Axiomatic Criteria for Quotients and Subobjects for Higher-Order Data Types

Jo Hannay

Department of Software Engineering,
Simula Research Laboratory, Pb. 134, NO-1325 Lysaker, Norway
johannay@simula.no

Abstract. Axiomatic criteria are given for the existence of higher-order maps over subobjects and quotients. These criteria are applied in showing the soundness of a method for proving specification refinement up to observational equivalence. This generalises the method to handle data types with higher-order operations, using standard simulation relations. We also give a direct setoid-based model satisfying the criteria. The setting is the second-order polymorphic lambda calculus and the assumption of relational parametricity.

1 Introduction

As a motivating framework for the results in this paper, we use specification refinement. We address specifications for data types whose operations may be higher order.

A *stepwise specification refinement* process transforms an abstract specification into one or more concrete specifications or program modules. If each step is proven correct, the resulting modules will be correct according to the initial abstract specification. This then describes a software development technique for producing small-scale certified components. Theoretical aspects to this idea have been researched thoroughly in the field of algebraic specification, see *e.g.,* [31,6].

When data types have higher-order operations, taking functions as arguments, several things in the refinement methodology break down. Most well-known perhaps, is the lack of correspondence between observational equivalence and the existence of simulation relations for data types, together with the lack of composability. The view is that standard notions of simulation relation are not adequate, and several remedies have been proposed; *pre-logical relations* [18,17], *lax logical relations* [28,20], *L-relations* [19], and *abstraction barrier-observing* simulation relations [11,12,13]. The latter, developed for System F in a logic [27] asserting *relational parametricity* [30], are directly motivated by the information-hiding mechanism in data types. Relational parametricity is in this context the logical assertion of the Basic Lemma [25,18] for simulation relations.

In this paper, we address a further issue. A general proof strategy for proving specification refinement up to observational equivalence is formalised in [4,3]. For data types with first-order operations, the strategy is expressed in the setting of System F and relational parametricity by axiomatising the existence of subobjects and quotients [29,36,9,12]. The axioms are sound w.r.t. the parametric *per* model of [1] which is a model for the logic in [27]. At higher order, more work is required, because in order to validate the axioms, one has to find a model which has higher-order operations over subobjects and quotients. Our solution is the core technical issue of this paper. First, we use a setoid-based semantics based on work on the syntactic level in [16]. Then we present general axiomatic criteria for the existence of higher-order functions over subobjects and quotients, and the setoid model is then an instance of this general schema. We think the axiomatic criteria are of general interest outside refinement issues. The results also answer the speculation in [36] about the soundness of similar axioms postulating quotients and subobjects at higher order.

Since simulation relations express observational equivalence, they play an integral part in the above proof strategy. At higher order, it is still possible to use standard simulation relations,

because the strategy relies on establishing observational equivalence from the existence of simulation relations. In this paper, we exploit this fact and devise the axiomatic criteria for standard simulation relations. For the strategy to be complete however, one must utilise one of the above alternative notions of simulation relation, since there may not exist a standard simulation relation even in the presence of observational equivalence. To this end, abstraction barrier-observing (*abo*) simulation relations were used in [10,12], together with *abo*-relational parametricity, and a special *abo*-semantics. That approach does indeed yield higher-order operations over quotients and subobjects, but to devise general axiomatic criteria for the existence of higher-order functions over subobjects and quotients with alternative notions of simulation relations, is ongoing research.

2 Syntax

We review relevant formal aspects. For full accounts, see [2,25,8,27,1].

The second-order lambda-calculus F_2, or System F, has abstract syntax

$$\begin{array}{ll}\text{(types)} & T ::= X \mid (T \to T) \mid (\forall X.T) \\ \text{(terms)} & t \ ::= x \mid (\lambda x{:}T.t) \mid (tt) \mid (\Lambda X.t) \mid (tT)\end{array}$$

where X and x range over type and term variables respectively. This provides polymorphic functionals and encodings of self-iterating inductive types [5], *e.g.,* $\mathsf{Nat} \stackrel{def}{=} \forall X.X \to (X \to X) \to X$, with constructors, destructors and conditionals. Products $U_1 \times \cdots \times U_n$ encode as inductive types.

We use the logic for parametric polymorphism due to [27]; a second-order logic augmented with relation symbols, relation definition, and the axiomatic assertion of relational parametricity. See also [22,34]. Formulae now include relational statements as basic predicates and quantifiables,

$$\phi ::= (t =_A u) \mid t \ R \ u \mid \ \cdots \ \mid \forall R \subset A \times B \ . \ \phi \mid \exists R \subset A \times B \ . \ \phi$$

where R ranges over relation variables. Relation definition is accommodated by the syntax,

$$\Gamma \rhd (x{:}A, y{:}B) \ . \ \phi \quad \subset A \times B$$

where ϕ is a formula. For example $\mathsf{eq}_A \stackrel{def}{=} (x{:}A, y{:}A).(x =_A y)$.

We write $\alpha[\xi]$ to indicate possible occurrences of variable ξ in type, term or formula α, and write $\alpha[\beta]$ for the substitution $\alpha[\beta/\xi]$, following the appropriate rules regarding capture.

We get the *arrow-type relation* $\rho \to \rho' \subset (A \to A') \times (B \to B')$ from $\rho \subset A \times B$ and $\rho' \subset A' \times B'$ by

$$(\rho \to \rho') \stackrel{def}{=} (f{:}A \to A', g{:}B \to B') \ . \ (\forall x{:}A . \forall y{:}B \ . \ (x \rho y \ \Rightarrow \ (fx) \rho' (gy)))$$

The *universal-type relation* $\forall(Y, Z, R \subset Y \times Z)\rho[R] \quad \subset \quad (\forall Y.A[Y]) \times (\forall Z.B[Z])$ is defined from $\rho[R] \subset A[Y] \times B[Z]$, where Y, Z and $R \subset Y \times Z$ are free, by

$$\forall(Y, Z, R \subset Y \times Z)\rho[R] \stackrel{def}{=} (y{:}\forall Y.A[Y], z{:}\forall Z.B[Z]) \ . \ (\forall Y . \forall Z . \forall R \subset Y \times Z \ . \ ((yY)\rho[R](zZ)))$$

For n-ary $\boldsymbol{X}$, $\boldsymbol{A}$, $\boldsymbol{B}$, $\boldsymbol{\rho}$, where $\rho_i \subset A_i \times B_i$, we get $T[\boldsymbol{\rho}] \subset T[\boldsymbol{A}] \times T[\boldsymbol{B}]$, the *action of* $T[\boldsymbol{X}]$ *on* $\boldsymbol{\rho}$, by

$$\begin{array}{lll} T[\boldsymbol{X}] = X_i : & & T[\boldsymbol{\rho}] = \rho_i \\ T[\boldsymbol{X}] = T'[\boldsymbol{X}] \to T''[\boldsymbol{X}] : & & T[\boldsymbol{\rho}] = T'[\boldsymbol{\rho}] \to T''[\boldsymbol{\rho}] \\ T[\boldsymbol{X}] = \forall X'.T'[\boldsymbol{X}, X'] : & & T[\boldsymbol{\rho}] = \forall(Y, Z, R \subset Y \times Z)T'[\boldsymbol{\rho}, R] \end{array}$$

The proof system is intuitionistic natural deduction, augmented with inference rules for relation symbols in the obvious way. There are standard axioms for equational reasoning implying extensionality for arrow and universal types.

