Записки научных
семинаров ПОМИ
Том 421, 2014 г.

S. N. Baranov, S. V. Soloviev

CONDITIONALLY REVERSIBLE COMPUTATIONS AND WEAK UNIVERSALITY IN CATEGORY THEORY

Abstract. Main attention is directed to the notion of weak universality in category theory. While the definitions based on the ordinary universal constructions usually hold up to isomorphisms, that is, unconditionnally reversible arrows, weakly universal constructions may be seen "positively" as defined up to conditionnally reversible arrows. It is shown that weak universality is closely connected with intensional equality, typically considered in categories used in computer science. As a possible application of weakly universal categorical constructions we suggest the notion of conditionnally reversible computation in the theory of computations.

§1. Introduction

Reversible computations attract attention of many researchers; however, such computations without conditions occur rarely. In practical computations physical limitations and imperfections must be taken into account (e.g., the probability of an error). In mathematical models reversibility depends in general on the model of computation. For example, it is well known that certain computations (like arithmetic expressions) may be reversible in the model of exact real arithmetic and not reversible in the floating point arithmetic (e.g., adding an integer constant). So, reversibility is conditional at least in this sense.

The conditions may be concrete (e.g., expressed in terms of values of certain parameters, like non-zero determinant of a matrix), or more abstract (expressed in general terms characterizing the environment or the history of computations). In both cases the study of reversibility is commonly associated with concrete models of computation, fixed in advance (logic gates and circuits, quantum gates[1] and circuits, various versions of real arithmetic, etc., cf. [6]).

Key words and phrases: weak universality in categories, extensional and intensional equality, conditionnally reversible computations.

This work was partially supported by the Climt project, ANR-11-BS02-016.

[1]The definition of quantum gates uses the notion of unitary matrix, i.e. the matrix U such that $UU^T = I$ where U^T is the conjugate transpose of U and I is identity

In this work we attempt to study very general abstract conditions of reversibility, unrelated to a concrete domain (such as real arithmetics) and a concrete model of computation. In fact, our starting point was an observation that in category theory the difference between the so called universal constructions and weakly universal constructions corresponds roughly to the difference between unconditional and conditional reversibility. It turned out that already at the level of basic examples there exists a relationship between this observation and the problem of canonical elements of datatypes. This problem is well known in logic and theoretical computer science [5]. Due to the difference between canonical and non-canonical elements (one may speak also about "concrete" and "abstract" elements), many constructions considered as universal in category theory (for example, product) are only weakly universal in computer science models. Thus, conditional reversibility may play an important role at the very basic level, when representation of data is considered.

§2. Universal and weakly universal constructions

Mac Lane [4, p. 55], defines the notion of a universal arrow as follows.

Definition 2.1. *If $S : D \to C$ is a functor and c is an object of C, then a universal arrow from c to S is a pair $\langle r, u\rangle$ consisting of an object r of D and an arrow $u : c \to Sr$ of C, such that to every pair $\langle d, f\rangle$ with d an object of D and $f : c \to Sd$ an arrow of C, there is a unique arrow $f' : r \to d$ of D with $Sf' \circ u = f$. In other words, every arrow f to S factors uniquely through the universal arrow u, as in the commutative diagram*

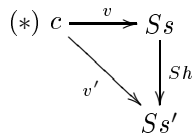
$$\begin{array}{ccc} c & \xrightarrow{u} & Sr \\ \| & & \downarrow{\scriptstyle Sf'} \\ c & \xrightarrow{f} & Sd \end{array} \qquad \begin{array}{c} r \\ \downarrow{\scriptstyle f'} \\ d \end{array}$$

Equivalently (Mac Lane continues), $u : c \to Sr$ is universal from c to S when the pair $\langle r, u\rangle$ is an initial object in the comma category $(c \downarrow S)$ .... As with any initial object, it follows that $\langle r, u\rangle$ is unique up to *isomorphism* in $(c \downarrow S)$; in particular, the object r of D is unique up to isomorphism in D.

---

matrix. Such condition in general cannot be verified constructively. This, to our opinion, illustrates the level of "conditionality" of certain models of reversibility.

