A relational perspective on types with laws

Ed Voermans Jaap van der Woude

January 13, 1993

Abstract

With relational transformational programming in mind, an extension of a "lawless" relational theory of datatypes is proposed in order to study and manipulate quotient types within a Tarski-like calculus of relations. The extended notion of type, pertype (from partial equivalence relation), is shown to admit a complete lattice structure by constructing the order via a Galois connection. A pertyping of relations is developed and inductive pertypes generated by equations are discussed. Pertypes do occur in model theory for λ-calculus but we are unaware of manipulations with inductive "lawful" types based on a simple relational calculus.

1 Introduction

Program construction and its theory are developing steadily towards an algebraic discipline combining methods from category theory, algebraic logic and lattice theory. The driving forces are the needs for uninterpreted manipulation, problem structuring and generalisation, and polymorphy, with a major role for calculational type theory and transformational program derivation. This development, mainly in the realm of functional programming, is well illustrated by the Bird-Meertens-Formalism (e.g. [7]).
In order to incorporate nondeterminism (e.g. for postponing design decisions) the idea of relational programming has been gaining attention [13, 4, 10, 8, 3]. In [1, 2] a relational theory of datatypes is under development with as goal the construction of an elegant, powerful, polymorphic calculational theory for transformational relational programming. This theory, however, is somewhat unsatisfactory in its current form because of the limited possibilities to treat (data)types with laws (equations), due to too restricted notions of type and typebuilder (relator, compare functor).

Here we propose an extended notion of type, the pertype (per from partial equivalence relation), and we discuss types with laws generated by equations. In this short paper we give a brief sketch of the framework and we treat only the type part, while only a few (representative) proofs are given. In a forthcoming extensive paper a full exhibition with proofs and more attention to the design decisions will be given.

The major features of this paper are the derivation of a complete lattice order for the pertypes, the construction of per domains and the formulation and the discussion of the notion of inductive type with law. Pertypes do figure in the literature on models for λ-calculus ([14, 9]) and types with laws are abundant, but we are unaware of a combination of the two based on a simple axiomatic relational calculus.

2 Spec algebra and monotypes

The axiomatic relational calculus, the *spec algebra*, on which the relational theory of datatypes is based is largely that of Tarski [15] with some notational changes. The set $\mathcal{A}$ of axiomatic relations (called *spec*ification*s*) is equipped with a complete distributive complemented lattice order ($\sqsubseteq$, top $\top\!\!\top$ and bottom $\perp\!\!\perp$), with a monoid structure for composition ($\circ$, unit I) and with an involution ($^\cup$, called *wok*). The interfaces are:

- $R\circ$ and $\circ R$ have Galois adjoints $R\backslash$ and $/R$ respectively; i.e.

(1) $$R\circ S \sqsubseteq T \equiv S \sqsubseteq R\backslash T \quad\text{and}\quad T \sqsupseteq S\circ R \equiv T/R \sqsupseteq S$$

with the useful cancellations:

(2) $$R\circ R\backslash T \sqsubseteq T \quad\text{and}\quad T \sqsupseteq T/R\circ R$$

- $^\cup$ is its own Galois adjoint: $R^\cup \sqsubseteq S \equiv R \sqsubseteq S^\cup$
- $^\cup$ is contravariant for $\circ$: $(R\circ S)^\cup = S^\cup \circ R^\cup$

A few other axioms (in particular, extensionality) don't play a role here and are omitted.

The spec algebra gets a structure that resembles an order enriched category with specs as arrows by determining objects and a typing of specs with respect to those objects. In the *monotype system* the objects are the subelements of the unit I for "$\circ$", the *monotypes* (they may be interpreted as the embeddings of subsets of the universe as partial diagonals). $\mathcal{M} = \{a \mid a \sqsubseteq I\}$ with the relative order is also a "powerset-like" lattice. Like set theoretical relations, specs have left and right domains in $\mathcal{M}$, the *monodomains*:

(3) $$R_< = R\circ\top\!\!\top \sqcap I = R\circ R^\cup \sqcap I \quad\text{and}\quad R_> \equiv \top\!\!\top\circ R\sqcap I = R^\cup\circ R \sqcap I$$

(4) $$b\circ R = R \equiv R_< \sqsubseteq b \quad\text{and}\quad R\circ a = R \equiv R_> \sqsubseteq a$$

compare the set version $R_< = \{x\in I \mid x\,R\,y \text{ for some } y \in I\}$.
The mono-typing of specs is defined as follows, for a, $b \in \mathcal{M}$ and spec R

(5) $$R : b \sim a \quad\equiv\quad b\circ R = R = R\circ a \quad (\equiv \quad b \sqsupseteq R_< \quad\wedge\quad R_> \sqsubseteq a)$$

Some familiar properties related to monotypes are, for a spec $R : b\sim a$:
totality ($R^\cup\circ R \sqsupseteq a$), *surjectivity* ($b \sqsubseteq R\circ R^\cup$), *functionality* ($b \sqsupseteq R\circ R^\cup$) and *injectivity* ($R^\cup \circ R \sqsubseteq a$). These definitions and their interpretations according to their nomenclature are the first (and only) indications for the "direction" of the composition.

