Roshan James <rpjames@umail.iu.edu>

# [ICFP 2012] Rejected paper #46 "The Two Dualities of Computation:..."

**robby@eecs.northwestern.edu** <robby@eecs.northwestern.edu> Mon, May 28, 2012 at 7:15 PM
To: Roshan James <rpjames@indiana.edu>
Cc: Robby Findler <robby@eecs.northwestern.edu>

Dear Roshan James,

```
      Title: The Two Dualities of Computation: Negative and Fractional
             Types
    Authors: Roshan P. James (Indiana University)
             Amr Sabry (Indiana University)
```

I'm sorry to inform you that your paper has been rejected from ICFP 2012.

The reviews for your paper are appended to this email.  I hope that you
find them useful.

Please feel free to contact with any questions or concerns.

Best regards,
Robby

```
===========================================================================
                        ICFP 2012 Review #46A
                Updated Friday 6 Apr 2012 5:26:55pm CDT
---------------------------------------------------------------------------
  Paper #46: The Two Dualities of Computation: Negative and Fractional
             Types
---------------------------------------------------------------------------

               Reviewer expertise: Z. Passing familiarity with the area

                       ===== Paper summary =====
```

The paper explores the meaning of negative values and types, as well as fractional values and types in a reversible computation model (a model where no information is lost and computation can run backward). The negative and fractional types arise naturally as inverses of sum and products, such that b + (-b) = 0, i.e. the empty type, and b x (1/b) = 1, i.e., the unit type. Full syntax and small-step semantics are given for this language and a graphical wiring notation is used to introduce ever more complex examples. The semantics of negative types and values turns out to be a form of backward flow, whereas fractional values are used as a form of constraints along with unification. The paper shows how negative types can be used to construct functions and delimited continuations, and fractional types can be used as communication channels. The paper builds up to a formulation of a Sat solver written in this formalism.

===== Comments for author =====

This paper is great fun to read. It opened my eyes to a different universe that seems exotic and interesting. Whether of this work has any future impact on the direction of computation/type systems, I cannot tell at this point, but the formulation of the language PI\eta\eps and the illustrative examples are a gem.

As a complete outsider to this area, I was able to follow the paper easily, which attests to the quality of the presentation.

From a technical point, one thing I didn't get that seems fishy is on page 9, where the construction of iso_f requires that the heap has appropriate values. First, it is unclear what the appropriate values are. Second, and more importantly, to perform this kind of check, it seems that we would need the same construction again (ie. what we are doing for iso_f), meaning that it requires turtles all the way down. It would be great to clear this up in the final version.

The other point I'd like to see addressed by the authors is what tangible impact they see negative and fractional types as potentially having in the area of functional programing and type systems, beyond mathematical curiosity. E.g., in the introduction, you state that negative and fractional types ought to be part of the vocabulary of every programmer. The paper does not do that justice in that I cannot go down the hallway and convince my fellow programmers that they need fractional and negative types at this point.

Details:

page 3, right:
 - the graphical notation on sum types doesn't mirror the forms shown by the paragraph on product types in that it shows on the right the forms for the value on the left or the right, whereas to be consistent, the first line should show a value of b1+b2 flowing on a single wire.

page 5, left:
 - involution: I had to look this up, defining it will require only a few words or a footnote.

page 6, left:
 - you use a box labeled "id". It appears to mean that you transform the single wire of a pair/sum into the 2 wire format. This is not explained.
 - In other (+words) c: b1 ...   (words is missing)

page 6, right:
 - You should remind the reader at this point what the linear function types stand for. They were briefly introduced on page 2 in the introduction.

page 8, left:
 - 4.3, point a), creates two values 1/v and 1/v. Copy paste error? The second 1/v should be v, not?

page 10, left:
 - The Lambek-Grishin (+calculus) exhibits ...

page 10, right:
 - however even more exciting tha(n/t) plain ...

page 11, left:
 - that we can extend(-ed) \Pi...

===========================================================================
ICFP 2012 Review #46B
Updated Wednesday 16 May 2012 12:59:32am CDT
---------------------------------------------------------------------------
Paper #46: The Two Dualities of Computation: Negative and Fractional Types
---------------------------------------------------------------------------

Reviewer expertise: X. Expert

===== Paper summary =====

A programming language equipped with two dualities is proposed. The duality for

sum types, expressed by negative types, is used for the backward flow of
information (similar to the particle-style GoI), whereas the duality for
product types, expressed by fractional types, is for the flow of constraints
(which is similar to the wave-style GoI). The authors describe its categorical
semantics, operational semantics and some programming examples.