Let us elaborate this in slightly more details. Recall that *comma category* $(c \downarrow S)$ for an object $c \in \mathrm{Ob}\,(C)$ and a functor $S : D \to C$ is the category whose objects are pairs $\langle s, v\rangle$ with $s \in \mathrm{Ob}\,(D)$ and $v : c \to Ss \in \mathrm{Mor}\,(C)$, and whose morphisms $h : \langle s, v\rangle \to \langle s', v'\rangle$ are the morphisms $h : s \to s' \in \mathrm{Mor}\,(C)$ such that the diagram



is commutative [4, p. 46].

Notice that the equality of objects in $(c \downarrow S)$ is "heterogenous:" $\langle s, v\rangle = \langle s', v'\rangle$ iff $s = s'$ in D and $v = v'$ in C. The equality of morphisms comes from D: $h : \langle s, v\rangle \to \langle s', v'\rangle$ iff $h = h'$ in D.

Remark 2.2. Still, other equality relations may be of use. In our discussion of conditional reversibility we shall consider the following equality:

$$h =_w h' : \langle s, v\rangle \to \langle s', v'\rangle \;\text{ iff }\; Sh \circ v = Sh' \circ v.$$

Obviously, if we take $=_w$ instead of $=$ we obtain a factor category of $(c \downarrow S)$ that we will denote by $(c \downarrow S)^*$.

Initiality of $\langle r, u\rangle$ above means that for any other object $\langle r', u'\rangle$ there exists a unique arrow $f : r \to r'$ that makes (*) commutative. If we have another realization $\langle r', u'\rangle$ of the universal arrow, then there exist unique $f : r \to r'$ and $f' : r' \to r$ that must be mutually inverse isomorphisms.

In spite of its triviality, let us remind the proof of this fact, since we will need in the end of this section to show exactly what the difference is in case of weak universality.

First, let us take the same pair $\langle u, r\rangle$ as $\langle u', r'\rangle$. The identity morphism $1_r : r \to r$ may be taken as f' in the definition, and because of unicity it is the only f' possible. Now, if we take a different pair $\langle u', r'\rangle$ then by definition we will have certain $f' : r \to r'$ and $f'' : r' \to r$, such that $u = Sf'' \circ (Sf' \circ u) = S(f'' \circ f') \circ u$. By unicity $f'' \circ f' = 1_r$. In a similar way, we derive that $f' \circ f'' = 1_{r'}$ and hence f' and f'' are mutually inverse isomorphisms in D.

Thus, as for isomorphisms in general, one may speak about *non-conditional reversibility* of arrows between the realizations of a universal arrow.

To stress the importance of universal constructions (universal arrows are a particular case of these constructions), one may recall principal examples

from [4]. In these examples the notion of a universal arrow is used in category theory (in a very general abstract way) to define:

- bases of vector spaces;
- free categories from graphs;
- fields of quotients;
- complete metric spaces.

Universal constructions play central role in category theory, and this list may be easily extended. Our main interest, though, is to a more constructive notion of reversibility, and below we will consider in more details examples that are closer to computer science.

As we have seen, universality implies unicity up to isomorphism of the universal arrow $\langle r, u\rangle$. The definition of a *weak universal arrow* ([4, p. 235]) differs from the definition of a universal arrow only in that f' in the diagram is *not* required to be unique. As Mac Lane remarks, it is possible to modify all the various types of universals, defining weak products, weak limits, weak coproducts (requiring just existence rather than uniqueness in each case). There is no more unicity up to isomorphism, but it does not mean that instead of isomorphisms we will have arbitrary arrows. Some *conditional* reversibility will be preserved.

In the proposition below we use the same notation as in the definition 2.1 above.

Proposition 2.3. *Let the pair $\langle u, r\rangle$ be a weak universal arrow. Then r is unique up to the isomorphism in the factor category $(c \downarrow S)^*$.*

Proof. Without unicity condition, we still have the equalities

$$u = S(f'' \circ f') \circ u \quad \text{and} \quad u' = S(f' \circ f'') \circ u',$$

and they correspond exactly to the definition of isomorphism in $(c \downarrow S)^*$. □

Remark 2.4. The property that defines an isomorphism in $(c \downarrow S)^*$ may be seen as *conditional reversibility* (in this case, the reversibility that has the composition with u as a precondition, and "modulo" application of S).

§3. Categories and computer science: computational aspects

Approaches to categories of the "mainstream" category theory and of the "mainstream" computer science are very different, but mathematical models based on categories *are* often used in computer science [1].