Parametric polymorphism prompts all instances of a polymorphic functional to exhibit a uniform behaviour [33,1,30]. We adopt *relational parametricity* [30,21]; a polymorphic functional instantiated at two related domains, should give related instances. This is asserted by the schema

$$\textsc{Param}: \ \forall \boldsymbol{Z}.\forall u{:}(\forall X.U[X,\boldsymbol{Z}]) \ . \ u \ (\forall X.U[X,\mathbf{eq}_{\boldsymbol{Z}}]) \ u$$

The logic with PARAM is sound; we have the parametric *per*-model of [1] and the syntactic models of [14]. Relational parametricity yields the fundamental *Identity Extension Lemma*:

$$\forall \boldsymbol{Z}.\forall u,v{:}T[\boldsymbol{Z}] \ . \ (u \ T[\mathbf{eq}_{\boldsymbol{Z}}] \ v \ \Leftrightarrow \ (u =_{T[\boldsymbol{Z}]} v))$$

Constructs such as products, sums, initial and final (co-)algebras are encodable in System F [5]. With PARAM, these become provably universal constructions.

3 Specification Refinement

A specification determines a collection of data types realising the specification. A signature provides the desired namespace, and a set of formulae give properties to be fullfilled. Depending on refinement stage, these range from abstract to concrete implementational. A data type consists of a data representation and operations. In the logic, these are respectively a type A, and a term $\mathfrak{a} : T[A]$, where $T[X]$ plays the role of a signature. For instance, using a labeled product notation, $T_{\mathsf{STACK}_{\mathsf{Nat}}}[X] \stackrel{def}{=} (\mathsf{empty}{:}X,\ \mathsf{push}{:}\mathsf{Nat}{\to}X{\to}X,\ \mathsf{pop}{:}X{\to}X,\ \mathsf{top}{:}X{\to}\mathsf{Nat})$. Each $f_i : T_i[X]$ is a *profile* of the signature. Abstract properties are *e.g.,* $\forall x : \mathsf{Nat}, s : X \ . \ \mathfrak{x}.\mathsf{pop}(\mathfrak{x}.\mathsf{push}\,x\ s) = s \wedge \mathfrak{x}.\mathsf{top}(\mathfrak{x}.\mathsf{push}\,x\ s) = x$. A data type realising this stack specification, consists *e.g.,* of inductive type $\mathsf{List}_{\mathsf{Nat}}$ and $\mathfrak{l}$, where $\mathfrak{l}.\mathsf{empty} = \mathsf{nil}$, $\mathfrak{l}.\mathsf{push} = \mathsf{cons}$, $\mathfrak{l}.\mathsf{pop} = \lambda l : \mathsf{List}_{\mathsf{Nat}}.(\mathsf{cond}\ \mathsf{List}_{\mathsf{Nat}}\,(\mathsf{isnil}\,l)\,\mathsf{nil}\,(\mathsf{cdr}\,l))$, and $\mathfrak{l}.\mathsf{top} = \lambda l : \mathsf{List}_{\mathsf{Nat}}.(\mathsf{cond}\ \mathsf{Nat}\,(\mathsf{isnil}\,l)\,0\,(\mathsf{car}\,l))$. For encapsulation, data types would be given as packages of existential type, but our technical results are on the component level, so we omit this.

To each refinement stage, a set *Obs* of *observable types* is associated, containing inductive types, and also parameters. Two data types are interchangeable if it makes no difference which one is used in an observable computation. For example, an observable computation on natural-number stacks could be $\Lambda X.\lambda\mathfrak{x}{:}T_{\mathsf{STACK}_{\mathsf{Nat}}}[X] \ . \ \mathfrak{x}.\mathsf{top}(\mathfrak{x}.\mathsf{push}\,n\ \mathfrak{x}.\mathsf{empty})$. Thus, for A, B, $\mathfrak{a}{:}T[A], \mathfrak{b}{:}T[B]$, *Obs*,

Observational Equivalence: $\bigwedge_{D\in Obs} \forall f{:}\forall X.(T[X]{\to}D) \ . \ (f\,A\,\mathfrak{a}) = (f\,B\,\mathfrak{b})$

Observational equivalence can be hard to prove. A more manageable criterion for interchangeability lies in the concept of *data refinement* [15,7] and the use of relations to show *representation independence* [23,32,30], leading to *logical relations* for lambda calculus [24,25,35,26]. In the relational logic of [27] one uses the action of types on relations to express the above ideas. Two data types are related by a simulation relation if there exists a relation R on their respective data representations that is preserved by their corresponding operations:

Existence of Simulation Relation: $\exists R{\subset}A{\times}B \ . \ \mathfrak{a}\,(T[R])\,\mathfrak{b}$

With relational parametricity we get a connection to observational equivalence.

Theorem 1. *The following is derivable in the logic using* PARAM.

$$\forall A,B.\forall \mathfrak{a}{:}T[A],\mathfrak{b}{:}T[B] \ . \ \exists R{\subset}A{\times}B \ . \ \mathfrak{a}(T[R])\mathfrak{b} \ \Rightarrow \ \bigwedge_{D\in Obs} \forall f{:}\forall X.(T[X]{\to}D) \ . \ (f\,A\,\mathfrak{a}) = (f\,B\,\mathfrak{b})$$

Proof: This follows from the PARAM-instance $\forall Y.\forall f{:}\forall X.(T[X]{\to}Y) \ . \ f(\forall X.T[X]{\to}\mathsf{eq}_Y)f$. □

Consider the assumption that $T[X]$ has only first-order function profiles:

FADT$_{Obs}$**:** Every profile $T_i[X] = T_{i1}[X] \to \cdots \to T_{n_i}[X] \to T_{c_i}[X]$ of $T[X]$ is first order, and such that $T_{c_i}[X]$ is either X or some $D \in Obs$.

Assuming ***FADT***$_{Obs}$ for $T[X]$, Theorem 1 becomes a two-way implication [11,12].

For data types with higher-order operations, we only have Theorem 1 in general. More apt relational notions for explaining interchangeability of data types have been found; prelogical relations [18,17], lax logical relations and L-relations [28,20,19], and *abo*-simulation relations [11,12,13].

For specification refinement one is interested in establishing observational equivalence. For this it suffices to find a simulation relation and then use Theorem 1. The problem at higher order is that there might not exist a simulation relation, even in the presence of observational equivalence.

Nonetheless, it is in many cases possible to find simulation relations at higher order. It is worthwhile to utilise this, since it is harder to deal with the alternative notions in practice; prelogical relations involve an infinite family of relations, *abo*-relations involve definability. Therefore, this paper establishes a proof strategy for refinement at higher order using standard simulation relations.

The strategy for proving observational refinement formalised by Bidoit *et al* [4,3], expresses observational abstraction in terms of a congruence. Using this congruence, one quotients over the data representation. Additionally, it may be necessary to restrict the data representation before quotienting, and in that case one also needs to construct subobjects. For example, sets might be implemented using lists for data representation, but the operations may be optimised, and otherwise fail, by assuming sorted lists. Since lists represent the same set up to duplication of elements, the list algebra is quotiented by a partial congruence that equates lists modulo duplicates, and which is defined only on sorted lists. This strategy is implemented in the type-theoretical setting by extending the logic with the following axiom schemata. They are tailored specifically for refinement.

