_{\approx}(\mathsf{later}_{\downarrow}\,p_1)\,(\mathsf{now}_{\downarrow}\,p_2)) = \downarrow_{\approx}(\mathsf{later}_{\downarrow}\,p_1)\,(\mathsf{now}_{\downarrow}p_2)\\
&\mathsf{canon}{\approx}\,(\downarrow_{\approx}(\mathsf{later}_{\downarrow}\,p_1)\,(\mathsf{later}_{\downarrow}\,p_2)) = \mathsf{later}_{\approx}\,(\mathsf{canon}{\approx}\,(\downarrow_{\approx}p_1\,p_2))\\
&\mathsf{canon}{\approx}\,(\mathsf{later}_{\approx}\,p) = \mathsf{later}_{\approx}\,(\mathsf{canon}{\approx}\,p)
\end{aligned}$$

The function $\mathsf{canon}{\approx}$ canonizes a given weak bisimilarity proof by maximizing the number of applications of the constructor $\mathsf{later}_{\approx}$. This function is indeed constant, i.e., one can prove $\prod_{p_1,p_2:c_1\approx c_2} p_1 \cong p_2$ for all $c_1, c_2 : \mathsf{D}\,X$, where the relation $\cong$ is strong bisimilarity on proofs of $c_1 \approx c_2$, coinductively defined by the rules:

$$\frac{}{\downarrow_{\approx} p_1\, p_2 \cong \downarrow_{\approx} p_1\, p_2} \qquad \frac{p_1 \cong p_2}{\mathsf{later}_{\approx}\, p_1 \cong \mathsf{later}_{\approx}\, p_2}$$

Similarly to extensionality of delayed computations, we assume that strongly bisimilar weak bisimilarity proofs are equal, i.e., that we have an inhabitant for

$$\approx\mathsf{Ext} = \prod_{\{X:\mathcal{U}\}} \prod_{\{c_1,c_2:\mathsf{D}\,X\}} \prod_{p_1,p_2:c_1\approx c_2} p_1 \cong p_2 \to p_1 \equiv p_2 \qquad \square$$

5 Multiplication: What Goes Wrong?

Consider now the type functor $\bar{\mathsf{D}}$, defined by $\bar{\mathsf{D}}\,X = \mathsf{D}\,X/{\approx}$. Let us try to equip it with a monad structure. Let X be a type. As the unit $\bar{\eta} : X \to \mathsf{D}\,X/{\approx}$, we can take $[\_] \circ \mathsf{now}$. But when we try to construct a multiplication $\bar{\mu} : \mathsf{D}\,(\mathsf{D}\,X/{\approx})/{\approx} \to \mathsf{D}\,X/{\approx}$, we get stuck immediately. Indeed, $\bar{\mu}$ must be of the form $\mathsf{lift}\,\bar{\mu}'\,p$ for some $\bar{\mu}' : \mathsf{D}\,(\mathsf{D}\,X/{\approx}) \to \mathsf{D}\,X/{\approx}$ with $p : \mathsf{compat}\,\bar{\mu}'$, but we cannot define such $\bar{\mu}'$ and p. The problem lies in the coinductive nature of the delay datatype. A function of type $\mathsf{D}\,(\mathsf{D}\,X/{\approx}) \to \mathsf{D}\,X/{\approx}$ should send a converging computation to its converging value and a non-terminating one to the equivalence class of non-termination. This discontinuity makes constructing such a function problematic. Moreover, one can show that a right inverse of $[\_] : \mathsf{D}\,X \to \mathsf{D}\,X/{\approx}$, i.e., a canonical choice of representative for each equivalence class in $\mathsf{D}\,X/{\approx}$, is not definable. Therefore, we cannot even construct $\bar{\mu}'$ as a composition $[\_] \circ \bar{\mu}''$ with $\bar{\mu}'' : \mathsf{D}\,(\mathsf{D}\,X/{\approx}) \to \mathsf{D}\,X$, since we do not know how to define $\bar{\mu}''(\mathsf{now}\,q)$ for $q : \mathsf{D}\,X/{\approx}$.

A function $\bar{\mu}'$ would be constructable, if the type $\mathsf{D}\,(\mathsf{D}\,X/{\approx})$ were a quotient of $\mathsf{D}\,(\mathsf{D}\,X)$ by the equivalence relation $\mathsf{D}{\approx}$ (remember that $\mathsf{D}{\approx}$ is a synonym of $\sim_{\approx}$, the functorial lifting of $\approx$ from $\mathsf{D}\,X$ to $\mathsf{D}\,(\mathsf{D}\,X)$). In fact, the function $[\_] \circ \mu : \mathsf{D}\,(\mathsf{D}\,X) \to \mathsf{D}\,X/{\approx}$ is $\mathsf{D}{\approx}$-compatible, since $x_1(\mathsf{D}{\approx})x_2 \to \mu\,x_1 \approx \mu\,x_2$, and therefore the elimination principle would do the job. But how "different" are $\mathsf{D}\,(\mathsf{D}\,X/{\approx})$ and the quotient $\mathsf{D}\,(\mathsf{D}\,X)/\mathsf{D}{\approx}$? More generally, how "different" are $\mathsf{D}\,(X/R)$ and the quotient $\mathsf{D}\,X/\mathsf{D}\,R$, for a given type X and equivalence relation R on X?