One point where this difference of approaches may be seen very clearly, is the role played by computational aspects of equality. Usually in category theory equality of objects and morphisms plays purely technical role in the definition of categories. In computer science its conceptual importance is much greater.

There is, for example, an opposition between so called intensional and extensional equalities. Two programs are extensionally equal if they always produce equal outputs for equal inputs. Intensional equality is defined usually w.r.t. a certain system of conversions (syntactic transformations corresponding to certain basic identities).

For illustrative purposes, we shall consider λ-terms of simply typed λ-calculus with inductive types as representations of programs (for details see, e.g. [2]; below all explanations are limited to a necessary minimum). Let us denote this system by T_{ind}.

Intensional equality in T_{ind} is defined w.r.t. α-conversion (renaming of bound variables), β-conversion $(\lambda x : A.t)s = t[s/x]$ where A is a type, $t[s/x]$ denotes the result of substitution of s for x in t with renaming of bound variables to avoid capture, η-conversion $\lambda x : A.(tx) = t$ if x is not free in t, and ι-conversion representing one step of recursion.[2]

There are several variants of definition of extensional equality, depending on what is considered as elements of a type (admissible inputs of a program), for example all terms of type A, closed terms of type A, or so called "canonical elements" (for inductive types).

Intensional equality of terms implies their extensional equality, but not vice versa. Intensional equality in the main systems of λ-calculus considered in computer science (including the system mentioned above) is decidable due to strong normalization and confluence results [1, 2].

The situation is different for extensional equality. Consider, e.g., the inductive type $\mathrm{Nat} =_{\mathrm{def}} \mathrm{Ind}\,(\alpha)(0 : \alpha, \mathrm{succ} : \alpha \to \alpha)$[3]. The functions $\mathrm{Nat} \to \mathrm{Nat}$ definable in T_{ind} include all primitive recursive functions and their natural extensional equality is algorithmically undecidable.

There is a natural structure of category defined on T_{ind}. By abuse of notation we shall denote this category by T_{ind} as well.

[2]Terms of T_{ind} include recursion operators over inductive types.

[3]The symbol α in the defintions of inductive types is a "placeholder," i.e., the types of constructors of an inductive type are obtained by substitution of (the name of) this type for α, for example $0 : \mathrm{Nat}$, $\mathrm{succ} : \mathrm{Nat} \to \mathrm{Nat}$.

Objects of T_{ind} are types. Morphisms from A to B are closed λ-terms of the type $A \to B$. Identity is represented by the term $\lambda x : A.x$ and composition of $s : A \to B$ and $t : B \to C$ by $\lambda x : A.(t(sx))$ (where x does not occur freely in s, t). The terms must be considered up to a certain equivalence relation defining equality of morphisms. Since several kinds of equality may be considered, the precise definition of the category structure on T_{ind} depends on this choice. For example, we may take terms up to intensional equality, i.e., up to an equvalence relation defined by conversions (one may also define $t \equiv s$ iff the normal forms of t and s are identical), or up to one of the kinds of extensional equality mentioned above.

What may be said about universal constructions in T_{ind}, if we take into account the computational aspects of equality?

Some universal constructions will fail with respect to intensional equality, and even w.r.t. extensional equality. Some will become weakly universal.

§4. Case studies

4.1. The system T_{ind}. The system of λ-calculus considered below is a subsystem of the simply typed λ-calculus with inductive types, considered in detail in [2]. The system below is more restricted: we excluded from the syntax a "canonical" terminal object and pairing. In [2], the relationship of this "canonical" data with singletons and pairing defined using inductive type construction is studied in presence of additional reductions. Here we want to use it only to illustrate the general principles discussed above, and these extra data would be a distraction.

Definition 4.1. *Types are either atomic types or obtained by application of type constructors.*

Atomic types are elements of a finite or infinite set $\mathcal{S} = \{\alpha, \beta, \ldots\}$ of type variables.