Definition 1 (Existence of Subobjects (SUB) [9]**).**

$$\begin{aligned}\textsc{Sub} : \forall X\,.\,\forall \mathfrak{x}{:}T[X]\,.\,\forall R \subset X\times X\;.\;(\mathfrak{x}\ T[R]\ \mathfrak{x})\ \wedge\ (\mathfrak{x}\ T[P_R]\ \mathfrak{x})\ \Rightarrow\\ \exists S\,.\,\exists \mathfrak{s}{:}T[S]\,.\,\exists R' \subset S\times S\,.\,\exists \mathsf{mono}{:}S\to X\;.\\ \forall s{:}S\;.\;s\ R'\ s \qquad\qquad \wedge\\ \forall s,s'{:}S\;.\;s\ R'\ s'\ \Leftrightarrow\ (\mathsf{mono}\,s)\ R\ (\mathsf{mono}\,s')\ \wedge\\ \mathfrak{x}\ (T[(x{:}X, s{:}S)\;.\;(x =_X (\mathsf{mono}\,s))])\ \mathfrak{s}\end{aligned}$$

where $P_R \stackrel{def}{=} (x{:}X, y{:}X)\;.\;(x =_X y \wedge x\ R\ x)$. Intuitively, this essentially states that for any data type $\langle X, \mathfrak{x}\rangle$, if R is a relation that is compatible with the signature $T[X]$, then there exists a data type $\langle S, \mathfrak{s}\rangle$, a relation R', and a monomorphism from $\langle S, \mathfrak{s}\rangle$ to $\langle X, \mathfrak{x}\rangle$, such that R' is total on $\langle S, \mathfrak{s}\rangle$ and a restriction of R via mono, and such that $\langle S, \mathfrak{s}\rangle$ is a subalgebra of $\langle X, \mathfrak{x}\rangle$.

Definition 2 (Existence of Quotients (QUOT) [29]**).**

$$\begin{aligned}\textsc{Quot} : \forall X\,.\,\forall \mathfrak{x}{:}T[X]\,.\,\forall R \subset X\times X\;.\quad (\mathfrak{x}\ T[R]\ \mathfrak{x} \wedge \mathit{equiv}(R))\ \Rightarrow\\ \exists Q\,.\,\exists \mathfrak{q}{:}T[Q]\,.\,\exists \mathsf{epi}{:}X\to Q\;.\\ \forall x,y{:}X\;.\;xRy\ \Leftrightarrow\ (\mathsf{epi}\,x) =_Q (\mathsf{epi}\,y)\ \wedge\\ \forall q{:}Q.\exists x{:}X\;.\;q =_Q (\mathsf{epi}\,x) \qquad\qquad \wedge\\ \mathfrak{x}\ (T[(x{:}X, q{:}Q).((\mathsf{epi}\,x) =_Q q)])\ \mathfrak{q}\end{aligned}$$

where $\mathit{equiv}(R)$ *specifies* R *to be an equivalence relation.*

Intuitively, this states that for any data type $\langle X, \mathfrak{x}\rangle$, if R is an equivalence relation on $\langle X, \mathfrak{x}\rangle$, then there exists a data type $\langle Q, \mathfrak{q}\rangle$ and an epimorphism from $\langle X, \mathfrak{x}\rangle$ to $\langle Q, \mathfrak{q}\rangle$, such that $\langle Q, \mathfrak{q}\rangle$ is a quotient algebra of $\langle X, \mathfrak{x}\rangle$.

Theorem 2. SUB, QUOT *hold in the parametric per-model of* [1]*, under* ***FADT***$_{Obs}$.

The proof of this theorem [12] relies on the model's ability to provide subobjects and quotients, and maps over these for any given morphism.

4 Higher-Order Quotient and Subobject Maps

In the *per*-model, first-order maps over subobjects and quotients are constructed from a given map by reusing the realiser. This does not work at higher order, since for functional arguments we have to contravariantly do this in reverse.

Consider *e.g.,* sequences over $\mathbb{N}$, whose encodings in $\mathbb{N}$ we write as the sequences themselves. Consider a function *rfi* on $\mathbb{N}$ that given a sequence, returns the sequence with the first item repeated. Define the *per*s $\mathcal{L}ist$, $\mathcal{B}ag$, and $\mathcal{S}et$ by

$$\begin{aligned} n\ \mathcal{L}ist\ m &\ \Leftrightarrow\ n \text{ and } m \textit{ encode the same list} \\ n\ \mathcal{B}ag\ m &\ \Leftrightarrow\ n \text{ and } m \textit{ encode the same list, modulo permutation} \\ n\ \mathcal{S}et\ m &\ \Leftrightarrow\ n \text{ and } m \textit{ encode the same list, modulo permutation and repetition} \end{aligned}$$

Here, *rfi* is a realiser for a map $f_{rfi} : \mathcal{S}et \to \mathcal{S}et$, but is not a realiser for any map in $\mathcal{B}ag \to \mathcal{B}ag$, *i.e.,* we have *rfi* $(\mathcal{S}et \to \mathcal{S}et)$ *rfi* but not *rfi* $(\mathcal{B}ag \to \mathcal{B}ag)$ *rfi*.

In fact, the general problem is that there may not be a suitable function at all, let alone one sharing the same realiser.

In the following we sketch a setoid model based on ideas in [16]. This allows the construction of subobject and quotient maps by reusing realisers, also at higher order. Then we give axiomatic criteria for the construction of subobject and quotient maps at higher order. The setoid model fulfils these criteria.

We will work under the following reasonable assumption.

$\mathtt{HADT}_{Obs}$: Every profile $T_i[X] = T_{i1}[X] \to \cdots \to T_{n_i}[X] \to T_{c_i}[X]$ of signature $T[X]$ is such that $T_{ij}[X]$ has no occurrences of universal types other than those in Obs, and $T_{c_i}[X]$ is either X or some $D \in Obs$.

4.1 A Setoid Model

Types are now interpreted as setoids, *i.e.,* pairs $\langle \mathcal{A}, \sim_{\mathcal{A}} \rangle$, consisting of a *per* $\mathcal{A}$ and a *per* $\sim_{\mathcal{A}}$ on $\mathcal{A}$, *i.e.,* a saturated *per* on $Dom(\mathcal{A}) \times Dom(\mathcal{A})$, giving the desired equality on the interpreted type. Given setoids $\langle \mathcal{A}, \sim_{\mathcal{A}} \rangle$ and $\langle \mathcal{B}, \sim_{\mathcal{B}} \rangle$, we form a setoid $\langle \mathcal{A}, \sim_{\mathcal{A}} \rangle \to \langle \mathcal{B}, \sim_{\mathcal{B}} \rangle \stackrel{def}{=} \langle \mathcal{A} \to \mathcal{B}, \sim_{\mathcal{A} \to \mathcal{B}} \rangle$, where $\sim_{\mathcal{A} \to \mathcal{B}}$ is the saturated relation $\sim_{\mathcal{A}} \to \sim_{\mathcal{B}} \subseteq Dom(\mathcal{A} \to \mathcal{B}) \times Dom(\mathcal{A} \to \mathcal{B})$. Saturation of $\sim$ is the condition $(m\ \mathcal{A}\ n\ \wedge\ n\ \sim\ n'\ \wedge\ n'\ \mathcal{B}\ m') \Rightarrow m \sim m'$.

A relation $\mathcal{R}$ between setoids $\langle \mathcal{A}, \sim_{\mathcal{A}} \rangle$ and $\langle \mathcal{B}, \sim_{\mathcal{B}} \rangle$ is now given by a saturated relation on $Dom(\sim_{\mathcal{A}}) \times Dom(\sim_{\mathcal{B}})$. Complex relations are defined as one would expect. The setoid definition of subobjects and quotients go as follows.