A function $\theta^{\mathsf{D}} : \mathsf{D}\,X/\mathsf{D}\,R \to \mathsf{D}\,(X/R)$ always exists, $\theta^{\mathsf{D}} = \mathsf{lift}\,(\mathsf{D}\,[\_])\,p$. The compatibility proof p follows directly from $c_1(\mathsf{D}\,R)c_2 \to \mathsf{D}\,[\_]\,c_1 \sim \mathsf{D}\,[\_]\,c_2$. But an inverse function $\psi^{\mathsf{D}} : \mathsf{D}\,(X/R) \to \mathsf{D}\,X/\mathsf{D}\,R$ is not definable. This phenomenon can be spotted more generally in non-wellfounded trees, i.e., the canonical function $\theta^{T} : T\,X/T\,R \to T\,(X/R)$ does not have an inverse, if $T\,X$ is coinductively defined, where $T\,R$ is the functorial lifting of R to $T\,X$. On the other hand, a large class of purely inductive types, namely, the datatypes of

wellfounded trees where branching is finite, is free of this problem. As an example, for binary trees the inverse $\psi^{\mathsf{BTree}} : \mathsf{BTree}\,(X/R) \to \mathsf{BTree}\,X/\mathsf{BTree}\,R$ of $\theta^{\mathsf{BTree}} : \mathsf{BTree}\,X/\mathsf{BTree}\,R \to \mathsf{BTree}\,(X/R)$ is defined as follows:

$$\begin{aligned}
&\psi^{\mathsf{BTree}} : \mathsf{BTree}\,(X/R) \to \mathsf{BTree}\,X/\mathsf{BTree}\,R \\
&\psi^{\mathsf{BTree}}\,(\mathsf{leaf}\,q) = \mathsf{lift}\,(\lambda\,x.\,[\mathsf{leaf}\,x])\,p_{\mathsf{leaf}}\,q \\
&\psi^{\mathsf{BTree}}\,(\mathsf{node}\,t_1\,t_2) = \mathsf{lift}_2\,(\lambda\,s_1\,s_2.\,[\mathsf{node}\,s_1\,s_2])\,p_{\mathsf{node}}\,(\psi^{\mathsf{BTree}}\,t_1)\,(\psi^{\mathsf{BTree}}\,t_2)
\end{aligned}$$

where lift_2 is the two-argument version of lift. The simple compatibility proofs p_{leaf} and p_{node} are omitted. Wellfounded non-finitely branching trees are affected by the same issues that non-wellfounded trees have. And in general, for a W-type T, the function $\theta^T : T\,X/T\,R \to T\,(X/R)$ is not invertible, since for function spaces the function $\theta^{\to} : (Y \to X)/(Y \to R) \to (Y \to X/R)$ is not invertible. Invertibility of the function $\theta^{\to} : (Y \to X)/(Y \to R) \to (Y \to X/R)$, for all types Y, X and equivalence relation R on X, has been analyzed in the Calculus of Inductive Constructions [4]. It turns out that surjectivity of $\theta^{\to}$ is logically equivalent to the full axiom of choice (AC)[3], i.e., the following type is inhabited:

$$\mathsf{AC} = \prod_{\{X,Y:\mathcal{U}\}} \prod_{P:X\to Y\to\mathcal{U}} \left(\prod_{x:X} \left\|\sum_{y:Y} P\,x\,y\right\|\right) \to \left\|\sum_{f:X\to Y} \prod_{x:X} P\,x\,(f\,x)\right\|$$

Together with weak effectiveness (Proposition 2), AC not only implies surjectivity of $\theta^{\to}$, but also the existence of an inverse $\psi^{\to} : (Y \to X/R) \to (Y \to X)/(Y \to R)$. We refrain from proving these facts, but we prove Lemma 2 and Proposition 5, which are weaker statements, but have analogous proofs.