Type constructors are:

- $\to$ *for functional types, which constructs $A \to B$ for any types A and B*
- Ind, *defined as follows: let $\mathcal{C}$ be an infinite set of* introduction operators *(constructors of* elements *of inductive types), with $\mathcal{C} \cap \mathcal{S} = \varnothing$. an inductive type with n constructors $c_1, \ldots, c_n \in \mathcal{C}$, each*

of them having the arity k_i (with $1 \leqslant i \leqslant n$), has the form:

$$\mathrm{Ind}(\alpha)\{c_1 : A_1^1 \to \ldots \to A_1^{k_1} \to \alpha\,;\ \ldots\ ;\ c_n : A_n^1 \to \ldots \to A_n^{k_n} \to \alpha\},$$

Here, every $A \equiv A_i^1 \to \ldots \to A_i^{k_i} \to \alpha$ is an inductive schema, i.e., A_i^j is:

- *either a type not containing α; (we call this A_i^j a* non-recursive operator*);*
- *or a type of the form $A_i^j \equiv C_1 \to \ldots \to C_m \to \alpha$, where α does not appear in any $C_{\ell \in 1..m}$ (such A_i^j are called* strictly positive operators*).*

Here Ind (α) binds *the variable α.*

Example 4.1. (Definition of types Bool, Nat, and T_ω, the type of ω-trees.)
$\mathrm{Bool} = \mathrm{Ind}(\alpha)\{T : \alpha \,|\, F : \alpha\}$
$\mathrm{Nat} = \mathrm{Ind}(\alpha)\{0 : \alpha \,|\, \mathrm{succ} : \alpha \to \alpha\}$
$T_\omega = \mathrm{Ind}(\alpha)\{0_\omega : \alpha \,|\, \mathrm{succ}_\omega : \alpha \to \alpha \,|\, L_\omega : (\mathrm{Nat} \to \alpha) \to \alpha\}$.

Definition 4.2. *Let $\mathcal{V}$ be an infinite set of variables $\mathcal{V}$ (with $\mathcal{V} \cap \mathcal{S} \cap \mathcal{C} = \varnothing$). The set of* λ-terms *is generated by the following grammar rules:*

$$M \ ::= \ c \,|\, \mathrm{Rec}^{B \to D} \,|\, x \,|\, (\lambda x : B \cdot M) \,|\, (M \ M)$$

where $x \in \mathcal{V}$, $c \in \mathcal{C}$, B and D are arbitraty types, and $\mathrm{Rec}^{B \to D}$ denotes the recursion operator from B to D (for details, see [2, 3]).

All terms and types are considered up to α-conversion, *i.e.*, renaming of bound variables. Context Γ is a set of term variables with types $x_1 : A_1, \ldots, x_n : A_n$ ($x_1, \ldots, x_n$ should be distinct). Γ, Δ denotes union of the contexts Γ, Δ (we assume that Γ, Δ have no common term variables).

Definition 4.3. *There are the following typing axioms and rules for the terms defined above (A, B, D denote arbitrary types, Γ is an arbitrary context).*

Axioms:

- $\Gamma, x : A \vdash x : A$,*;*
- *For each inductive type $C = \mathrm{Ind}(\alpha)\{c_1 : A_1 \,|\, \ldots \,|\, c_n : A_n\}$ and $1 \leqslant i \leqslant n$*

$$\Gamma \vdash c_i : A_i[B/\alpha]$$

(for example, if $C = \mathrm{Nat}$, then $\Gamma \vdash 0 : \mathrm{Nat}$ and $\Gamma \vdash \mathrm{succ} : \mathrm{Nat} \to \mathrm{Nat}$);

- *For C as above and any type D the axiom:*[4]

  $\Gamma \vdash \mathrm{Rec}^{C\to D} : \Upsilon_C(A_1, D) \to \ldots \to \Upsilon_C(A_n, D) \to C \to D.$

Typing rules.

$$\frac{\Gamma, x : A \vdash M : B}{\Gamma \vdash (\lambda x : A \cdot M) : A \to B}(\lambda)$$

$$\frac{\Gamma \vdash M : A \to B \quad \Gamma \vdash N : A}{\Gamma \vdash (M\ N) : B}(\mathrm{app})$$

The constant $\mathrm{Rec}^{C\to D}$ is called the *recursor* from C to D. Notice that applying it (using the rule app) to the terms $M_1 : \Upsilon_C(A_1, D), \ldots, M_n : \Upsilon_C(A_n, D)$ we define the function $\mathrm{Rec}^{C\to D} M_1 \ldots M_n : C \to D$. The following derived rule is often included:

$$\frac{\Gamma \vdash M_i : \Upsilon_C(A_i, D)\ \ (1 \leqslant i \leqslant n)}{\Gamma \vdash (\mathrm{Rec}^{C\to D}\ M_1\ \ldots\ M_n) : C \to D}(\mathrm{elim})$$