Definition 3 (Subobject Setoid). *Let $\mathcal{P}$ be a predicate on setoid $\langle \mathcal{X}, \sim_{\mathcal{X}} \rangle$, meaning that $\mathcal{P}$ fulfils the unary saturation condition $\mathcal{P}(x) \wedge x \sim_{\mathcal{X}} y \Rightarrow \mathcal{P}(y)$. Define the relation, also denoted $\mathcal{P}$, on $\langle \mathcal{X}, \sim_{\mathcal{X}} \rangle$ by $x\ \mathcal{P}\ y \stackrel{def}{\Leftrightarrow} x \sim_X y \wedge \mathcal{P}(x)$. Then the* subobject $\mathsf{R}_{\mathcal{P}}(\langle \mathcal{X}, \sim_{\mathcal{X}} \rangle)$ *of* $\langle \mathcal{X}, \sim_{\mathcal{X}} \rangle$ restricted on $\mathcal{P}$, *is defined by* $\langle \mathcal{X}, \mathcal{P} \rangle$.

Definition 4 (Quotient Setoid). *Let $\mathcal{R}$ be a equivalence relation on setoid $\langle \mathcal{X}, \sim_{\mathcal{X}} \rangle$. Define the* quotient $\langle \mathcal{X}, \sim_{\mathcal{X}} \rangle / \mathcal{R}$ *of* $\langle \mathcal{X}, \sim_{\mathcal{X}} \rangle$ w.r.t. $\mathcal{R}$ *by* $\langle \mathcal{X}, \mathcal{R} \rangle$.

Theorem 3. *Suppose $T[X]$ adheres to $\mathtt{HADT}_{Obs}$. Then* Sub *and* Quot *hold in the setoid model indicated above.*

With setoids we may construct quotient maps from a given map and *vice versa* by reusing realisers, since the original domain inhabitation is preserved by subobjects and quotients. However, Theorem 3 is given as a corollary to a general result of the axiomatic criteria in the next section.

4.2 Axiomatic Criteria for Subobject and Quotient Maps

We now develop a general axiomatic scheme for obtaining subobject and quotient maps. The setoid approach in the previous section is an instance of this scheme.

For quotients, the general problem is that for a given map $f : X/R \to X/R$, there need not exist a map $g : X \to X$ such that for all $x : X$, $[g(x)] = f([x])$, *i.e.,* $epi(g(x)) = f(epi(x))$, where $epi : X \to X/R$ maps an element to its equivalence class. This is the case for the *per*-model. The axiom of choice (AC) gives such a map g, because then epi has an inverse, and the desired g is given by $\lambda x{:}X.epi^{-1}(f\,epi(x))$. AC does not hold in the *per*-model, nor does it hold in the setoid model of the previous section. In this section, we develop both a weaker condition sufficient to give higher-order quotient maps, and a condition for obtaining higher-order subobject maps.

According to HADT_{Obs}, we consider arrow types over types $U_0, U_1, \ldots$, where any U_i is either X or some $D \in Obs$. For this, define families U^i by

$$\begin{array}{ll} U^0 & = U_0 \\ U^{i+1} & = (U^i) \to U_{i+1} \end{array}$$

For example, $U^2 = ((U_0 \to U_1) \to U_2)$.

Quotient Maps For $U = U^n$, define $Q(U)^i$ for any equivalence relation R,

$$\begin{array}{ll} Q(U)^0 & = U_0 \\ Q(U)^1 & = U_0/R \to U_1 \\ Q(U)^{i+1} & = (Q(U)^{i-1} \to U_i/R) \to U_{i+1},\ 1 \le i \le n-1 \end{array}$$

where, $U_i/R = X/R$, if U_i is X, and $U_i/R = D$, if U_i is $D \in Obs$, *e.g.,* $Q(U^2)^2 = ((U_0 \to U_1/R) \to U_2)$. In any $Q(U)^i$, quotients U_j/R occur only negatively.

Given $Q(U)^n$, we get derived relations, functions and types by the substitution operators $Q(U)^n[\xi]^+$ and $Q(U)^n[\xi]^-$, according to ξ being a relation, function or type; $Q(U)^n[\xi]^+$ substitutes ξ for positive occurrences of X in $Q(U)^n$, and $Q(U)^n[\xi]^-$ substitutes ξ for every (negative) occurrence of X/R in $Q(U)^n$. Relational and functional identities are then denoted by their domains.

Thus for $U = U^n$ and the equivalence relation R, we can define the relation

$$R(U)^n \stackrel{def}{=} Q(U)^n[R]^+$$

In any $R(U)^i$, R occurs positively, and identities U_j/R occur only negatively. The point of all this is that, if R is an equivalence relation on X, then $R(U)^i$ is an equivalence relation on $Q(U)^i$. This means that we may form the quotient $Q(U)^i/R(U)^i$. For example, consider $U = U^1 = X \to X$. Then $Q(U)^1 = X/R \to X$ and $R(U)^1 = X/R \to R$, and $X/R \to R$ is an equivalence relation on $X/R \to X$. In contrast, $R \to X/R$ is not necessarily an equivalence relation on $X \to X/R$. However, $(R \to X/R) \to R$ is an equivalence relation on $(X \to X/R) \to X$, that is, $R(U^2)^2$ is an equivalence relation on $Q(U^2)^2$, for $U^2 = (X \to X) \to X$.

Now consider the relation $graph(\mathsf{epi}) \stackrel{def}{=} (x : X, q : X/R)\ .\ ((\mathsf{epi}\, x) =_{X/R} q)$ where the map $\mathsf{epi} : X \to X/R$ maps elements to their R-equivalence class. A sufficient condition for obtaining higher-order functions over quotients is now

***Quot-Arr*:** For R an equivalence relation on X, and any given $U = U^n$,

$$Q(U)^n/R(U)^n \;\cong\; Q(U)^n[X/R]^+$$

where the isomorphism $\mathsf{iso} : Q(U)^n[X/R]^+ \to Q(U)^n/R(U)^n$ is such that any f in the equivalence class $\mathsf{iso}(\beta)$ is such that $f\ (Q(U)^n[graph(\mathsf{epi})]^+)\ \beta$.

Note that ***Quot-Arr*** is not an extension to our logic; we do not have quotient types. Rather, ***Quot-Arr*** is a condition to check in any relevant model, in which the terminology concerning quotients is well defined. In [16], ***Quot-Arr*** is expressible in the logic, and ***Quot-Arr*** is shown strictly weaker than the axiom of choice.