This completes our brief summary of the spec algebra. More details about (e.g.) the introduction of units, sums, products and induction structures via the so-called *relators* (think of functors) may be found in [1, 2].

3 Quotient types

The spec algebra with monotypes (the monotype system) is a calculational theory of datatypes designed to facilitate algebraic manipulation. While many features of "standard" type theory are present, quotient types, i.e. datatypes with laws, are not studied internally. We cope with that omission in a rather standard way (e.g. see [14]):
From set theory it is well known that surjective images (quotients) are —upto isomorphism— in a one to one correspondence with equivalence relations. Moreover, such an image is isomorphic to a partitioning of the corresponding equivalence relation into cartesian squares (squares of the equivalence classes). In the situation at hand we have a host of sub(mono)types of I , the quotients of which can be represented as equivalence relations on those monotypes; or, what is the same, as partial equivalence relations on I .

We extend the notion of type from monotype to *pertype*: every *p*artial *e*quivalence *r*elation A is considered to be a type, the quotient type of its reflexive part $A_{\between}$ on which A is an *e*quivalence *r*elation. Of course this change in object choice requires redefinition of the typing of the specs. It also affects the notions of totality etc.; they should be extended to having pertypes as (co)domains. For the moment we only remark that a surjective total function between quotients, say $f : b/\!\!/B \leftarrow a/\!\!/A$, determines a total and surjective spec $R : b \sim a$ such that $R = R \circ R^{\cup} \circ R$. Such a spec is called a *difunctional* spec [12]. Moreover a total surjective difunctional spec induces a total surjective function between quotients. So the extension to pertypes also requires a study of difunctional specs.

4 The per lattice

A spec B is an *equivalence relation* (*er*) *on* $b \sqsubseteq I$ if

(6) $$B \circ B \sqsubseteq B \quad \wedge \quad B^{\cup} \sqsubseteq B \quad \wedge \quad B \sqcap I = b$$

and a *partial equivalence relation* (*per*) if it satisfies the first two conjuncts of (6), or equivalently

(7) $$B \circ B^{\cup} = B$$

Since a per is $^{\cup}$-symmetric, it follows that (introducing $B_{\between}$)

(8) $$B_{\between} := B \sqcap I = B_{<} = B_{>}$$

and a per B is an er on $B_{\between}$.
Let $\mathcal{B}_b = \{B \mid per.B \ \wedge \ B_{\between} = b\}$ denote the ers on b ; then $(\mathcal{B}_b , \sqsupseteq)$ is a complete lattice with top b , bottom $b \circ \top\!\!\top \circ b$ and (with $\bigvee$, $\bigwedge$ for lub and glb in $\mathcal{B}_b$)

(9) $$\bigvee \mathcal{A} = \sqcap \mathcal{A} \quad \text{and} \quad \bigwedge \mathcal{A} = (\sqcup \mathcal{A})^{+} \qquad \text{for} \quad \emptyset \neq \mathcal{A} \subseteq \mathcal{B}_b$$

where $^{+}$ is the transitive closure operator. Since the pers are the ers on the appropriate monotypes, this induces a rather naive sum-order on the pers: the *er order* ($\lhd\!|$)

(10) $\qquad A \lhd\!| B \;\equiv\; A_{\gtrless} \;=\; B_{\gtrless} \quad \wedge \quad A \;\sqsupseteq\; B \qquad (\equiv\; A_{\gtrless} \;\sqsubseteq\; B_{\gtrless} \quad \wedge \quad A \;\sqsupseteq\; B)$

However, this order leaves too many pers incomparable and it is certainly not a complete lattice order. Because we wish to exploit the Knaster-Tarski theorem when determining extreme solutions of equations, we seek a complete lattice order on the pertypes (as we had on the monotypes). Such an order does exist:

(11) Theorem *The relation* $\lhd$ *on pers defined by* $A \lhd B \;\equiv\; A \;=\; A \circ B$ *is a complete lattice order on pers that extends the er order* $\lhd\!|$