The existence of an inverse $\psi^{\to}$ of $\theta^{\to}$ would immediately allow us to define the bind operation for $\bar{\mathsf{D}}$. Let us consider the case where X is $\mathsf{D}\,X$ and R is weak bisimilarity, so $\psi^{\to} : (Y \to \mathsf{D}\,X/{\approx}) \to (Y \to \mathsf{D}\,X)/(Y \to {\approx})$. We define

$$\begin{aligned}
&\overline{\mathsf{bind}} : (Y \to \mathsf{D}\,X/{\approx}) \to \mathsf{D}\,Y/{\approx} \to \mathsf{D}\,X/{\approx} \\
&\overline{\mathsf{bind}}\,f\,q = \mathsf{lift}_2\,(\lambda\,g\,c.\,[\mathsf{bind}\,g\,c])\,p\,(\psi^{\to}\,f)\,q
\end{aligned}$$

where bind is the bind operation of the unquotiented delay monad. The compatibility proof p is obtained from the fact that $\mathsf{bind}\,g_1\,c_1 \approx \mathsf{bind}\,g_2\,c_2$ if $c_1 \approx c_2$ and $g_1\,y \approx g_2\,y$ for all $y : Y$.

AC is a controversial semi-classical axiom, generally not accepted in constructive systems [11]. We reject it too, since in our system the axiom of choice implies the law of excluded middle.

[3] Notice that AC is fundamentally different from the *type-theoretic* axiom of choice:

$$\prod_{\{X,Y:\mathcal{U}\}} \prod_{P:X\to Y\to\mathcal{U}} \left(\prod_{x:X} \sum_{y:Y} P\,x\,y\right) \to \sum_{f:X\to Y} \prod_{x:X} P\,x\,(f\,x)$$

which is provable in type theory.

Proposition 4. AC *implies* LEM.

Proof. Assume AC. With a proof analogous to Lemma 2, we can prove that the function $\lambda f.\, [\_] \circ f : (X \to Y) \to (X \to Y/R)$ is surjective, for any types X, Y and equivalence relation R on Y. In particular, given a type X and an equivalence relation R on X, we have that the type $\prod_{g:X/R\to X/R} \left\| \sum_{f:X/R\to X} [\_] \circ f \equiv g \right\|$ is inhabited. Instantiating g with the identity function on X/R, we obtain $\left\| \sum_{f:X/R\to X} \prod_{q:X/R} [f\, q] \equiv q \right\|$, i.e., the constructor $[\_]$ is a split epimorphism for all quotients X/R. By Proposition 3, this implies LEM. □

In the following sections, we show that the weaker axiom of countable choice is already enough for constructing a multiplication for $\bar{\mathsf{D}}$. Countable choice does not imply excluded middle and constructive mathematicians like it more [14, Ch. 4].

6 Axiom of Countable Choice and Streams of Quotients

The axiom of countable choice ($\mathsf{AC}\omega$) is a specific instance of AC where the binary predicate P has its first argument in $\mathbb{N}$:

$$\mathsf{AC}\omega = \prod_{\{X:\mathcal{U}\}} \prod_{P:\mathbb{N}\to X\to\mathcal{U}} \left(\prod_{n:\mathbb{N}} \left\| \sum_{x:X} P\, n\, x \right\| \right) \to \left\| \sum_{f:\mathbb{N}\to X} \prod_{n:\mathbb{N}} P\, n\, (f\, n) \right\|$$

We also introduce a logically equivalent formulation of $\mathsf{AC}\omega$ that will be used in Proposition 5:

$$\mathsf{AC}\omega_2 = \prod_{P:\mathbb{N}\to\mathcal{U}} \left(\prod_{n:\mathbb{N}} \|P\, n\| \right) \to \left\| \prod_{n:\mathbb{N}} P\, n \right\|$$

Let X be a type and R an equivalence relation on it. We show that $\mathsf{AC}\omega$ implies the surjectivity of the function $[\_]^{\mathbb{N}} : (\mathbb{N} \to X) \to (\mathbb{N} \to X/R)$, $[f]^{\mathbb{N}}\, n = [f\, n]$. This in turn implies the definability of a function $\psi^{\mathbb{N}} : (\mathbb{N} \to X/R) \to (\mathbb{N} \to X)/(\mathbb{N} \to R)$ that is inverse of the canonical function $\theta^{\mathbb{N}} = \mathsf{lift}\, [\_]^{\mathbb{N}}\, \mathsf{sound}^{\mathbb{N}}$, where

$$\begin{aligned} &\mathsf{sound}^{\mathbb{N}} : \mathsf{compat}\, [\_]^{\mathbb{N}} \\ &\mathsf{sound}^{\mathbb{N}}\, r = \mathsf{funext}\, (\lambda n.\, \mathsf{sound}\, (r\, n)). \end{aligned}$$

using $\mathsf{funext} : \mathsf{FunExt}$.