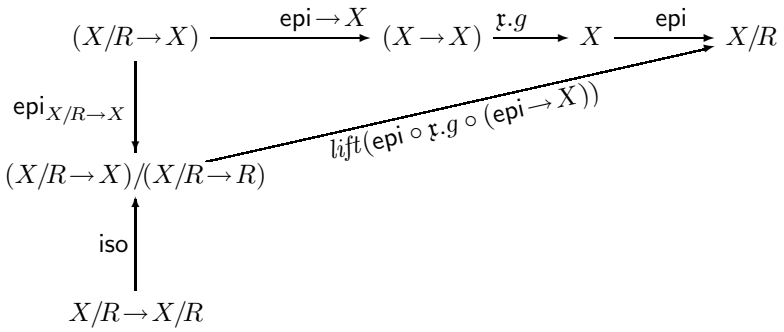
Let us exemplify why ***Quot-Arr*** suffices. The challenge of this paper is higher-order operations in data types, and then the soundness of QUOT and SUB where $T[X]$ has higher-order operation profiles. To illustrate the use of ***Quot-Arr*** in semantically validating QUOT, suppose $T[X]$ has a profile $g : (X \to X) \to X$ and that $R \subset X \times X$ is an equivalence relation. Consider now any $\mathfrak{x} : T[X]$. Assuming $\mathfrak{x}\ (T[R])\ \mathfrak{x}$, we must produce a $\mathfrak{q} : T[X/R]$, such that $\mathfrak{x}\ (T[\mathit{graph}(\mathsf{epi})])\ \mathfrak{q}$. For $\mathfrak{x}.g : (X \to X) \to X$, this involves finding a $\mathfrak{q}.g : (X/R \to X/R) \to X/R$, such that

$$\mathfrak{x}.g\ ((\mathit{graph}(\mathsf{epi}) \to \mathit{graph}(\mathsf{epi})) \to \mathit{graph}(\mathsf{epi}))\ \mathfrak{q}.g \tag{1}$$

Consider now the following instance of ***Quot-Arr***.

Quot-Arr$_1$**:** $(X/R \to X)/(X/R \to R) \ \cong\ (X/R \to X/R)$

With ***Quot-Arr***$_1$ we can construct the following commuting diagram.



where $\mathsf{epi} \to X$ maps any $f : X/R \to X$ to $\lambda x : X.f(\mathsf{epi}\, x)$, and iso is so that any f in the equivalence class $\mathsf{iso}(\beta)$ satisfies $f\ (\mathsf{eq}_{X/R} \to \mathit{graph}(\mathsf{epi}))\ \beta$. The desired $\mathfrak{q}.g : (X/R \to X/R) \to X/R$ is given by

$$\mathit{lift}(\mathsf{epi} \circ \mathfrak{x}.g \circ (\mathsf{epi} \to X)) \circ \mathsf{iso}$$

Here *lift* is the operation that lifts any $\gamma : Z \to Y$ to $\mathit{lift}(\gamma) : Z/\!\sim \to Y$, given an equivalence relation $\sim$ on Z, provided that γ satisfies $x \sim y \ \Rightarrow\ \gamma x = \gamma y$ for all $x, y : Z$. Then, $\mathit{lift}(\gamma)$ is the map satisfying $\mathit{lift}(\gamma) \circ \mathsf{epi} = \gamma$. To be able to lift $\mathsf{epi} \circ \mathfrak{x}.g \circ (\mathsf{epi} \to X)$ in this way, we must check that $\mathsf{epi} \circ \mathfrak{x}.g \circ (\mathsf{epi} \to X)$ satisfies $f\ (\mathsf{eq}_{X/R} \to R)\ f' \ \Rightarrow\ (\mathsf{epi} \circ \mathfrak{x}.g \circ (\mathsf{epi} \to X))(f) =_{X/R} (\mathsf{epi} \circ \mathfrak{x}.g \circ (\mathsf{epi} \to X))(f')$, for all $f, f' : (X/R \to X)$. Assuming $f\ (\mathsf{eq}_{X/R} \to R)\ f'$, we get $(\mathsf{epi} \to X)(f)\ (R \to R)\ (\mathsf{epi} \to X)(f')$. Then by $\mathfrak{x}\ T[R]\ \mathfrak{x}$, the result follows. This warrants the construction of $\mathfrak{q}.g$.

To show that $\mathfrak{q}.g$ is the desired function, we must check that it satisfies (1). This cannot be read directly from the above diagram; for instance, although $\mathfrak{q}.g$ is constructed essentially in terms of $\mathfrak{x}.g$, it is clear that $\mathsf{epi} \to X$ maps only to those α in $X \to X$ that do not discern between input of the same R-equivalence class, and these α might not cover the domain of inputs giving all possible outputs. Intuitively though, this suffices since R-equivalence is really all that matters.

More formally, suppose $\alpha : X \to X$ and $\beta : X/R \to X/R$ are such that

$$\alpha\ \ (\mathit{graph}(\mathsf{epi}) \to \mathit{graph}(\mathsf{epi}))\ \ \beta \tag{2}$$

We want $(\mathfrak{x}.g\,\alpha)\ \mathit{graph}(\mathsf{epi})\ (\mathfrak{q}.g\,\beta)$. First show for any $\alpha : X \to X$, there exists $f_\alpha : X/R \to X$, s.t. $(\mathsf{epi} \to X) f_\alpha\ \ (R \to R)\ \ \alpha$ and $\mathsf{iso}(\beta) = \mathsf{epi}_{X/R \to X}(f_\alpha)$, *i.e.,*

$$\lambda x : X.f_\alpha(\mathsf{epi}\, x))\ (R \to R)\ \alpha \tag{3}$$

$$\mathsf{iso}(\beta) = [f_\alpha]_{X/R \to R} \tag{4}$$

The assumption on iso in *Quot-Arr*$_1$ is that any f in the equivalence class $\mathsf{iso}(\beta)$ is such that

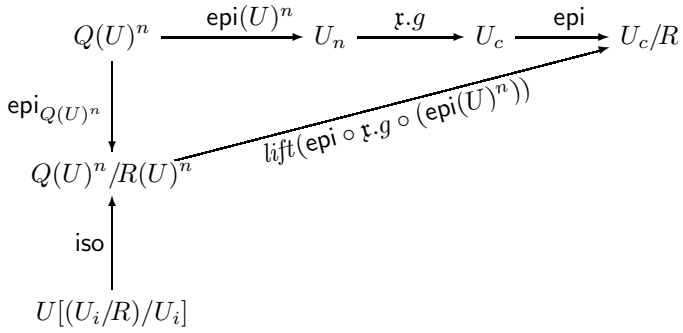
$$f \;\; (\mathsf{eq}_{X/R} \to \mathit{graph}(\mathsf{epi})) \;\; \beta \tag{5}$$

so any of these f are candidates for f_α. For such an f we show $a\ R\ a' \;\Rightarrow\; (\lambda x{:}X.f(\mathsf{epi}\,x)a)\ R\ \alpha a'$, *i.e.*, $[a] = [a'] \;\Rightarrow\; [f[a]] = [\alpha a']$. We have from (5), $[a] = [a'] \;\Rightarrow\; [f[a]] = \beta[a']$, and by (2), we have $[a] = [a'] \;\Rightarrow\; [\alpha a] = \beta[a']$. Together, this gives the desired property for f, so we have the existence of f_α satisfying (3) and (4). From (2) and (5) we also get

$$\lambda x{:}X.f_\alpha(\mathsf{epi}\,x) \;\; (\mathit{graph}(\mathsf{epi}) \to \mathit{graph}(\mathsf{epi})) \;\; \beta$$

From the above diagram, and (3) and (4), this gives $(\mathfrak{x}.g\,(\lambda x{:}X.f_\alpha(\mathsf{epi}\,x)))\ \ \mathit{graph}(\mathsf{epi})\ \ (\mathfrak{q}.g\,\beta)$. By $\mathfrak{x}\ T[R]\ \mathfrak{x}$, and since we have $\alpha\ \ (R \to R)\ \ (\lambda x{:}X.f_\alpha(\mathsf{epi}\,x))$, we thus get $(\mathfrak{x}.g\,\alpha)\ \mathit{graph}(\mathsf{epi})\ (\mathfrak{q}.g\,\beta)$.

The general form of this diagram for any given $U = U^n$ and U_c, is



where for a given $U = U^n$, we define the function $\mathsf{epi}(U)^n \stackrel{def}{=} Q(U)^n[\mathsf{epi}]^-$.