Proof. We only give a sketch of proof for the first part of the theorem. Let $\mathcal{E}$ be the collection of ers on I and define $\mathcal{Q} \subset \mathcal{M} \times \mathcal{E}$ by $\mathcal{Q} \;=\; \{(m, E) \,|\, m \circ E \sqsubseteq \top\!\!\top \circ m\}$. Then $\mathcal{Q}$ is closed under arbitrary lubs in the complete product lattice $(\mathcal{M} \times \mathcal{E}\,, (\sqsubseteq\,,\,\sqsupseteq\,))$, so it is a complete lattice itself. The pers and $\mathcal{Q}$ can be mapped to each other via $\lambda.(m, E) = m \circ E$ (see (13)) and $\tau.A \;=\; (A_{\gtrless},\, A\backslash A \,\sqcap\, (A\backslash A)^{\cup})$. Via the construction of the pair (τ, λ) as a Galois connection we derive the required order on the pers

$$
\begin{array}{ll}
 & \tau.A \;\; (\sqsubseteq\,,\,\sqsupseteq\,) \;\; (m,\, E) \\
\equiv & \{ \text{ def. } \tau \text{ and product order } \} \\
 & A_{\gtrless} \;\sqsubseteq\; m \quad \wedge \quad A\backslash A \,\sqcap\, (A\backslash A)^{\cup} \;\sqsupseteq\; E \\
\equiv & \{ \;\; A\,,\, E \text{ symmetric ; (1) ; } \; I \;\sqsubseteq\; E \;\; \} \\
 & A \circ m \;=\; A \quad \wedge \quad A \;=\; A \circ E \\
\equiv & \{ \text{ for " } \Leftarrow \text{" use (13) below to show } \; m \circ E \circ m \;=\; m \circ E \;\; \} \\
 & A \circ m \circ E \;=\; A \\
\equiv & \{ \text{ def. } \lambda \text{ ; derived order formula } \} \\
 & A \lhd \lambda.(m,\, E)
\end{array}
$$

It is easy to see that $\lhd$ is a partial order and that λ is surjective. Since $\mathcal{Q}$ is a complete lattice, it follows from the theory of Galois connections that the pers ordered via $\lhd$ form a complete lattice.
□

The per order on pertypes is not just a generalisation of the monotype order, it is a combination of a quotient and a subtype order; for (as is easily proved)

(12) $\qquad A \lhd B \;=\; A_{\gtrless} \;\sqsubseteq\; B_{\gtrless} \quad \wedge \quad A \;\sqsupseteq\; A_{\gtrless} \circ B$

If $A_{\gtrless} \circ B$ is a per it may be interpreted as a subtype of B and the second conjunct of (12) shows that A is a quotient of that subtype of B . The fact that $A_{\gtrless} \circ B$ is a per can be deduced from the second conjunct of (12) and the following useful technical lemma.

(13) Lemma *Let* A *be a per and* $b \sqsubseteq I$ *, then* $b \circ A$ *is a per iff* $b \circ A \sqsubseteq \top\!\!\top \circ b$ *.*

Proof.

$$
\begin{array}{l}
\quad b \circ A \ \text{ is a per} \\
\Rightarrow \ \{ \text{ pers are symmetric ; } b^{\cup} \ = \ b \ \} \\
\quad b \circ A \ \sqsubseteq \ \top\!\!\top \circ b \\
\Rightarrow \ \{ \ b \circ (\neg b \sqcap I) \ = \ \bot\!\!\bot \ \} \\
\quad b \circ A \circ (\neg b \sqcap I) \ = \ \bot\!\!\bot \\
\Rightarrow \ \{ \ b \circ A \ = \ b \circ A \circ b \sqcup b \circ A \circ (\neg b \sqcap I) \ \} \\
\quad b \circ A \ = \ b \circ A \circ b \\
\equiv \ \{ \ A \ = \ A \circ A^{\cup} \ ; \ b \ = \ b^{\cup} \ ; (7) \ \} \\
\quad b \circ A \ \text{ is a per}
\end{array}
$$

□

5 The per domain

The extension to pertypes requires also redefinition of the typing of a spec. A spec R relates quotients B and A if it makes no difference between B-equivalent inhabitants on the left nor between A-equivalent ones on the right; i.e.

$$(14) \qquad R \, : \, B \sim A \ \equiv \ B \circ R \ = \ R \ \wedge \ R \ = \ R \circ A$$

The difference with (5) is only the replacement of monotypes by pers. Such B and A as in (14) are left and right pertypes for R respectively. The *per right domain* $R_{\succ}$ of R is the $\lhd$-least right pertype for R . For a "construction" of the per right domain of R we need a technical lemma about the "symmetrisches Quotient" ([13]):

(15) Lemma *Let* $\grave{R} \ = \ R \backslash R \sqcap (R \backslash R)^{\cup}$ *then* $R \circ \grave{R} \ = \ R$ *and* $R_{>} \circ \grave{R}$ *is a per.*