Lemma 2. *Assume* $\mathsf{ac}\omega : \mathsf{AC}\omega$. *Then* $[\_]^{\mathbb{N}}$ *is surjective.*

Proof. Given any $g : \mathbb{N} \to X/R$, we construct a term $e_g : \left\| \sum_{f:\mathbb{N}\to X} [f]^{\mathbb{N}} \equiv g \right\|$. Since we are assuming the principle of function extensionality, it is sufficient to find a term $e'_g : \left\| \sum_{f:\mathbb{N}\to X} \prod_{n:\mathbb{N}} [f\, n] \equiv g\, n \right\|$. Define $P : \mathbb{N} \to X \to \mathcal{U}$ by $P\, n\, x = [x] \equiv g\, n$. We take $e'_g = \mathsf{ac}\omega\, P\, (\lambda n.\, [\_]\mathsf{surj}\, (g\, n))$, with $[\_]\mathsf{surj}$ introduced in Proposition 1. □

Proposition 5. *Assume* $\mathsf{AC}\omega$. *Then* $\theta^{\mathbb{N}} : (\mathbb{N} \to X)/(\mathbb{N} \to R) \to (\mathbb{N} \to X/R)$ *is invertible.*

Proof. We construct a term

$$r : \sum_{\psi^{\mathbb{N}}:(\mathbb{N}\to X/R)\to(\mathbb{N}\to X)/(\mathbb{N}\to R)} \prod_{g:\mathbb{N}\to X/R} \theta^{\mathbb{N}}\,(\psi^{\mathbb{N}}\,g) \equiv g$$

Given any $g : \mathbb{N} \to X/R$, we define:

$$h'_g : \left(\sum_{f:\mathbb{N}\to X} [f]^{\mathbb{N}} \equiv g\right) \to \sum_{q:(\mathbb{N}\to X)/(\mathbb{N}\to R)} \theta^{\mathbb{N}}\,q \equiv g$$
$$h'_g\,(f,p) = \Big([f], \mathsf{trans}\,(\mathsf{lift}_\beta\,[\_]^{\mathbb{N}}\,\mathsf{sound}^{\mathbb{N}}\,f)\,p\Big)$$

The function h'_g is constant. Indeed, let $f_1, f_2 : \mathbb{N} \to X$ with $p_1 : [f_1]^{\mathbb{N}} \equiv g$ and $p_2 : [f_2]^{\mathbb{N}} \equiv g$. By uniqueness of identity proofs, it is sufficient to show $[f_1] \equiv [f_2]$. By symmetry and transitivity, we get $[f_1]^{\mathbb{N}} \equiv [f_2]^{\mathbb{N}}$. We construct the following series of implications:

$$\begin{aligned}
[f_1]^{\mathbb{N}} \equiv [f_2]^{\mathbb{N}} &\to \prod_{n:\mathbb{N}} [f_1\,n] \equiv [f_2\,n] \\
&\to \prod_{n:\mathbb{N}} \|(f_1\,n)\,R\,(f_2\,n)\| \qquad \text{(by weak effectiveness)} \\
&\to \left\| \prod_{n:\mathbb{N}} (f_1\,n)\,R\,(f_2\,n) \right\| \quad \text{(by } \mathsf{AC}\omega \text{ and } \mathsf{AC}\omega \to \mathsf{AC}\omega_2) \\
&= \|f_1\,(\mathbb{N} \to R)\,f_2\| \\
&\to [f_1] \equiv [f_2]
\end{aligned}$$

The last implication is given by the elimination principle of propositional truncation applied to sound, which is a constant function by uniqueness of identity proofs. Therefore h'_g is constant and we obtain a function

$$h_g : \left\| \sum_{f:\mathbb{N}\to X} [f]^{\mathbb{N}} \equiv g \right\| \to \sum_{q:(\mathbb{N}\to X)/(\mathbb{N}\to R)} \theta^{\mathbb{N}}\,q \equiv g$$

We get $h_g\,e_g : \sum_{q:(\mathbb{N}\to X)/(\mathbb{N}\to R)} \theta^{\mathbb{N}}\,q \equiv g$, with e_g constructed in Lemma 2. We take $r = (\lambda g.\,\mathsf{fst}\,(h_g\,e_g), \lambda g.\,\mathsf{snd}\,(h_g\,e_g))$ and $\psi^{\mathbb{N}} = \mathsf{fst}\,r$.

We now prove that $\psi^{\mathbb{N}}\,(\theta^{\mathbb{N}}\,q) \equiv q$ for all $q : (\mathbb{N} \to X)/(\mathbb{N} \to R)$. It is sufficient to prove this equality for $q = [f]$ with $f : \mathbb{N} \to X$. By the computation rule of quotients, we have to show $\psi^{\mathbb{N}}\,[f]^{\mathbb{N}} \equiv [f]$. This is true, since

$$\psi^{\mathbb{N}}\,[f]^{\mathbb{N}} = \mathsf{fst}\,(h_{[f]^{\mathbb{N}}}\,e_{[f]^{\mathbb{N}}}) \equiv \mathsf{fst}\,(h_{[f]^{\mathbb{N}}}\,|f, \mathsf{refl}|) \equiv \mathsf{fst}\,(h'_{[f]^{\mathbb{N}}}(f, \mathsf{refl})) = [f] \qquad \square$$