Subobject Maps A similar story applies to subobjects. For any predicate P on X, we write $\mathsf{R}_P(X)$ for the subobject of X classified by those $x{:}X$ such that $P(x)$ holds. Let the monomorphism $\mathsf{mono}{:}\mathsf{R}_P(X) \to X$ map elements to their correspondents in X. For use in arrow-type relations, we construct a binary relation from P, also denoted P, by

$$P \stackrel{def}{=} (x{:}X, y{:}X)\ .\ (x =_X y\ \wedge\ \exists y'{:}\mathsf{R}_P(X)\ .\ y = (\mathsf{mono}\,y'))$$

Now for a given $U = U^n$, define $S(U)^i$ for some P as follows

$$\begin{array}{ll} S(U)^0 & = \mathsf{R}_P(U_0) \\ S(U)^1 & = U_0 \to \mathsf{R}_P(U_1) \\ S(U)^{i+1} & = (S(U)^{i-1} \to U_i) \to \mathsf{R}_P(U_{i+1}),\ 1 \le i \le n-1 \end{array}$$

where, if U_i is X then $\mathsf{R}_P(U_i) = \mathsf{R}_P(X)$, and if U_i is $D \in \mathit{Obs}$ then $\mathsf{R}_P(U_i) = D$. For example $S(U^2)^2 = ((\mathsf{R}_P(U_0) \to U_1) \to \mathsf{R}_P(U_2))$. In any $S(U)^i$, subobjects $\mathsf{R}_P(U_j)$ occur only positively. For any $U = U^n$ and some P, define the relation

$$P(U)^n \stackrel{def}{=} S(U)^n[P]^-$$

The substitution operators $S(U)^n[\xi]^-$ and $S(U)^n[\xi]^+$ are analogues to $Q(U)^n[\xi]^-$ and $Q(U)^n[\xi]^+$. Identities are denoted by their domains.

Intuitively, one would think that for any given $U = U^n$, we should now postulate an isomorphism between $\mathsf{R}_{P(U)^n}(S(U)^n)$ and $S(U)^n[(\mathsf{R}_P(X))]^-$. This would be in dual analogy to *Quot-Arr*.

However, this isomorphism does not exist even in the setoid model. For example, we will not be able to find an isomorphism between $\mathsf{R}_{P\to\mathsf{R}_P(X)}(X\to\mathsf{R}_P(X))$ and $\mathsf{R}_P(X)\to\mathsf{R}_P(X)$. However, it turns out that we can in fact use an outermost quotient instead of subobjects for the isomorphism, in the same way as we did for ***Quot-Arr***.

Thus, if P is a predicate on X, then $P(U)^i$ is an equivalence relation on $S(U)^i$. This means that we may form the quotient $S(U)^i/P(U)^i$, *e.g.,* if $U = U^1 = X\to X$, then $S(U)^1 = X\to\mathsf{R}_R(X)$ and $P(U)^1 = P\to\mathsf{R}_P(X)$, and $P\to\mathsf{R}_P(X)$ is an equivalence relation on $X\to\mathsf{R}_R(X)$. Again, in contrast, $\mathsf{R}_P(X)\to P$ is not necessarily an equivalence relation on $\mathsf{R}_P(X)\to X$. However, $(\mathsf{R}_P(X)\to P)\to\mathsf{R}_P(X)$ is an equivalence relation on $(\mathsf{R}_P(X)\to X)\to\mathsf{R}_P(X)$, that is, $P(U^2)^2$ is an equivalence relation on $S(U^2)^2$, for $U^2 = (X\to X)\to X$.

For the relation $\mathit{graph}(\mathsf{mono}) \stackrel{\mathit{def}}{=} (x:X, s:\mathsf{R}_P(X))\ .\ (x =_X (\mathsf{mono}\, s))$, a sufficient condition for obtaining higher-order functions over subobjects is,

***Sub-Arr*:** For P a predicate on X, and any given $U = U^n$,

$$S(U)^n/P(U)^n \ \cong\ S(U)^n[(\mathsf{R}_P(X))]^-$$

where the isomorphism $\mathsf{iso} : S(U)^n[(\mathsf{R}_P(X))]^- \to S(U)^n/P(U)^n$ is such that any f in the equivalence class $\mathsf{iso}(\beta)$ is such that $f\ (S(U)^n[\mathit{graph}(\mathsf{mono})]^-)\ \beta$.

Again, ***Sub-Arr*** is not an axiom in our logic, but is a condition that we can check for models in which the terminology in ***Sub-Arr*** has a well-defined meaning.

To illustrate ***Sub-Arr***, suppose $T[X]$ has a profile $g:(X\to X)\to X$. For any $\mathfrak{x}:T[X]$, assume $\mathfrak{x}\ T[P]\ \mathfrak{x}$. We must exhibit a $\mathfrak{s}:T[\mathsf{R}_P(X)]$, such that $\mathfrak{x}\ T[\mathit{graph}(\mathsf{mono})]\ \mathfrak{s}$. For $\mathfrak{x}.g:(X\to X)\to X$, this means finding a $\mathfrak{s}.g:(\mathsf{R}_P(X)\to\mathsf{R}_P(X))\to\mathsf{R}_P(X)$, s.t.

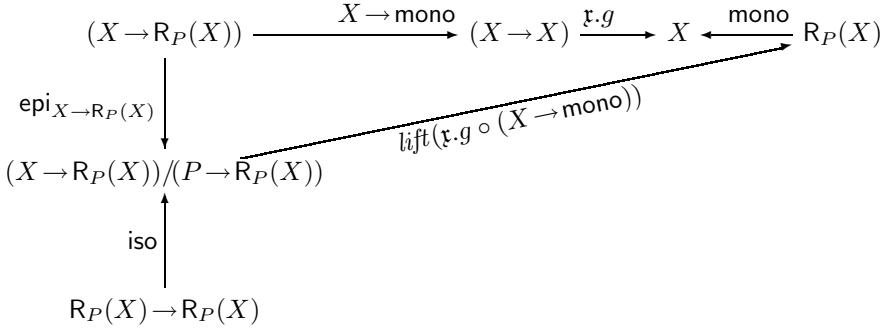
$$\mathfrak{x}.g\ ((\mathit{graph}(\mathsf{mono})\to\mathit{graph}(\mathsf{mono}))\to\mathit{graph}(\mathsf{mono}))\ \mathfrak{s}.g \tag{6}$$

Consider now the following instance of ***Sub-Arr***.

***Sub-Arr*$_1$:** For a predicate P on X,

$$(X\to\mathsf{R}_P(X))/(P\to\mathsf{R}_P(X))\ \cong\ \mathsf{R}_P(X)\to\mathsf{R}_P(X)$$

Using ***Sub-Arr*$_1$**, we can construct the following commuting diagram.