Proof.
$$R \circ \grave{R} \sqsubseteq R \circ R \backslash R \sqsubseteq R \ = \ R \circ I \ \sqsubseteq \ R \circ \grave{R}$$

$$R_{>} \circ \grave{R} \sqsubseteq \top\!\!\top \circ R \circ \grave{R} \ = \ \top\!\!\top \circ R \sqsubseteq \top\!\!\top \circ \top\!\!\top \circ R_{>} \ = \ \top\!\!\top \circ R_{>}$$

□ (Annotation left to the reader)

With the above techniques we prove that $R_{\succ} \ = \ R_{>} \circ \grave{R}$ (compare (4)):

(16) Theorem *Let* A *be a per, then* $R \circ A \ = \ R \ \equiv \ (R_{>} \circ \grave{R}) \lhd A$ *.*

Proof.

$$
\begin{array}{ll}
& R \circ A \;=\; R \\
\Rightarrow & \{\ (1)\ ;\ \ (P \circ Q)_{>} \sqsubseteq Q_{>}\ \ \} \\
& A_{\between} \;\sqsupseteq\; R_{>} \quad \wedge \quad A \sqsubseteq R \backslash R \\
\Rightarrow & \{\ \ A \text{ is symmetric ; } \ R_{>} \circ \ \text{; def. } \ \grave{R}\ \ \} \\
& R_{>} \sqsubseteq A_{\between} \quad \wedge \quad R_{>} \circ \grave{R} \;\sqsupseteq\; R_{>} \circ A \\
\equiv & \{\ \ R_{>} \circ \grave{R} \text{ is a per with reflexive part } R_{>}\ \text{; (12) }\} \\
& R_{>} \circ \grave{R} \lhd A \\
\Rightarrow & \{\ \text{def } \lhd \ \ (11)\ ;\ \ R \circ \ \ \} \\
& R \circ R_{>} \circ \grave{R} \;=\; R \circ R_{>} \circ \grave{R} \circ A \\
\equiv & \{\ \ R \circ R_{>} \;=\; R\ \ \text{; (15) }\} \\
& R \;=\; R \circ A
\end{array}
$$

□

For the pertyping of specs this means that $R : B \sim A \;\equiv\; R_{\prec} \lhd B \quad \wedge \quad R_{\succ} \lhd A$, where $R_{\prec} \;=\; \acute{R} \circ R_{<}$ and $\acute{R} \;=\; R/R \sqcap (R/R)^{\cup}$.

6 (Di)functional specs

As mentioned before we need the "per versions" of the notions totality, surjectivity, functionality and injectivity. Let R be a spec with per typing $R : B \sim A$. The spec R "is" total if it acts on every member of the quotient represented by A . Because R does not distinguish between A-equivalent inhabitants; this means that $R_{>} \;\sqsupseteq\; A_{\between}$, which is equivalent to $R_{\succ} \lhd\!| \ A$ (since $R_{\succ} \lhd A$). So we define R to be *per total on* A if $R_{\succ} \lhd\!| \ A$, and "dually" *per surjective on* B if $R_{\prec} \lhd\!| \ B$.
For functionality it is required that two R-results from the same A-member should be B-equivalent; i.e. $R \circ R^{\cup} \sqsubseteq B$. So we define R to be *per functional to* B if $R \circ R^{\cup} \sqsubseteq B$ and "dually" *per injective from* A if $R^{\cup} \circ R \sqsubseteq A$. If R is per functional to B then $R_{\prec} \;=\; R \circ R^{\cup} \;=\; R_{<} \circ B$, so $R \circ R^{\cup}$ is a subtype of B (compare mono functionality);

indeed:

$$
\begin{array}{ll}
& R_{<} \circ B \\
\sqsupseteq & \{\ \text{per functionality ; } \ R_{<} \circ R \;=\; R\ \ \} \\
& R \circ R^{\cup} \\
\sqsupseteq & \{\ \ R \circ R^{\cup} \;\sqsupseteq\; R/R \circ R \circ R^{\cup} \;\sqsupseteq\; \acute{R} \circ R_{<} \;=\; R_{\prec}\ \ \} \\
& R_{\prec} \\
\sqsupseteq & \{\ \ R_{\prec} \lhd B \ \ ;\ \ R_{\prec\between} \;=\; R_{<}\ \text{; (12) }\}
\end{array}
$$

$$R_< \circ B$$

It follows that R is per functional for some per typing iff R is per functional to $R_\prec$ iff $R \circ R^\cup \;=\; R_\prec$. The specs with this property are precisely the difunctional specs:

(17) Lemma $R \circ R^\cup \;=\; R_\prec$ *iff* $R \circ R^\cup \circ R \;=\; R$ *iff* $R^\cup \circ R \;=\; R_\succ$.