Corollary 1. *Assume* $\mathsf{AC}\omega$. *The type* $\mathbb{N} \to X/R$ *is the carrier of a quotient of* $\mathbb{N} \to X$ *by the equivalence relation* $\mathbb{N} \to R$. *The constructor is* $[\_]^{\mathbb{N}}$ *and we have the following dependent eliminator and computation rule: for every family of types* $Y : (\mathbb{N} \to X/R) \to \mathcal{U}_k$ *and function* $h : \prod_{f:\mathbb{N}\to X} Y\,[f]^{\mathbb{N}}$ *with* $p : \mathsf{dcompat}^{\mathbb{N}}\,h$, *there exists a function* $\mathsf{lift}^{\mathbb{N}}\,h\,p : \prod_{g:\mathbb{N}\to X/R} Y\,g$ *with the property that* $\mathsf{lift}^{\mathbb{N}}\,h\,p\,[f]^{\mathbb{N}} \equiv h\,f$ *for all* $f : \mathbb{N} \to X$, *where*

$$\mathsf{dcompat}^{\mathbb{N}}\,h = \prod_{\{f_1,f_2:\mathbb{N}\to X\}} \prod_{r:f_1\ (\mathbb{N}\to R)\ f_2} \mathsf{subst}\,Y\,(\mathsf{sound}^{\mathbb{N}}\,r)\,(h\,f_1) \equiv h\,f_2$$

7 Multiplication: A Solution Using $\mathsf{AC}\omega$

We can now build the desired monad structure on $\bar{\mathsf{D}}$ using the results proved in Sect. 6. In particular, we can define $\bar{\mu} : \mathsf{D}\,(\mathsf{D}\,X/{\approx})/{\approx} \to \mathsf{D}\,X/{\approx}$. We rely on $\mathsf{AC}\omega$.

7.1 Delayed Computations as Streams

In order to use the results of Sect. 6, we think of possibly non-terminating computations as streams. More precisely, let X be a type and $c : \mathsf{D}\,X$. Now c can be thought of as a stream $\varepsilon\,c : \mathbb{N} \to X + 1$ with at most one value element in the left summand X.

$$\begin{aligned}
&\varepsilon : \mathsf{D}\,X \to \mathbb{N} \to X + 1\\
&\varepsilon\,(\mathsf{now}\,x)\,\mathsf{zero} = \mathsf{inl}\,x\\
&\varepsilon\,(\mathsf{later}\,c)\,\mathsf{zero} = \mathsf{inr}\,\star\\
&\varepsilon\,(\mathsf{now}\,x)\,(\mathsf{suc}\,n) = \mathsf{inr}\,\star\\
&\varepsilon\,(\mathsf{later}\,c)\,(\mathsf{suc}\,n) = \varepsilon\,c\,n
\end{aligned}$$

Conversely, from a stream $f : \mathbb{N} \to X + 1$, one can construct a computation $\pi\,f : \mathsf{D}\,X$. This computation corresponds to the "truncation" of the stream to its first value in X.

$$\begin{aligned}
&\pi : (\mathbb{N} \to X + 1) \to \mathsf{D}\,X\\
&\pi\,f = \mathsf{case}\ f\,\mathsf{zero}\ \mathsf{of}\\
&\qquad \mathsf{inl}\,x \mapsto \mathsf{now}\,x\\
&\qquad \mathsf{inr}\,\star \mapsto \mathsf{later}\,(\pi\,(f \circ \mathsf{suc}))
\end{aligned}$$

We see that $\mathsf{D}\,X$ is a subset of $\mathbb{N} \to X + 1$ in the sense that, for all $c : \mathsf{D}\,X$, $\pi\,(\varepsilon\,c) \sim c$, and therefore $\pi(\varepsilon\,c) \equiv c$ by delayed computation extensionality.

Now let R be an equivalence relation on X. The canonical function $\theta^{+1} : (X+1)/(R+1) \to X/R+1$ has an inverse ψ^{+1} whose construction is similar to the construction of ψ^{BTree} for binary trees in Sect. 5. Therefore, for all $q : \mathsf{D}\,(X/R)$, we have $\pi\,(\theta^{+1} \circ (\psi^{+1} \circ \varepsilon\,q)) \equiv q$.

We define $[\_]^{\mathsf{D}} : \mathsf{D}\,X \to \mathsf{D}\,(X/R)$ by $[\_]^{\mathsf{D}} = \mathsf{D}\,[\_]$. This function is compatible with the relation $\mathsf{D}\,R$, i.e., there exists a term $\mathsf{sound}^{\mathsf{D}} : \mathsf{compat}\,[\_]^{\mathsf{D}}$.