Then, $\mathfrak{s}.g:(\mathsf{R}_P(X)\to\mathsf{R}_P(X))\to\mathsf{R}_P(X)$ is given by $\mathit{lift}(\mathfrak{x}.g\circ(X\to\mathsf{mono}))\circ\mathsf{iso}$. To justify the lifting of $\mathfrak{x}.g\circ(X\to\mathsf{mono})$, we must show for all $f, f':X\to\mathsf{R}_P(X)$ satisfying $f\ (P\to\mathsf{R}_P(X))\ f'$, that $\mathfrak{x}.g\circ(X\to\mathsf{mono})(f) =_X \mathfrak{x}.g\circ(X\to\mathsf{mono})(f')$. Note that $\mathit{lift}(\mathfrak{x}.g\circ(X\to\mathsf{mono}))$ then maps to X, so in addition we must show that $\mathit{lift}(\mathfrak{x}.g\circ(X\to\mathsf{mono}))$ in fact maps to $\mathsf{R}_P(X)$. Now, if $f\ (P\to\mathsf{R}_P(X))\ f'$, we get $(X\to\mathsf{mono})(f)\ (P\to P)\ (X\to\mathsf{mono})(f')$. By assumption, we have $\mathfrak{x}\ T[P]\ \mathfrak{x}$, in particular $\mathfrak{x}.g\ ((P\to P)\to P)\ \mathfrak{x}.g$, and the result follows. If for some y, $\exists y':\mathsf{R}_P(X)\ .\ \mathsf{mono}\, y' = y$, we

assume that it is elementary to find such a y'. Thus, since mono is a monomorphism, we may map $\mathit{lift}(\mathfrak{x}.g \circ (X \to \mathsf{mono}))$ to $\mathsf{R}_P(X)$, and so we have a function $\mathfrak{s}.g : ((\mathsf{R}_P(X) \to \mathsf{R}_P(X)) \to \mathsf{R}_P(X))$.

To show that $\mathfrak{s}.g$ is the desired function, we must check that it satisfies (6). Suppose $\alpha : X \to X$ and $\beta : \mathsf{R}_P(X) \to \mathsf{R}_P(X)$ are such that

$$\alpha \quad (\mathit{graph}(\mathsf{mono}) \to \mathit{graph}(\mathsf{mono})) \quad \beta \tag{7}$$

We want $(\mathfrak{x}.g\,\alpha)$ $\mathit{graph}(\mathsf{mono})$ $(\mathfrak{s}.g\,\beta)$. First show for any $\alpha : X \to X$, there exists $f_\alpha : X \to \mathsf{R}_P(X)$, such that $(X \to \mathsf{mono}) f_\alpha \quad (P \to P) \quad \alpha$ and $\mathsf{iso}(\beta) = \mathsf{epi}_{X \to \mathsf{R}_P(X)}(f_\alpha)$, *i.e.,*

$$\lambda x : X.\mathsf{mono}(f_\alpha x)\ (P \to P)\ \alpha \tag{8}$$

$$\mathsf{iso}(\beta) = [f_\alpha]_{P \to \mathsf{R}_P(X)} \tag{9}$$

The assumption on iso in $\mathit{Sub\text{-}Arr}_1$ is that any f in the equivalence class $\mathsf{iso}(\beta)$ is such that

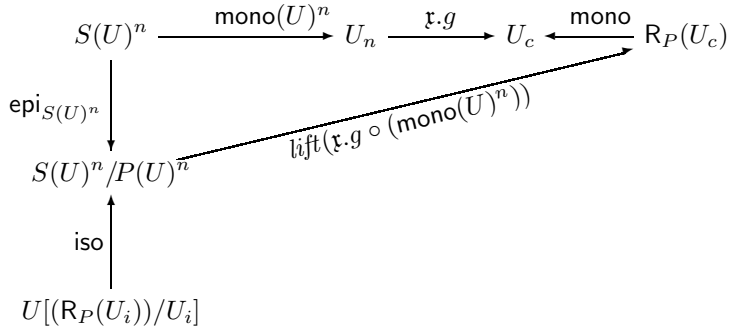
$$f \quad (\mathit{graph}(\mathsf{mono}) \to \mathsf{eq}_{\mathsf{R}_P(X)}) \quad \beta \tag{10}$$

so any of these f are candidates for f_α. For such an f, show (8), *i.e.,* $a = a' \wedge \exists a''.\mathsf{mono}\, a'' = a' \;\Rightarrow\; \mathsf{mono}(fa) = \alpha a' \wedge \exists b\ .\ \mathsf{mono}\, b = \alpha a'$. We have from (10), $a = \mathsf{mono}\ a'' \;\Rightarrow\; fa = \beta\ a''$, and by assumption on α and β, we have $a' = \mathsf{mono}\ a'' \;\Rightarrow\; \alpha a' = \mathsf{mono}(\beta a'')$. This gives the desired property for f, so we have the existence of f_α satisfying (8) and (9). From (10) we also get

$$\lambda x : X.\mathsf{mono}(f_\alpha x) \quad (\mathit{graph}(\mathsf{mono}) \to \mathit{graph}(\mathsf{mono})) \quad \beta$$

From the above diagram, and (8) and (9), this gives $(\mathfrak{x}.g\,(\lambda x : X.\mathsf{mono}(f_\alpha x)))$ $\mathit{graph}(\mathsf{mono})$ $(\mathfrak{s}.g\,\beta)$. By $\mathfrak{x}\ T[P]\ \mathfrak{x}$, and since $\alpha \quad (P \to P) \quad \lambda x : X.\mathsf{mono}(f_\alpha x)$, we thus get $(\mathfrak{x}.g\,\alpha) \quad \mathit{graph}(\mathsf{mono}) \quad (\mathfrak{s}.g\,\beta)$.

Here is the general form of this diagram for any given $U = U^n$ and U_c, is



where for a given $U = U^n$, we define the function $\mathsf{mono}(U)^n \stackrel{def}{=} S(U)^n[\mathsf{mono}]^+$.

This schema is more general than what is called for in the refinement-specific SUB. In SUB, the starting point is a relation R, and the predicate with which one restricts the domain X, is $P_R(x) \stackrel{def}{=} x\ R\ x$. The corresponding binary relation is then $P_R \stackrel{def}{=} (x : X, y : X)\ .\ (x =_X y \wedge x\ R\ x))$.

In closing, we mention that the *per* model, parametric or not, does not satisfy $\mathit{Quot\text{-}Arr}$ nor $\mathit{Sub\text{-}Arr}$. We summarise:

Theorem 4. *Suppose $T[X]$ adheres to HADT_{Obs}. Then* SUB *and* QUOT *hold in any model that satisfies $\mathit{Sub\text{-}Arr}$ and $\mathit{Quot\text{-}Arr}$.*

Theorem 5. *The setoid model satisfies $\mathit{Quot\text{-}Arr}$ and $\mathit{Sub\text{-}Arr}$, by the isomorphism being denotational equality.*

Proof: See [12]. □

Corollary 6. SUB *and* QUOT *hold in the setoid model indicated above.*

5 Final Remarks

We have devised and validated a method in logic for proving specification refinement for data types with higher-order operations. The method is based on standard simulation relations, accommodating the fact that these are easier to deal with than alternative notions when performing refinement. In general however, there may not exist standard simulation relations at higher order, even in the presence of observational equivalence. It is possible to devise specialised solutions to this using abstraction barrier-observing simulation relations [10,12], or pre-logical relations expressed in System F. Beyond that, it is desirable to find general axiomatic criteria analogous to *Sub-Arr* and *Quot-Arr*, using alternative notions of simulation relations. This is currently under investigation.

Acknowledgments Martin Hofmann has contributed with essential input.