While the approach taken here is novel and interesting, I think that there is a
serious technical problem in this paper and I recommend rejection.

===== Comments for author =====

I think that this work is an interesting attempt of combining (in GoI
terminology) the particle-style computation and the wave-style computation in
a single programming language. It would be wonderful if we could have such a
language, even in some restricted form (like the one proposed here).

However, I believe that the setting considered here is somewhat too rich to be
consistent. In particular, I think that there is no "natural" categorical model
for this language, as the following exercises on compact closed categories show:

Lemma. Suppose that (C,\times,1,1/-) is a compact closed category with an
object 0 and a natural isomorphism d_X:X\otimes 0 -> 0. Then 0 is a zero
object in C.

Proof: For any object X, we have morphisms between X and 0:

```
 X <--> 0\times (X/0) <---> 0
```

So 0 is a weak zero object. To see that it is actually zero, for any f:X->0,
we have:

```
f = (* by compact closedness *)
   (id\times\eta);(c\times id);(id\times f\times id);(\epsilon\times id)
 = (* since d is iso *)
   (id\times\eta);(c\times id);(id\times ((f\times id);d_0;d_0^{-1}));(\epsilon\times id)
 = (* since d is natural *)
   (id\times\eta);(c\times id);(id\times (d_X;d_0^{-1}));(\epsilon\times id)
```

Hence such an f is unique.

(By the way, I do not understand the analogy with the field of rational numbers,
because of the presence of the fractional types X/0.)

Lemma. Any compact closed category whose unit object is a terminal object is
equivalent to a trivial category.

Proof: Easy.

By combining these two lemmas, we obtain:

Proposition. If a category has two compact closed structures (\times,1) and
(+,0) and a natural isomorphism X\times 0 -> 0, it is equivalent to a trivial
category.

It might be possible to find a non-tirivial example in which the naturality

does not hold; but giving such a model looks a fairly delicate task and far
from simple (at least for me). Instead, I think that it is better to consider
a smaller language based on a category which is compact closed for (\times,1)
and *traced* for (+,0) - I guess that this is not too far from the approach
taken in the authors' POPL2012 paper. Note that there are many such categories;
for example, the category Rel of sets and binary relations is compact closed
for products and traced for sums.

Regarding the operational semantics, I would like to see some minimum results
on its "correctness" like type soundness, progress lemma, logical reversibility
and so on, which are all missing from this paper.

In summary, although I like the ideas and approach of this paper, I think that
its technical contents, including the design of the language, should be carefully
and substantially revised.

** Comments added after having look at authors' response **

My view on this paper has not been much changed after reading
the authors' response (and other reviews). In general (like many
other reviewers) I like the ideas and approach of this work, but
I believe that further efforts should be made for making the paper
technically sound and ready for publication. Most seriously, unless
the inconsistency problem of the categorical models is solved, the
graphical presentations (string diagrams based on the categorical
semantics) used throughout the paper do not make sense at all.

In the response, the authors say that they actually want to have a
categorification of the field of rational numbers. While I agree that
that can be a potentially interesting research direction, I am less
sure if that is relevant to the technical development in this paper:
though likely to be inconsistent, the categorical semantics proposed
in this paper looks much closer to the intuition suggested by the
syntax and operational semantics of the language. I can be wrong, but
I got the impression that this language does not have a good
correspondence to a categorification of rationals.

(Also, I am not sure if a language with no conventional mathematical
semantics deserves to be said "elementary", "simple" or "natural"...)

I hope that authors have studied my proof of the inconsistency;
there is not much deep issue in it, just some basics of monoidal
categories.
One strong assumption I made in the proof is the naturality of the
isomorphisms for distributivity. However, I should say that in some
categories coming from programming languages we find interesting
*non-natural* transformations. One such example is the control
operator
C_A: ((A -> void) -> void) -> A
which cannot be natural in a non-degenerate cartesian closed category;
instead it is natural only for some special ("focal" or "central")
morphisms, but it took us years to understand this fact properly (cf.
Selinger's MSCS paper on control categories). So, though I cannot
think of a solution now, I do not deny the possibility of discovering a

non-degenerate model with non-natural distributivity, which can be
interesting. It would be great if the authors can derive a "term model"
from the syntax and operational semantics, but that might not be easy
because of the non-determinism (just like we do not yet have an agreed
notion of a term model for classical sequent calculi with
non-determistic cut-elimination).