Theorem 1. *The type* $\mathsf{D}\,(X/R)$ *is the carrier of a quotient of* $\mathsf{D}\,X$ *by the equivalence relation* $\mathsf{D}\,R$*. The constructor is* $[\_]^{\mathsf{D}}$ *and we have the following dependent eliminator and computation rule: for every family of types* $Y : \mathsf{D}\,(X/R) \to \mathcal{U}_k$ *and function* $h : \prod_{c:\mathsf{D}\,X} Y\,[c]^{\mathsf{D}}$ *with* $p : \mathsf{dcompat}^{\mathsf{D}}\,h$*, there exists a function* $\mathsf{lift}^{\mathsf{D}}\,h\,p : \prod_{q:\mathsf{D}\,(X/R)} Y\,q$ *such that* $\mathsf{lift}^{\mathsf{D}}\,h\,p\,[c]^{\mathsf{D}} \equiv h\,c$ *for all* $c : \mathsf{D}\,X$*, where*

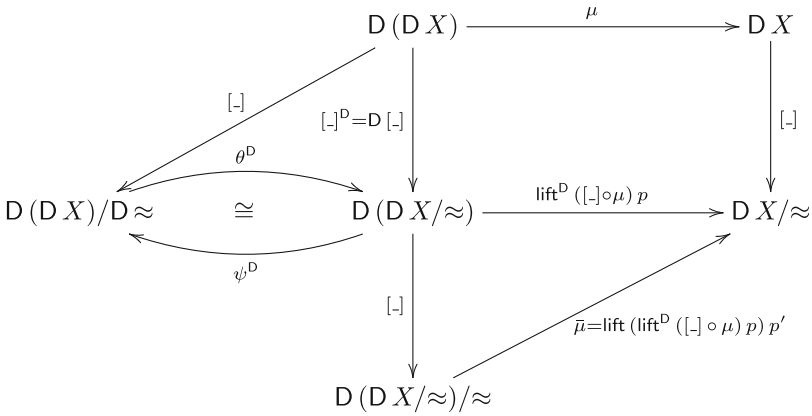
$$\mathsf{dcompat}^{\mathsf{D}}\,h = \prod_{\{c_1,c_2:\mathsf{D}\,X\}} \prod_{r:c_1\,(\mathsf{D}\,R)\,c_2} \mathsf{subst}\,Y\,(\mathsf{sound}^{\mathsf{D}}\,r)\,(h\,c_1) \equiv h\,c_2$$

Proof. We only define the dependent eliminator. Let $h : \prod_{x:\mathsf{D}\,X} Y\,[x]^{\mathsf{D}}$ with $p : \mathsf{dcompat}^{\mathsf{D}}\,h$, and $q : \mathsf{D}\,(X/R)$. Let $g : \mathbb{N} \to (X+1)/(R+1)$, $g = \psi^{+1} \circ \epsilon\,q$ so $\pi\,(\theta^{+1} \circ g) \equiv q$.

We prove $Y(\pi\,(\theta^{+1} \circ g))$. By Corollary 1, it suffices to construct a function $h' : \prod_{f:\mathbb{N}\to X+1} Y(\pi\,(\theta^{+1} \circ [f]^{\mathbb{N}}))$ together with a proof $r : \mathsf{dcompat}^{\mathbb{N}}\,h'$. One can easily construct a proof $s : [\pi\,f]^{\mathsf{D}} \equiv \pi\,(\theta^{+1} \circ [f]^{\mathbb{N}})$, so we take $h'\,f = \mathsf{subst}\,Y\,s\,(h\,(\pi\,f))$. A proof $r : \mathsf{dcompat}^{\mathbb{N}}\,h'$ can be constructed by observing that, for all $f_1, f_2 : \mathbb{N} \to X+1$ satisfying $f_1\,(\mathbb{N} \to R+1)\,f_2$, one can prove $\pi\,f_1\,(\mathsf{D}\,R)\,\pi\,f_2$. □

7.2 Construction of $\bar{\mu}$

Using the elimination rule of the quotient $\mathsf{D}\,(X/R)$ defined in Theorem 1, we can finally define the multiplication $\bar{\mu}$ of $\bar{\mathsf{D}}$.



The above diagram makes sense only, if one constructs two compatibility proofs $p : \mathsf{compat}^{\mathsf{D}}\,([\_] \circ \mu)$ and $p' : \mathsf{compat}\,(\mathsf{lift}^{\mathsf{D}}\,([\_] \circ \mu)\,p)$, where $\mathsf{compat}^{\mathsf{D}}$ is the non-dependent version of $\mathsf{dcompat}^{\mathsf{D}}$.

The first proof is easy, since $c_1(\mathsf{D}\approx)c_2 \to \mu\,c_1 \approx \mu\,c_2$ for all $c_1, c_2 : \mathsf{D}\,(\mathsf{D}\,X)$.