Proof. Note that all three properties imply perness of $R \circ R^\cup$ and $R^\cup \circ R$.

$$
\begin{array}{ll}
& R^\cup \circ R \;=\; R_\succ \\
\equiv & \{\; R^\cup \circ R \lhd R_\succ \text{ since } R \circ R_\succ \;=\; R \;\} \\
& R_\succ \lhd R^\cup \circ R \\
\equiv & \{\; (16) \;\} \\
& R \circ R^\cup \circ R \;=\; R
\end{array}
$$

□

7 (Weak) relators

Many pers and specs can be constructed using the spec operators; additionally one would like to define specs via extreme solutions of equations, say $X :: X \;=\; \mathsf{F}.X$, where F is some spec transformer. We focus on special spec transformers, the so called *weak relators*, that preserve the spec algebra structures:

(18) $\mathsf{F}.(R \circ S) \;=\; \mathsf{F}.R \circ \mathsf{F}.S \quad , \quad \mathsf{F}.(R^\cup) \;=\; (\mathsf{F}.R)^\cup$ and $R \sqsubseteq S \;\Rightarrow\; \mathsf{F}.R \sqsubseteq \mathsf{F}.S$

The first two conjuncts in (18) imply that F preserves pers, difunctionals and the per order ($\lhd$). Because of $\sqsubseteq$-monotonicity F is also $\lhd\!|$-monotonic, which follows from

(19) Lemma $(\mathsf{F}.R)_> \;=\; (\mathsf{F}.R_>)_>$ *and* $(\mathsf{F}.R)_\succ \lhd\!| \;\mathsf{F}.R_\succ$ *with equality for difunctional* R
□ (proof omitted)

Examples of specs defined by a weak relator F are its $\sqsubseteq$- and $\lhd$-least fixpoints $\mu_s\mathsf{F}$ and $\mu_p\mathsf{F}$, which turn out to be equal:

(20) Lemma $\mu_s\mathsf{F} \circ A \;=\; \mu_s\mathsf{F}$ *for every* A *such that* $A \;=\; \mathsf{F}.A$ *. In particular,* $\mu_s\mathsf{F}$ *is a per and* $\mu_s\mathsf{F} \;=\; \mu_p\mathsf{F}$ *.*

Proof. (Sketch) Since both sides of the equation are F-fixpoints only the $\sqsubseteq$-part needs proof. To that end use (1) for "/" and $\sqsubseteq$-least fixpoint induction to obtain the stronger $\mathsf{F}.(\mu_s\mathsf{F}/A) \circ A \;\sqsubseteq\; \mu_s\mathsf{F}$. Since $A \;=\; \mathsf{F}.A$ and F is a relator we arrive via (2) at $\mathsf{F}.\mu_s\mathsf{F} \;\sqsubseteq\; \mu_s\mathsf{F}$, which is true.
For the "in particular" instantiate $(\mu_s\mathsf{F})^\cup$ and $\mu_p\mathsf{F}$ for A to show that $\mu_s\mathsf{F}$ is a per and that $\mu_s\mathsf{F} \lhd \mu_p\mathsf{F}$ and note that $\mu_p\mathsf{F} \lhd \mu_s\mathsf{F}$ by μ_p .
□

8 F-Types and laws

Just pers as unstructured types are not really interesting, for programming one would like to have types with "members" built by structural induction. To that end we define F-types and laws on them. Those definitions are slight adaptations of the usual ones in the functional world: weak relators replace the functors to determine the shape of the domain of the constructor, pers serve as types and the constructors are difunctional specs.

Let F be a (fixed) weak relator. An F-*type* is a pair $(T\,,\,\tau)$, where T is a per (the type) and $\tau\ :\ T \sim \mathsf{F}.T$ is a difunctional (the constructor), such that

(21) $$T \ = \ \tau_{\prec} \quad \text{and} \quad \tau_{\succ} \lhd\!| \ \mathsf{F}.T$$

meaning that τ is total on $\mathsf{F}.T$ and that τ is surjective and per-functional to T .
An example of an F-type is the $\sqsubseteq$-least fixpoint $\mu\mathsf{F}$ of F : $(\mu\mathsf{F}\,,\,\mu\mathsf{F})$ here the type is its own constructor. Such F-types (i.e. with $T \lhd\!| \ \mathsf{F}.T$) are called F-*inductive* types.
A law L on an F-type T is an "equivariant" er on T :

(22) $$L \lhd\!| \ T \quad \text{and} \quad L \ \sqsupseteq \ \tau \circ \mathsf{F}.L \circ \tau^{\cup}$$

The second conjunct may be interpreted as "constructions with L-equivalent type members are L-equivalent again". Indeed, $\tau^{\cup}$ breaks a T-member down to its constituents, all of those constituents are L-related to new constituents by $\mathsf{F}.L$ and composed to a new type member by τ ; the original and the result should be L-related.
The uninspiring proof of the following expected lemma is omitted.