[A possible direction is to develop a semantics for a fragment of this
language in the category Rel (of sets and binary relations, or
non-deterministic functions) first; I guess that this is relatively
straightforward, as Rel is suitable for interpreting non-determinism of
the language. Then one can study what are missing from this Rel-based
model. ]

Finally, let me add a few words on "duality", which might be of some use.
There are many different notions of duality used in mathematics, but as
far as I understand most of them find some place in the following
classification:
(1) self-duality:
  asking for a category C to be equipped with a contravariant
  equivalence. Hence C is equivalent to its opposite category C^op,
  and the duality exists on a single category.
  Duality in classical linear logic (*-autonomous categories) is such
  an instance. The dualities considered in this paper (compact closed
  categories) also fits in this group.
(2) (contravariant) equivalence between categories:
  asking for an equivalence between a category C and the opposite of
  another category D, hence C is equivalent to D^op. (Note that (1) is
  the special case with C=D.)
  Many important dualities in mathematics are of this kind, including
  Stone duality, Tannaka duality etc. The Filinski-Selinger dualtity
  between call-by-name and call-by-name is also an instance of this,
  in that a model of CbN (control category) is equivalent to the
  opposite of a model of CbV (co-control category).
(3) (contravariant) adjunction between categories:
  asking for an adjunction between a category C and the opposite of
  another category D. Since every equivalence gives rise to an adjoint
  equivalence, this is a generalization of (2). There are many such,
  including Galois connections.

===========================================================================
ICFP 2012 Review #46C
Updated Wednesday 2 May 2012 10:37:33pm CDT
---------------------------------------------------------------------------
Paper #46: The Two Dualities of Computation: Negative and Fractional
Types
---------------------------------------------------------------------------

Reviewer expertise: Z. Passing familiarity with the area

===== Paper summary =====

This paper explores negative and fractional types a-b and a/b which have properties that closely resemble those of rational numbers: a-b is 0, the uninhabited type, and a/a is 1, the unit type. The authors argue that in an information-preserving computational setting where computations can run in reverse, negative types can be seen as being inhabited by values that "flow backwards". On the other hand, the multiplicative inverse 1/a of a corresponds to a constraint, namely that a value of type a must be furnished at some point.
The paper describes the "core reversible language" which comes with ordinary sum and product types and then

extends it with negative and fractional types together with corresponding negation and inverse operations. The resulting system is discussed both in terms of a graphical representation and the corresponding monoidal categories.
The paper concludes with an exciting look towards adding recursion, which seems to naturally lead to types that correspond to algebraic and even complex numbers.

===== Comments for author =====

I spent a good deal of time trying to understand this paper. Everything in here seems exciting for the sheer beauty of its construction and the mystery of the cross-connections with numbers.
However, I have to confess that I failed: the paper did not succeed in providing me with an intuition for what's going on here. Perhaps that's just the nature of the beast and an expression of my own limitations. On the other hand, maybe there is a better way of introducing this material to an informed outsider such as myself.

As exciting as the paper is to me, I am afraid that it might just be somewhat out of sync with the rest of the conference and, therefore, of limited interest.

One sentence in the outline of the paper I found amusing: "We develop programming intuition and argue that negative and fractional types ought to be part of the vocabulary of EVERY programmer." (Emphasis mine.) Well, having recently been in a conversation with "real" programmers who tried to argue that knowledge of RECURSION is not necessary to be a successful programmer (a point of view that I obviously vehemently disagree with), I fear that the authors are just a little bit out of touch with the notion of "every programmer" here.

In conclusion, I am fascinated by this topic. The paper in its current form, however, merely piqued my curiosity and did not succeed in helping me to develop an intuition.

===========================================================================
ICFP 2012 Review #46D
Updated Friday 4 May 2012 7:44:05am CDT
---------------------------------------------------------------------------
Paper #46: The Two Dualities of Computation: Negative and Fractional Types
---------------------------------------------------------------------------

Reviewer expertise: Y. Knowledgeable outsider

===== Paper summary =====

This paper studies the reversible language \Pi extended with negative and fractional types -- both of which have a "continuation-based" interpretation.