It is more complicated to prove compatibility of the second function. Let $q_1, q_2 : \mathsf{D}\,(\mathsf{D}\,X/\approx)$. We have to show $q_1 \approx q_2 \to \mathsf{lift}^{\mathsf{D}}\,([\_] \circ \mu)\,p\,q_1 \equiv \mathsf{lift}^{\mathsf{D}}\,([\_] \circ \mu)\,p\,q_2$. By the elimination principle of the quotient $\mathsf{D}\,(\mathsf{D}\,X/\approx)$, described in Theorem 1, it is sufficient to prove $[x_1]^{\mathsf{D}} \approx [x_2]^{D} \to \mathsf{lift}^{\mathsf{D}}\,([\_] \circ \mu)\,p\,[c_1]^{D} \equiv \mathsf{lift}^{\mathsf{D}}\,([\_] \circ \mu)\,p\,[c_2]^{D}$ for some $c_1, c_2 : \mathsf{D}\,(\mathsf{D}\,X)$. Applying the computation rule of

the quotient $\mathsf{D}\,(\mathsf{D}\,X/\approx)$ and spelling out the definition of the constructor $[\_]^{\mathsf{D}}$, it remains to show $\mathsf{D}\,[\_]\,c_1 \approx \mathsf{D}\,[\_]\,c_2 \to [\mu\,c_1] \equiv [\mu\,c_2]$, which holds, if one can prove $\mathsf{D}\,[\_]\,c_1 \approx \mathsf{D}\,[\_]\,c_2 \to \mu\,c_1 \approx \mu\,c_2$. This is provable thanks to Lemma 1. It is easy to see why Lemma 1 is important for completing the compatibility proof of $\mathsf{lift}^{\mathsf{D}}\,([\_] \circ \mu)\,p$. The difficult case in the proof of $\mathsf{D}\,[\_]\,c_1 \approx \mathsf{D}\,[\_]\,c_2 \to \mu\,c_1 \approx \mu\,c_2$ is the case where $c_1 = \mathsf{now}\,y_1$ and $c_2 = \mathsf{now}\,y_2$, so we are given an assumption of type $[y_1] \equiv [y_2]$. From this, by Lemma 1, we obtain $\mu\,(\mathsf{now}\,y_1) = y_1 \approx y_2 = \mu\,(\mathsf{now}\,y_2)$.

8 A Monad or an Arrow?

Hughes [7] has proposed arrows as a generalization of monads. Jacobs et al. [8] have sorted out their mathematical theory.

We have seen that it takes a semi-classical principle to show that quotienting the functor D by weak bisimilarity preserves its monad structure. In contrast, quotienting the corresponding profunctor KD, defined by $\mathsf{KD}\,X\,Y = X \to \mathsf{D}\,Y$, by pointwise weak bisimilarity can easily be shown to preserve its (strong) arrow structure (whose Freyd category is isomorphic to the Kleisli category of the monad) without invoking such principles.

Indeed, the arrow structure on KD is given by $\mathsf{pure} : (X \to Y) \to \mathsf{KD}\,X\,Y$, $\mathsf{pure}\,f = \eta \circ f$ and $\lll : \mathsf{KD}\,Y\,Z \to \mathsf{KD}\,X\,Y \to \mathsf{KD}\,X\,Z$, $\ell \lll k = \mathsf{bind}\,\ell \circ k$.

Now, define the quotiented profunctor by $\overline{\mathsf{KD}}\,X\,Y = (X \to \mathsf{D}\,Y)/(X \to \approx)$. We can define $\overline{\mathsf{pure}} : (X \to Y) \to \overline{\mathsf{KD}}\,X\,Y$ straightforwardly by $\overline{\mathsf{pure}}\,f = [\mathsf{pure} f]$. But we can also construct $\overline{\lll} : \overline{\mathsf{KD}}\,Y\,Z \to \overline{\mathsf{KD}}\,X\,Y \to \overline{\mathsf{KD}}\,X\,Z$ as $\ell \overline{\lll} k = \mathsf{lift}_2\,(\lll)\,p\,\ell\,k$, where p is an easy proof of $\ell_1\ (Y \to \approx)\ \ell_2 \to k_1\ (X \to \approx)\ k_2 \to (\ell_1 \lll k_1)\ (X \to \approx)\ (\ell_2 \lll k_2)$.

This works entirely painlessly, as there is no need in this construction for a coercion $(X \to Y/\approx) \to (X \to Y)/(X \to \approx)$ (cf. the discussion above in Sect. 5). From the beginning, we quotient the relevant function types here rather than their codomains.

There are some further indications that quotienting the arrow may be a righter thing to do than quotienting the monad. In particular, the work by Cockett et al. [5] suggests that working with finer quotients of the arrow considered here may yield a setting for dealing with computational complexity rather computability constructively.