(23) Lemma *Let* (T,τ) *be an* F*-type and* L *a law on* T *. Then* T *is a law on* $(T\,,\,\tau)$ *,* $(L\,,\,L \circ \tau)$ *is an* F*-type and* L *is an* F*-inductive type if* T *is*
$\square$

Note that a law L on an F-inductive type T is not only an F-inductive type itself, it is just an F-congruence on T , i.e. $L \lhd\!| \ T$ and $L \ \sqsupseteq \ \mathsf{F}.L$

For
$$\begin{array}{ll}
& L \ \sqsupseteq \ T \circ \mathsf{F}.L \circ T \\
\Leftarrow & \{ \ T \circ, \circ T \ \ ; \ L \ = \ T \circ L \circ T \ \text{ since } \ L \lhd T \ \} \\
& L \ \sqsupseteq \ \mathsf{F}.L \\
\equiv & \{ \ \mathsf{F}.L \ = \ \mathsf{F}.T \circ \mathsf{F}.L \circ \mathsf{F}.T \ \} \\
& L \ \sqsupseteq \ (\mathsf{F}.T)_{\between} \circ \mathsf{F}.L \circ (\mathsf{F}.T)_{\between} \\
\Leftarrow & \{ \ T \lhd\!| \ \mathsf{F}.T \ \text{ so } \ (\mathsf{F}.T)_{\between} \ = \ T_{\between} \ \ ; \ T_{\between} \sqsubseteq T \ \} \\
& L \ \sqsupseteq \ T \circ \mathsf{F}.L \circ T
\end{array}$$

The fact that F-inductive types give rise to more attractive formulae while they are closed under law-quotients (and contain the most interesting types) invites us to restrict our attention to the F-inductive types from now on.
Some interesting specimens may be constructed as follows: (proof omitted)

(24) Lemma *If* $X = R \circ \mathsf{F}.X$ *and* $X_{>} = (\mathsf{F}.X)_{>}$ *then* $X_{\succ}$ *is* F*-inductive.*

(For readers familiar with catamorphisms: every total F-catamorphism has an F-inductive right per-domain.)

9 Congruences

Traditionally, laws on types are equations and they induce congruences (in this paper laws are just those congruences). Since a congruence is an F-invariant equivalent relation, the construction of the induced congruence can be described in two phases: Construct the equivalence relation $E.\mathcal{S}$ induced by the equations ($\mathcal{S}$) and construct the $\sqsubseteq$-least F-invariant equivalence relation L containing $E.\mathcal{S}$. This separation of concerns is followed here and the second construction is treated first: the construction of the congruence induced by an er E .

To that end let T be an F-inductive type with domain $b = T_{\divideontimes}$ and let E be an er on b ; i.e. $E \lhd\!\!| \; b$. The F-congruence $T /\!\!/ E$ on T induced by E is the $\sqsubseteq$-least solution of the equation (in C)

$$C \; :: \; C \lhd\!\!| \; T \quad \wedge \quad C \sqsupseteq \mathsf{F}.C \quad \wedge \quad C \sqsupseteq E \tag{25}$$

Since T is F-inductive (on b), $b = (\mathsf{F}.b)_{\divideontimes}$; so the set $\mathcal{B} = \{A \,|\, A \lhd\!\!| \; b\}$ is F-invariant: $(\mathsf{F}.A)_{\divideontimes} = (\mathsf{F}.A_{\divideontimes})_{\divideontimes}$ by (19) and $(\mathsf{F}.A_{\divideontimes})_{\divideontimes} = (\mathsf{F}.b)_{\divideontimes} = b$. On $\mathcal{B}$ the orders $\sqsupseteq$ and $\lhd\!\!|$ coincide, and because of the first conjunct in (25), the equation involves only $\mathcal{B}$-members. Therefore, the definition of $T /\!\!/ E$ may be rephrased as the $\lhd\!\!|$-*greatest* solution of

$$C \; :: \; C \lhd\!\!| \; \mathsf{H}.C \quad \text{where} \quad \mathsf{H}.C = T \bigwedge \mathsf{F}.C \bigwedge E \tag{26}$$

Such a solution exists via Knaster-Tarski by virtue of $\lhd\!\!|$-monotonicity of F (and thus of H), completeness of the lattice $\mathcal{B}$ and F-invariance of $\mathcal{B}$ (and thus H-invariance).