Being reversible, \Pi programs are built using (type) isomorphisms. The paper first presents the basics of Pi: its values, types, the combinators to build the isomorphisms, as well as a (categorily founded) graphical model of the language and an operational semantics in the form of an abstract machine.

Second the paper goes on to describe the two extensions, which are given similar treatment. Specifically, negative types are interpreted as reversing the control flow, whereas fractional types are given logical unification-based interpretation.

Third the paper illustrates the expressibility of the developed language by encoding a SAT solver.

Throughout the paper provides an extensive discussion of related work from the literature.

This is a well-written, well-structured, and thought-provoking
submission.

The paper builds on

- the author's recent POPL'12 paper on the reversible language Pi, and

- a vast amount of other previous work (category theory, continuations, duality, operational semantics, GoI, ...), which bears the risk of being too demanding of a reader.

I don't find that to be the case though. I think the average ICFP
attendee is likely to read and understand (a majority of) the paper
without problems.

Since the language and the extensions are given

- a (graphically illustrated) categorical model as well as
- an operational semantics in the form of an abstract machine,

I would however have expected to see some formal indication that the two
models agree.

One other related line of work (I know: there are many) which is not
discussed in the submission is Galois connections (as found in
abstract interpretation and the program calculation community). Being
generalizations of isomorphisms, several concepts from the submission
can be explained in terms of these, e.g.,

- the combinator pairs serve as lower and upper adjoints, respectively, and
- the reverse sequential combinator, has a natural explanation as the upper adjoint of two serially composed Galois connections.

On that train of thoughts, Oliviera et al's work on Galculator -- a
Galois connection (and hence isomorphism)-based theorem prover seems
related, especially the combinator-based DSL of said system.

The authors mention a forthcoming Haskell implementation in the
submission, which would give a nice practical angle to back up this
language. However at this point as readers we have to take the authors
on faith.

Finally a few less important points:

- The new negative and fractional constructions are given classical inference type rules on p.5, col.2, but the submission contains no such system connecting the values and types of the base language \Pi.

- The concept of an 'involution' (first occurring on p.5) is not introduced neither explained.

Overall I found the paper to be an inspiring read, which has potential
to spur discussion, out-of-the-box thinking, and further follow-up
work.

-----

Detailed comments:

```
 - p.6, col.1:
   ..., we can build isomorphisms that the dual operation is an involution.
  -->
   ..., we can build isomorphisms such that the dual operation is an involution.


 - p.6, col.1 (note: two corrections)
   In other words c : ... maps to neg c : ... in the additive monoid and to inv
   c : ... in the multiplicative monoid.
  -->
   In other c : ... to neg c : ... in the additive monoid and to inv
   c : ... in the multiplicative monoid.


 - p.7, col.2:
   ... can be seen to suggest either a unified or separate implementations.
  -->
   ... can be seen to suggest either unified or separate implementations.


 - p.8, col.1:
   As explained previously \eta^x creates two values 1/v and 1/v ...
  -->
   As explained previously \eta^x creates two values 1/v and v ...


 - p.9, col.2:
   The SAT-solver is is completed by tracing ...
  -->
   The SAT-solver is completed by tracing ...


 - p.10, col.2:
   ...; arbitrary datatypes are however even more exiting that plain rationals...
  -->
   ...; arbitrary datatypes are however even more exiting than plain rationals...


 - p.11 (make sure BibTeX preserves upper-case F):
   [13] ... Interpretation of system f ...
 -->
   [13] ... Interpretation of system {F} ...
```

```
===========================================================================
                          ICFP 2012 Review #46E
               Updated Thursday 17 May 2012 4:31:46am CDT
---------------------------------------------------------------------------
  Paper #46: The Two Dualities of Computation: Negative and Fractional
             Types
---------------------------------------------------------------------------
```

Reviewer expertise: X. Expert

===== Paper summary =====

In logic, De Morgan duality relates conjunction to disjunction,

```
 ~(x & y)  =  (~x) \/ (~y)
 ~~x       =  x
```

This has direct relevance to type systems for lambda calculus, since through the lens of Curry-Howard, conjunction becomes product (tuples) and disjunction becomes sums (variants). Indeed, in a programming language with call/cc we can show

```
 ~(x * y)  =  (~x) + (~y)
 ~~ x      =  x
```

taking ~x = x -> 0, where 0 is the empty type. One line of work shows that this form of duality interchanges call-by-name and call-by-value, where for call-by-name we define x -> y = (~x) \/ y, and for call-by-value we define x -> y = ~(x * (~y)). Other notions of duality also appear, more or less closely related to logic and the interpretation given here.