Normalization and equality. The terms of the system T_{ind} are considered up to equality generated by conversion relation. The α-conversion (renaming of bound variables) was already mentioned. Other conversions are:[5]

(i) β-conversion $(\lambda x : A.M)N = [N/x]M$;

(ii) η-conversion $\lambda x : A.(Mx) = M$ (where x must not be free in M);

(iii) and ι conversion for recursion. The ι-conversion corresponds to one step in recursive computation. For example, in case of $\mathrm{Rec}^{\mathrm{Nat}\to\mathrm{Nat}}$ it is

- $(\mathrm{Rec}^{\mathrm{Nat}\to\mathrm{Nat}} \mathrm{ag})(0) \to_\iota a$,
- $(\mathrm{Rec}^{\mathrm{Nat}\to\mathrm{Nat}} \mathrm{ag})(Sx) \to_\iota gx((\mathrm{Rec}^{\mathrm{Nat}\to\mathrm{Nat}} \mathrm{ag})x)$.

T is confluent and strongly normalizing with respect to $\beta\eta\iota$-reductions (directed conversions). Detailed description and normalization theorems for T_{ind} can be found in [2]. Thus, the equivalence relation on terms based on conversion (often called $\beta\eta\iota$-equality) is decidable.

---

[4] $\Upsilon_C(A, D)$ are certain auxilliary types used to define recursion from C to D. They correspond to the types of functions that appear in standard recursive equations over C. For example, if $C = D = \mathrm{Nat}$, $A_1 = \alpha$ (the type of constant 0), $A_2 = \alpha \to \alpha$ (the type of successor succ) in the definition of Nat, then $\Upsilon_{\mathrm{Nat}}(A_1, \mathrm{Nat}) = \mathrm{Nat}$, $\Upsilon_{\mathrm{Nat}}(A_2, \mathrm{Nat}) = \mathrm{Nat} \to \mathrm{Nat} \to \mathrm{Nat}$. In more general dependent type case a detailed description of these auxilliary types may be found in [3, p. 178]

[5] We omit the contexts and types of terms.

Categorical structure on T_{ind}. The objects of the category T_{ind} are types, described above. The morphisms from A to B are closed terms (i.e., terms that do not contain free term variables) derivable in T_{ind} $\vdash f : A \to B$ considered up to $\beta\eta\iota$-equality. Below "intensional equality" means $\beta\eta\iota$-equality. Speaking about these terms we shall usually omit $\vdash$. The composition of $f : A \to B$ and $f' : B \to C$ is defined as the (equivalence class of) $f' \circ f =_{\mathrm{def}} \lambda x : A.(f'(fx))$. The identity is defined as the (equivalence class of) $\mathrm{id}_A =_{\mathrm{def}} \lambda x : A.x : A \to A$. The axioms of category are trivially satisfied.

Other kinds of equivalence of terms of functional types $f, f' : A \to B$ that we shall consider below and call "extensional equality" are all defined using the conditions of the form:

- For all t of type A satisfying certain condition $ft =_{\beta\eta\iota} f't$.

The equality $f =_{\beta\eta\iota} f'$ implies $ft =_{\beta\eta\iota} f't$. Thus the "extensional equality" below always contains the "intentional equality" and thus defines another categorical structure on T_{ind}.

Below we shall consider several representative examples.

4.2. Intensional and extensional equality in T_{ind}. Let us consider two terms of T_{ind}:

- $f_1 = \mathrm{succ} : \mathrm{Nat} \to \mathrm{Nat}$ and
- $f_2 = \mathrm{Rec}^{\mathrm{Nat}\to\mathrm{Nat}}(\lambda y : \mathrm{Nat}.\mathrm{succ})(\mathrm{succ}\, 0) : \mathrm{Nat} \to \mathrm{Nat}$.

Each term is a morphism of T_{ind}. Each term represents also a function on the terms of type Nat defined by $f_i(t) =_{\mathrm{def}} f_i t$.

Canonical elements of Nat are represented by the terms $0, \mathrm{succ}\, 0, \ldots, S(\ldots(\mathrm{succ}\, 0))$, and on any canonical element $n : \mathrm{Nat}$ both f_1 and f_2 have the value $\mathrm{succ}\, n$. At the same time f_1 and f_2 are not intensionally equal: both are already in normal form and these normal forms are different.