References

1. E.S. Bainbridge, P.J. Freyd, A. Scedrov, and P.J. Scott. Functorial polymorphism. *Theoretical Computer Science*, 70:35–64, 1990.
2. H.P. Barendregt. Lambda calculi with types. In S. Abramsky, D.M. Gabbay, and T.S.E. Maibaum, editors, *Handbook of Logic in Computer Science*, volume 2, pages 118–309. Oxford University Press, 1992.
3. M. Bidoit and R. Hennicker. Behavioural theories and the proof of behavioural properties. *Theoretical Computer Science*, 165:3–55, 1996.
4. M. Bidoit, R. Hennicker, M. Wirsing. Proof systems for structured specifications with observability operators. *Theoretical Computer Science*, 173:393–443, 1997.
5. C. Böhm and A. Berarducci. Automatic synthesis of typed λ-programs on term algebras. *Theoretical Computer Science*, 39:135–154, 1985.
6. M. Cerioli, M. Gogolla, H. Kirchner, B. Krieg-Brückner, Z. Qian, and M. Wolf, *eds.. Algebraic System Specification and Development. Survey and Annotated Bibliography, 2nd Ed.*, *BISS Monographs*, vol. 3. Shaker Verlag, 1997.
7. O.-J. Dahl. *Verifiable Programming, Revised version 1993.* Prentice Hall Int. Series in Computer Science; C.A.R. Hoare, Series Editor. Prentice-Hall, UK, 1992.
8. J.-Y. Girard, P. Taylor, and Y. Lafont. *Proofs and Types.* Cambridge Tracts in Theoretical Computer Science. Cambridge University Press, 1990.
9. J. Hannay. Specification refinement with System F. In *Computer Science Logic. Proc. of CSL'99*, vol. 1683 of *Lecture Notes in Comp. Sci.*, pages 530–545. Springer Verlag, 1999.
10. J. Hannay. Specification refinement with System F, the higher-order case. In *Recent Trends in Algebraic Development Techniques. Selected Papers from WADT'99*, volume 1827 of *Lecture Notes in Comp. Sci.*, pages 162–181. Springer Verlag, 1999.
11. J. Hannay. A higher-order simulation relation for System F. In *Foundations of Software Science and Computation Structures. Proc. of FOSSACS 2000*, vol. 1784 of *Lecture Notes in Comp. Sci.*, pages 130–145. Springer Verlag, 2000.
12. J. Hannay. *Abstraction Barriers and Refinement in the Polymorphic Lambda Calculus.* PhD thesis, Laboratory for Foundations of Computer Science (LFCS), University of Edinburgh, 2001.
13. J. Hannay. Abstraction barrier-observing relational parametricity. In *Typed Lambda Calculi and Applications. Proc. of TLCA 2002*, *Lecture Notes in Comp. Sci.*, Springer Verlag, 2002. To appear.
14. R. Hasegawa. Parametricity of extensionally collapsed term models of polymorphism and their categorical properties. In *Theoretical Aspects of Computer Software. Proc. of TACS'91*, vol. 526 of *Lecture Notes in Comp. Sci.*, pages 495–512. Springer Verlag, 1991.
15. C.A.R. Hoare. Proofs of correctness of data representations. *Acta Informatica*, 1:271–281, 1972.
16. M. Hofmann. *Extensional Concepts in Intensional Type Theory, Report CST-117-95 and Technical Report ECS-LFCS-95-327.* PhD thesis, Laboratory for Foundations of Computer Science (LFCS), University of Edinburgh, 1995.

17. F. Honsell, J. Longley, D. Sannella, and A. Tarlecki. Constructive data refinement in typed lambda calculus. In *Foundations of Software Science and Computation Structures. Proc. of FOSSACS 2000*, vol. 1784 of *Lecture Notes in Comp. Sci.*, pages 161–176. Springer Verlag, 2000.
18. F. Honsell and D. Sannella. Prelogical relations. *Information and Computation*, 178:23–43, 2002.
19. Y. Kinoshita, P.W. O'Hearn, J. Power, M. Takeyama, and R.D. Tennent. An axiomatic approach to binary logical relations with applications to data refinement. In *Theoretical Aspects of Computer Software. Proc. of TACS'97*, vol. 1281 of *Lecture Notes in Comp. Sci.*, pages 191–212. Springer Verlag, 1997.
20. Y. Kinoshita and J. Power. Data refinement for call-by-value programming languages. In *Computer Science Logic. Proc. of CSL'99*, vol. 1683 of *Lecture Notes in Comp. Sci.*, pages 562–576. Springer Verlag, 1999.
21. Q. Ma and J.C. Reynolds. Types, abstraction and parametric polymorphism, part 2. In *Mathematical Foundations of Programming Semantics, Proc. of MFPS*, vol. 598 of *Lecture Notes in Comp. Sci.*, pages 1–40. Springer Verlag, 1991.
22. H. Mairson. Outline of a proof theory of parametricity. In *Functional Programming and Computer Architecture. Proc. of the 5th acm Conf.*, vol. 523 of *Lecture Notes in Comp. Sci.*, pages 313–327. Springer Verlag, 1991.
23. R. Milner. An algebraic definition of simulation between programs. In *Joint Conferences on Artificial Intelligence, Proceedings of the 2nd International Conference, JCAI, London (UK)*, pages 481–489. Morgan Kaufman Publishers, 1971.
24. J.C. Mitchell. On the equivalence of data representations. In V. Lifschitz, editor, *Artificial Intelligence and Mathematical Theory of Computation: Papers in Honor of John McCarthy*, pages 305–330. Academic Press, 1991.
25. J.C. Mitchell. *Foundations for Programming Languages*. Foundations of Computing. MIT Press, 1996.
26. P.W. O'Hearn and R.D. Tennent. Relational parametricity and local variables. In *20th SIGPLAN-SIGACT Symposium on Principles of Programming Languages, Proceedings*, pages 171–184. ACM Press, 1993.
27. G.D. Plotkin and M. Abadi. A logic for parametric polymorphism. In *Typed Lambda Calculi and Applications. Proc. of TLCA'93*, vol. 664 of *Lecture Notes in Comp. Sci.*, pages 361–375. Springer Verlag, 1993.
28. G.D. Plotkin, J. Power, D. Sannella, and R.D. Tennent. Lax logical relations. In *Automata, Languages and Programming. Proc. of ICALP 2000*, vol. 1853 of *Lecture Notes in Comp. Sci.*, pages 85–102. Springer Verlag, 2000.
29. E. Poll and J. Zwanenburg. A logic for abstract data types as existential types. In *Typed Lambda Calculus and Applications. Proc. of TLCA'99*, vol. 1581 of *Lecture Notes in Comp. Sci.*, pages 310–324. Springer Verlag, 1999.
30. J.C. Reynolds. Types, abstraction and parametric polymorphism. In *Information Processing 83, Proc. of the IFIP 9th World Computer Congress*, pages 513–523. Elsevier Science Publishers B.V. (North-Holland), 1983.
31. D. Sannella and A. Tarlecki. Essential concepts of algebraic specification and program development. *Formal Aspects of Computing*, 9:229–269, 1997.
32. O. Schoett. Behavioural correctness of data representations. *Science of Computer Programming*, 14:43–57, 1990.
33. C. Strachey. Fundamental concepts in programming languages. Lecture notes from the International Summer School in Programming Languages, Copenhagen, 1967.
34. I. Takeuti. An axiomatic system of parametricity. *Fundamenta Informaticae*, 20:1–29, 1998.
35. R.D. Tennent. Correctness of data representations in Algol-like languages. In A.W. Roscoe, editor, *A Classical Mind: Essays in Honour of C.A.R. Hoare*. Prentice Hall International, 1997.
36. J. Zwanenburg. *Object-Oriented Concepts and Proof Rules: Formalization in Type Theory and Implementation in Yarrow*. PhD thesis, Tech. Univ. Eindhoven, 1999.