9 Conclusions

In this paper, studied the question of whether the delay datatype quotiented by weak bisimilarity is still a monad? As we saw, different approaches to quotients in type theory result in different answers. In the quotients-as-setoids, the answer is immediately positive. We focussed on the more interesting and (as it turned out) more difficult case of the quotient types à la Hofmann. The main issue in this case, highlighted in Sect. 5, is that quotient types interact badly with infinite datatypes, such as datatypes of non-wellfounded or non-finitely branching

trees; such datatypes do not commute with quotienting. For the delay datatype, and more generally for types that can be injectively embedded into streams or countably branching trees, a solution is possible assuming the axiom of countable choice.

In the type theory that we are considering, the employment of semi-classical principles, such as countable choice, is unavoidable. In homotopy type theory with higher inductive types [15, Ch. 6], the problem may have a different solution. One might be able to implement the delay type quotiented by weak bisimilarity as an higher inductive type, proceeding similarly to the construction of Cauchy reals in [15, Sect. 11.3], mutually defining the type and the equivalence relation, and adding a 1-constructor stating that the equivalence has to be read as equality. Note that this technique is not immediately applicable, since the delay datatype is coinductive and weak bisimilarity is mixed inductive-coinductive. One would have to come up with a different construction. We think that the idea should be to construct the intended monad as a datatype delivering free completely Elgot algebras [1]. Notice that this would be analogous to the already mentioned implementation of Cauchy reals, which are constructed as the free completion of the rational numbers.

Acknowledgement. We thank Thorsten Altenkirch, Andrej Bauer, Bas Spitters and our anonymous referees for comments.

This research was supported by the ERDF funded Estonian CoE project EXCS and ICT national programme project "Coinduction", the Estonian Science Foundation grants No. 9219 and 9475 and the Estonian Ministry of Education and Research institutional research grant IUT33-13.

References

1. Adámek, J., Milius, S., Velebil, J.: Elgot algebras. Log. Methods Comput. Sci. **2**(5:4), 1–31 (2006)
2. Benton, N., Kennedy, A., Varming, C.: Some domain theory and denotational semantics in Coq. In: Berghofer, S., Nipkow, T., Urban, C., Wenzel, M. (eds.) TPHOLs 2009. LNCS, vol. 5674, pp. 115–130. Springer, Heidelberg (2009)
3. Capretta, V.: General recursion via coinductive types. Log. Methods Comput. Sci. **1**(2:1), 1–28 (2005)
4. Chicli, L., Pottier, L., Simpson, D.: Mathematical quotients and quotient types in Coq. In: Geuvers, H., Wiedijk, F. (eds.) TYPES 2002. LNCS, vol. 2646, pp. 95–107. Springer, Heidelberg (2003)
5. Cockett, R., Díaz-Boïls, J., Gallagher, J., Hrubes, P.: Timed sets, complexity, and computability. In: Berger, U., Mislove, M. (eds.) Proceedings of 28th Conference on the Mathematical Foundations of Program Semantics, MFPS XXVIII. Electron. Notes in Theor. Comput. Sci., vol. 286, pp. 117–137. Elsevier, Amsterdam (2012)
6. Hofmann, M.: Extensional Constructs in Intensional Type Theory. CPHS/BCS Distinguished Dissertations. Springer, London (1997)
7. Hughes, J.: Generalising monads to arrows. Sci. Comput. Program. **37**(1–3), 67–111 (2000)

8. Jacobs, B., Heunen, C., Hasuo, I.: Categorical semantics for arrows. J. Funct. Program. **19**(3–4), 403–438 (2009)
9. Kraus, N., Escardó, M., Coquand, T., Altenkirch, T.: Notions of anonymous existence in Martin-Löf type theory. Manuscript (2014)
10. Maietti, M.E.: About effective quotients in constructive type theory. In: Altenkirch, T., Naraschewski, W., Reus, B. (eds.) TYPES 1998. LNCS, vol. 1657, pp. 166–178. Springer, Heidelberg (1999)
11. Martin-Löf, P.: 100 years of Zermelo's axiom of choice: what was the problem with it? Comput. J. **49**(3), 345–350 (2006)
12. Moggi, E.: Notions of computation and monads. Inf. Comput. **93**(1), 55–92 (1991)
13. Norell, U.: Dependently typed programming in Agda. In: Koopman, P., Plasmeijer, R., Swierstra, D. (eds.) AFP 2008. LNCS, vol. 5832, pp. 230–266. Springer, Heidelberg (2009)
14. Troelstra, A.S., Van Dalen, D.: Constructivism in Mathematics: An Introduction, v. I. Studies in Logic and the Foundations of Mathematics, vol. 121. North-Holland, Amsterdam (1988)
15. The Univalent Foundations Program: Homotopy Type Theory: Univalent Foundations of Mathematics. Institute for Advanced Study, Princeton, NY (2013). http://homotopytypetheory.org/book