We can define also a one side inverse to f_1 with respect to intensional equality, given by $f' =_{\mathrm{def}} \mathrm{Rec}^{\mathrm{Nat}\to\mathrm{Nat}}(\lambda x : \mathrm{Nat}.\lambda y : \mathrm{Nat}.x)0$ (the value of f' on $\mathrm{succ}\, n$ will be n, the value on 0 will be 0). For the composition with f_1,

$$\lambda x : \mathrm{Nat}.((\mathrm{Rec}^{\mathrm{Nat}\to\mathrm{Nat}}(\lambda x : \mathrm{Nat}.\lambda y : \mathrm{Nat}.x)0)(\mathrm{succ}\, x)) \to_\iota$$

$$\lambda x : \mathrm{Nat}.(\lambda x : \mathrm{Nat}.\lambda y : \mathrm{Nat}.x)x(\mathrm{succ}\, x)) =_\beta \lambda x : \mathrm{Nat}.x =_{\mathrm{def}} \mathrm{id}_{\mathrm{Nat}}.$$

If we compose f' with f_2, applying the composition to canonical elements we have

$$f'(f_2 0) =_{\beta\eta\iota} 0, \quad f'(f_2(n)) =_{\beta\eta\iota} f'(\mathrm{succ}\, n) =_{\beta\eta\iota} n,$$

but the composition itself

$\lambda x : \mathrm{Nat}.(\mathrm{Rec}^{\mathrm{Nat}\to\mathrm{Nat}}(\lambda x : \mathrm{Nat}.\lambda y : \mathrm{Nat}.x)0(\mathrm{Rec}^{\mathrm{Nat}\to\mathrm{Nat}}(\lambda y : \mathrm{Nat}.\mathrm{succ})(\mathrm{succ}\,0)x))$

is normal (i.e., does not admit any reduction) and so it is not equal to $\mathrm{id}_{\mathrm{Nat}}$.

- It can be shown that f_2 does not have left inverse w.r.t. intensional equality at all (because $(\mathrm{Rec}^{\mathrm{Nat}\to\mathrm{Nat}}(\lambda y : \mathrm{Nat}.\mathrm{succ})(\mathrm{succ}\,0)x)$ is normal).
- This behaviour can be seen as a case of conditional reversibility: f_2 is reversible at the left if the arguments are canonical elements.
- One may notice, that the notion of canonical element can be easily generalized to inductive types different from Nat. In fact, it can be shown (by induction on the structure of normal terms in T_{ind}) that any closed term M of the type A where A is an inductive type begins by application of one of introduction operators (constructors of elements) of A. If A does not have constructors with functional arguments (e.g., is a so called "algebraic type") then M is constant and represents a canonical element of A.
- More categorical view at this conditional reversibility would be that some functor from the category T_{ind} to the category of sets such that the types become sets of their canonical elements is applied first (and equality of morphisms in this "target" category is the extensional equality of functions represented by λ-terms).

4.3. Weak terminal objects in T_{ind}. An inductive type with one element may be defined in T_{ind} as $\mathrm{Ind}(\alpha)\{c : \alpha\}$. Allowing some abuse of notation, we shall denote this type by $\{c\}$. The constant c may be considered as (the name of) its unique element. The related typing axiom is $\Gamma \vdash c : \{c\}$. There are other such types, obtained by changing c.

The definition of a terminal object $\top \in T_{\mathrm{ind}}$ as a universal construction (in the strong sense) is equivalent to the condition that for every object $A \in T_{\mathrm{ind}}$ there exists the unique $f : A \to \top$. For a weak terminal object only the existence is required. If we take any of the types $\{c\}$, for any A there is $\lambda x : A.c : A \to \{c\}$, but other non-equivalent closed terms of the same type may exist (for example, defined using recursors).