Of course one may impose several laws on a type by considering several ers, either as a complete package or as a cascade; the resulting type should be the same. Indeed:

(27) Lemma *Let* E *and* M *be ers on* b *, then* $(T /\!\!/ E) /\!\!/ M = T /\!\!/ (E \wedge M)$ *.*

Proof. Since either side of the equation to be proven is F-inductive, it suffices to show for each F-inductive C that $C \lhd\!\!| \; (T /\!\!/ E) /\!\!/ M \;\; \equiv \;\; C \lhd\!\!| \; T /\!\!/ (E \wedge M)$. But this follows from the fact that, for F-inductive C and S and for $Q \lhd\!\!| \; S_{\divideontimes}$:

$$
\begin{array}{ll}
& C \lhd\!\!| \; S /\!\!/ Q \;\equiv\; C \lhd\!\!| \; S \wedge Q \\
\equiv & \{ \;\; S /\!\!/ Q \lhd\!\!| \; S \wedge Q \;\; ; \;\; C \lhd\!\!| \; \mathsf{F}.C \;\; \} \\
& C \lhd\!\!| \; S /\!\!/ Q \;\Leftarrow\; C \lhd\!\!| \; S \wedge \mathsf{F}.C \wedge Q \\
\equiv & \{ \;\; \lhd\!\!|\text{-greatest fixpoint induction} \; \} \\
& \mathsf{true}
\end{array}
$$

□

A spec X is said to *respect a per* E *on the right* if $X \circ E = X$, i.e. X does not distinguish between E-equivalent arguments. If L is a law on an F-inductive type T , say $L = T /\!\!/ E$, then one would expect X to respect L iff X respects both T and E . This is indeed the case for specs X that have the same F-recursive structure as T on the right side (We just wanted you to know, the proof is omitted):

(28) Theorem *Let* $X = R \circ \mathsf{F}.X$ *for some* R *. Then*

$$X \circ T /\!\!/ E = X \;\equiv\; X \circ T = X \;\wedge\; X \circ E = X$$

□

This was the first step: congruences induced by ers. Now for the second: ers induced by equations.
An equation "equates" two terms, possibly under certain conditions. A term λ is a function from a structure $\mathsf{G}.T$ over the type T to the type T ; i.e. $\lambda : T \leftarrow \mathsf{G}.T$.
This description is more precise than we need for this paper (but it does play a role if naturality conditions are to be put on terms). For our purpose a *term* on T is just a spec R with $T \sqsupseteq R \circ R^{\cup}$; so R has unnamed arguments and it values in the type T such that two values generated by R from the same argument are T-equivalent. An *equation* on T is a pair of terms with the same domain. So $(R\,,\,S)$ is an equation on T if

$$T \sqsupseteq R \circ R^{\cup} \sqcup S \circ S^{\cup} \quad \text{and} \quad R_{>} = S_{>}$$

Such an equation is to "equate" R- and S-values resulting from the same argument, i.e.

$$x = y \;\Leftarrow\; (\mathbf{E}n :: x\,R\,n \;\wedge\; y\,S\,n) \qquad (\equiv\; x\,(R \circ S^{\cup})\,y)$$

therefore the er $E.(R,S)$ generated by the equation $(R\,,\,S)$ is the $\sqsubseteq$-least er on $b = T_{\gtrless}$ that contains $R \circ S^{\cup}$, i.e. (where * is the reflexive transitive closure)

$$E.(R,S) = b \circ (R \circ S^{\cup} \sqcup S \circ R^{\cup})^{*}$$

Note that $b \sqsubseteq b \circ (R \circ S^{\cup} \sqcup S \circ R^{\cup})^{*} \sqsubseteq \top\!\!\top \circ b$; so, indeed, $E.(R,S)$ is an er on b .

Let $\mathcal{S}$ be a collection of equations, then the er $E.\mathcal{S}$ generated by $\mathcal{S}$ is

(29) $$E.\mathcal{S} = b \circ (\, \sqcup\{R \circ S^{\cup} \sqcup S \circ R^{\cup} \mid (R,S) \in \mathcal{S}\,\}\,)^{*}$$

A spec X is expected to respect $E.\mathcal{S}$ iff X doesn't distinguish between $\mathcal{S}$-equal terms:

(30) Theorem *Let* X *be a spec with* $X \circ T = X$ *then*

$$X \circ E.\mathcal{S} = X \quad \equiv \quad (\mathbf{A}(R,S) : (R,S) \in \mathcal{S} : X \circ R = X \circ S)$$