The recursor $\mathrm{Rec}^{\{c\}\to A}$ has the type $A \to \{c\} \to A$, i.e., the functions from $\{c\}$ to A are defined by application of $\mathrm{Rec}^{\{c\}\to A}$ to $a : A$, an obvious interpretation is that they are defined by their value on the unique element $c : \{c\}$. Still, with respect to intensional equality $\mathrm{Rec}^{\{c\}\to\{c'\}}c' : \{c\} \to \{c'\}$

is not equal to $\lambda x : \{c\}.c' : \{c\} \to \{c'\}$. Moreover, with respect to this equality the morphisms $\lambda x : \{c\}.c' : \{c\} \to \{c'\}$, $\lambda x : \{c'\}.c : \{c'\} \to \{c\}$, $\mathrm{Rec}^{\{c\}\to\{c'\}}c' : \{c\} \to \{c'\}$, $\mathrm{Rec}^{\{c'\}\to\{c\}}c : \{c'\} \to \{c\}$ are not mutually inverse isomorphisms. For example, the composition of first two gives

$$\lambda y : \{c\}.(\lambda x : \{c\}.c'((\lambda x : \{c'\}.c)y)) =_{\beta\eta\iota} \lambda y : \{c\}.c \neq \lambda y : \{c\}.y$$
$$=_{\mathrm{def}} \mathrm{id}_{\{c\}}.$$

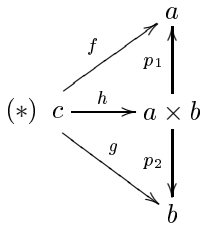
The composition of second two is a normal term and so also is not equal to $\mathrm{id}_{\{c\}}$. It is possible to show that with respect to intensional equality they are not isomorphisms at all.

The same remark as in the end of the previous subsection can be added concerning conditional reversibility.

4.4. Product as a weakly universal construction. Let us take as an example the notion of product $a \times b$ of two objects a, b of a category C. It can be defined using the notion of universal arrow from diagonal functor $\Delta : C \to C \times C$ (in functor category) to the functor $F : \{1, 2\} \to C$ from discrete category $\{1, 2\}$ to C (with $F(1) = a, F(2) = b$). The details can be found in [4, p. 69]. We shall skip them (only the fact that this may be seen as a particular case of the notion of universal arrow is important) and pass directly to more common equivalent definition using projections.

The object $a \times b \in \mathrm{Ob}(C)$ is called product of two objects $a, b \in \mathrm{Ob}(C)$ iff

- there exist the unique arrows $p_1 : a \times b \to a$ and $p_2 : a \times b \to b$ (called projections) such that
- for every object $c \in \mathrm{Ob}(C)$ and two arrows $f : c \to a, g : c \to b$ there exists a unique arrow $h : c \to a \times b$ that makes the following diagram commute:



The arrow h is denoted $\langle f, g\rangle$. It is usually called product (or pair) of f, g, and f, g are called its components. Universality (in the strong sense) of this construction is reflected by the condition of unicity of projections and h. One of the consequences is that $a \times b$ is unique up to isomorphism.

Let us consider now, how all this will work in T_{ind}. Given two types A, B, an inductive type usually called product of A, B (cf. [3]) is defined as follows:

$$A \times B =_{\mathrm{def}} \mathrm{Ind}(\alpha)(\mathrm{pair} : A \to (B \to \alpha)).$$

Its canonical elements are terms of the form $(\mathrm{pair}s)t$ where $s : A$ and $t : B$. There may be other elements that do not have the constructor pair at their head. For example, if we admit open terms as elements, the variable $x : A \times B$ is a non-canonical element.

It turns out that in T_{ind} with intensional equality $A \times B$ can not be considered as product in the sense of strong universality.

As any inductive type, $A \times B$ in T_{ind} comes equipped with recursion operators. The recursion operator from $A \times B$ to D is a constant $R : (A \to (B \to D)) \to (A \times B \to D)$. The corresponding ι-conversion is $(\mathrm{Rf})((\mathrm{pair}t_1)t_2) = (ft_1)t_2$ with $f : A \to (B \to D)$, $t_1 : A, t_2 : B$. Notice that if $s : A \times B$ is not of the form $(\mathrm{pair}t_1)t_2$ then the conversion is not applicable. Let us denote R_1 and R_2 the recursion operators from $A \times B$ to A and B respectively. Projections are defined now as $p_1 = R_1(\lambda x : A.\lambda y : B.x) : A \times B \to A$ and $p_2 = R_2(\lambda x : A.\lambda y : B.y) : A \times B \to B$.

Given two terms $f : C \to A$ and $g : C \to B$, h of the diagram $(*)$ may be defined as $h = \langle f, g\rangle =_{\mathrm{def}} \lambda z : C.(\mathrm{pair}f(z))g(z)$. The diagram will be commutative, but what about the unicity of h?

Let $h : C \to A \times B$ be a variable. Let $f = p_1 \circ h$, $g = p_2 \circ h$. The diagram $(*)$ will be commutative. Let us take $h' = \langle p_1 \circ h, p_2 \circ h\rangle$. The diagram will be commutative, but h and h' are not equal w.r.t. the intensional equality.

The product in T_{ind} is only weakly universal. It is possible to define another product as $A \times' B =_{\mathrm{def}} \mathrm{Ind}(\alpha)(\mathrm{pair}' : A \to (B \to \alpha))$ (the only modification is the name of the constructor). The "products" $A \times B$ and $A \times' B$ will *not* be isomorphic in T_{ind} with intensional equality.

More precisely, let $\langle f, g\rangle' =_{\mathrm{def}} \lambda z : C.((\mathrm{pair}'f(z))g(z))$. The "candidates" to the role of isomorphisms are obvious:

$$\theta = \langle p_1, p_2\rangle' : A \times B \to A \times' B, \quad \theta' : \langle p_1', p_2'\rangle : A \times' B \to A \times B,$$

but they are not mutually inverse w.r.t. intensional equality. (It is possible to show that there is no isomorphism at all.)

Consider the following diagram:

$$C \xrightarrow{h} A \times B \underset{\theta'}{\overset{\theta}{\rightleftarrows}} A \times' B \ .$$

The morphisms θ and θ' are not mutually inverse, but they are mutually inverse conditionally, in the following sense. If $h = \langle f, g\rangle$ for some $f : C \to A$ and $g : C \to B$ then $(\theta' \circ \theta) \circ h = h$. The precondition of inversibility is that the morphism is applied in some sense to canonical elements (it is guaranteed by $h = \langle f, g\rangle$).

§5. Conclusion

The aim of this paper was to attract attention to the fact that conditional reversibility (as opposed to unconditional reversibility) may be connected to another, very fundamental opposition between "strong" and weak universality in category theory, which is connected in its turn to the opposition between the non-constructive approach of the "mainstream" category theory and the constructive approach of computer science. This is work in progress, and we were able to present only some general considerations and, we believe, a representative example. Product is a very simple example of a universal construction and of an inductive type, but similar problems arise with more complex inductive types and universal constructions. This example, in particular, emphasizes the problem of the "well-preparedness" of input data that will guarantee the reversibility of computations (in the example represented by condition $h = \langle f, g\rangle$).

Another question is future work. Categorical diagrams are an excellent formalism. They may be treated constructively as attributed graphs. We did work before with graph transformations and we plan to apply this approach to categorical diagrams defining universal constructions in order to localize the "weak" arrows, i.e., the arrows where the unicity will be lost when we take a constructive notion of equality. Practical applications of this approach seem to be quite promising in formal verification of software programs and root cause analysis of software defects. Reversing computation backward from a point where an error was detected, allows the user to easily come to the respective wrong inputs or wrong logic which caused the error, provided the forward computations were reversible. The discussed approach allows us to specify criteria for such reversibility; so the user's

task is to design a realization of the desired computation which satisfies these criteria for reversibility.

References

1. A. Asperti, G. Longo, *Categories, types, and structures – an introduction to category theory for the working computer scientist.* — Found. Comput. MIT Press, 1991.
2. D. Chemouil, *Isomorphisms of simple inductive types through extensional rewriting.* — Math. Structures Computer Sci. **15**(5) (2005), 875–917.
3. Z. Luo, *Computation and reasoning. A type theory for computer science.* — International Series of Monographs on Computer Science **11**, Oxford Science Publications, Clarendon Press, Oxford, 1994.
4. S. Mac Lane, *Categories for the working mathematician.* 2nd edition, Graduate Texts in Mathematics **5**, Springer-Verlag, 1998.
5. P. Martin-Löf, *Intuitionistic type theory.* — Notes by G. Sambin of a series of lectures given in Padua, June 1980. Bibliopolis, 1984.
6. M. Saeedi, I. L. Markov, *Syntheisis and optimization of reversible circuits.* — A Survey. `arXiv: 1110.2574v1` [cs.ET](12 Oct. 2011).

SPIIRAS, Russian Academy of Sciences,
St.Petersburg, Russia
E-mail: `SNBaranov@gmail.com`

IRIT, University of Toulouse, France
E-mail: `soloviev@irit.fr`

Поступило 12 ноября 2013 г.