$\Box$ *(proof omitted)*

For the characterisation of $T /\!\!/ E.\mathcal{S}$ this means that $T /\!\!/ E.\mathcal{S}$ is the $\lhd$-greatest solution of

$$C :: \ C \lhd T \quad \wedge \quad C \lhd \mathsf{F}.C \quad \wedge \quad (\mathbf{A}(R,S) : (R,S) \in \mathcal{S} : C \circ R = C \circ S)$$

for $C \circ E.\mathcal{S} = C \ \equiv \ C \lhd E.\mathcal{S}$. Similarly, it follows from (30) that

$$X \circ T /\!\!/ E.\mathcal{S} = X \quad \equiv \quad X \circ T = X \ \wedge \ (\mathbf{A}(R,S) : (R,S) \in \mathcal{S} : X \circ R = X \circ S)$$

Moreover, since $E.(\mathcal{S}_0 \cup \mathcal{S}_1) = E.\mathcal{S}_0 \bigwedge E.\mathcal{S}_1$, we have $T /\!\!/ E.(\mathcal{S}_0 \cup \mathcal{S}_1) = T /\!\!/ E.\mathcal{S}_0 /\!\!/ E.\mathcal{S}_1$.

10 Epilogue

The development of a theoretical and calculational basis for relational programming has just started, and unless that basis is succesfully applied for some time to a variety of problems it is unclear whether the chosen approach is any good at all. Although theory and application should go hand in hand, the foundation should be satisfactory to start with. We feel, but such is a matter of personal taste, that some more "pure theoretical" research is necessary before engaging in the hard practical work. (In fact a lot more has already been done than could be presented here.)

The chosen approach extends the spec algebra as in [1, 2], and was only possible because of the calculational and methodological achievements in that lawless theory for relational programming. A similar development can be seen in the categorical theory of types for functional programming. The lawless theory is, by now, almost standard knowledge, but as soon as laws are considered, category theory is abandoned and signatures in all their baroqueness are all over the place. However, recent work of Fokkinga [5] shows that the lawless categorical approach can be extended to a lawful categorical theory. In parallel we see from the work of De Moor [3], that the categorical theory for functional programming can be extended to a categorical theory for relations and starting off with a suitable topos his relational system is much like the spec algebra (be it less calculational). Further research in the interrelations of those three approaches is necessary (and planned).

As far as factual program construction is concerned, we are strengthened in our belief, that the theory is heading in the right direction, by the work of Rietman [11] and Hoogendijk [6]. Their results seem to imply that the Bird-Meertens Formalism can be embedded completely in the spec algebra (with laws), but they also show that shaping up notation and theory is a conditio sine qua non.

References

[1] R.C. Backhouse et al. Relational catamorphisms. In B. Möller, editor, *Constructing Programs From Specifications*. North-Holland, may 1991.

[2] R.C. Backhouse et al. A relational theory of datatypes I, II. Lecture Notes STOP 1992 Summerschool on Constructive Algorithmics Part 3, 1992.

[3] O. de Moor. *Categories, relations and dynamic programming*. PhD thesis, Computing Laboratory, Oxford, 1992. Technical Monograph PRG-98.

[4] J. Desharnais and N.M. Madhavji. Abstract relational specifications. In M. Broy and C.B. Jones, editors, *Programming Concepts And Methods*, pages 267–284. North-Holland, 1990.

[5] M.M. Fokkinga. *Law and Order in Algorithmics*. PhD thesis, TU Twente, 1992.

[6] P.F. Hoogendijk. (Relational) programming laws in the Boom hierarchy of types. to appear in proc. Mathematics of Program Construction, Oxford, 1992.

[7] J.T. Jeuring. *Theories for Algorithm Calculation*. PhD thesis, RU Utrecht, 1993.

[8] G. Jones and M. Sheeran. Designing arithmetic circuits by calculation. Lecture Notes STOP 1992 Summerschool on Constructive Algorithmics Part 1, 1992.

[9] J.C. Mitchell. A type-inference approach to reduction properties and semantics of polymorphic expressions (summary). In G. Huet, editor, *Logical foundations of functional programming*, pages 197–212. Addison-Wesley, 1990.

[10] B. Möller. Relations as a program development language. In B. Möller, editor, *Constructing Programs From Specifications*. North-Holland, may 1991.

[11] F. Rietman. An aggregated segment sum theorem in the relational system. Master Thesis, RU Groningen, 1991.

[12] J. Riguet. Relations binaires, fermetures, corrrespondances de galois. *Bulletin de la Societé mathématique de France*, 76, 1948.

[13] G. Schmidt and T. Ströhlein. *Relationen und Graphen*. Reihe Mathematik für Informatiker. Springer Verlag, 1989.

[14] D. Scott. Data types as lattices. *SIAM J. Computing*, 5:522–587, 1976.

[15] A. Tarski. On the calculus of relations. *Journal of Symbolic Logic*, 6(3):73–89, 1941.