The current paper shows that in the context of reversible computation there exist *two* notions of duality:

```
 -(x + y)   =  (-x) + (-y)
 --x        =  x

 1/(x * y)  =  (1/x) * (1/y)
 1/(1/x)    =  x
```

And further introduces *two* notions of function, where we define x ->^+ y = (-x) + y and x ->^* y = (1/x) * y. This differs from the previous notion, in that it does not interchange conjunction with disjunction. Roughly speaking, the type (-x) corresponds to execution `backward in time' while the type (1/x) corresponds to imposing constraints that are resolved via unification and backtracking.

The new notion of duality differs significantly from the previous one, and is not entirely clear how they relate---to this reviewer, it is not clear that the relation is anything stronger than a superficial resemblance in form. Nonetheless, both are of interest for the same underlying reason: symmetry is a powerful concept, and is worth pursuing whereever it is found. For this reason I recommend acceptance.

I read the paper fairly carefully and spotted no significant errors. The paper is well written and the central ideas are clearly explained. I found it particularly interesting that from a small number of primitives, corresponding to the laws

```
 x + (-x) + x   =  x
 x * (1/x) * x  =  x
```

(which arise in standard category-theory contexts related to quantum theory), one could derive other interesting operators and laws, including two versions of trace (a looping or fixpoint operator, well

known in category theory) and the duality and involution laws for +
and * given above (in the paragraph that mentions '*two* notions of
duality').

I believe the paper would be stronger if it downplayed the relationship
to De Morgan duality. In De Morgan, conjunction and disjunction are
dual to each other, here + and * are (independently) dual to themselves.
While the relationship here may in future be seen to be connected with
De Morgan duality, that is far from obvious and not justified by the
material in the paper. Therefore, I believe that lines like the
ones quoted below should be deleted. The material in Section 6 should
be retained, but listed as a review of other notions of duality that
have appeared in the literature, removing any claims that those notions
of duality are necessarily related to the ones that appear here.

Examples of overstrong claims:

(p.1, first paragraph of introduction) "Although these types have
previously appeared in the literature (see Sec. 6), they have
typically appeared in the context of conventional languages with
information effects, which limited their appeal and obscured their
properties."

(p.2, last paragraph of introduction) "We relate our notions of
negative and fractional types to previous work on
continuations. Briefly, we argue that conventional continuations
conflate negative and fractional components. This observation allows
us to relate two apparently unrelated lines of work: the first
pioneered by Filinski [11] relating continuations to negative types
and the second [4] relating continuations to the fractional types of
the Lambek-Grishin calculus. (Sec. 6)"

Note: While I'm not convinced of a strong connection to continuations
or De Morgan duality, there may be stronger connections to
Lambek-Grishin calculus or to quantum theory. I'm happy to see all
possible connections listed, but I believe more evidence is required
before making any strong claim of connection. Instead, the question
of the connections can be left as future work.

I list my expertise as X. I am expert in De Morgan duality, but only
a knowledgeable outsider in reversible computation or quantum
computing.

Additional comments after reading Review B and author response
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

Review B raises a valid question about categorical semantics for a
system such as the one the authors present. However, given that the
authors present an implementation, I think it is reasonable to treat
the lack of categorical semantics as raising a question for research
rather than showing that the system is contradictory or nonsense. I
still favour accepting the paper.

These views are based on the assumption that the categorical collapse
spotted by Referee B depends on division by zero. If Referee B knows
of other ways to collapse the system, further discussion may be
required.
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

There are consequences of the point raised by Review B:

* The authors must make it formally clear that division by zero is not
allowed, rather than leaving this implicit.

* They should include in their paper the argument given by Review B,
to show the consequences of not ruling out division by zero.

* They should note the difficulties of a categorical semantics and
raise this as an issue for future research.

* Referee B's point also reinforces my belief that the authors should
refrain from strong claims about their system. I see this paper more
as raising a question than providing an answer.

===== Comments for author =====

The analogy of the operators described here to a rational field is
highly suggestive. However, rational fields prohibit division by
zero, which appears to be permitted here. It would be helpful to
bring attention to this difference.

Typically, a development like the one presented here would include
a statement of type safety, such as preservation and progress for
small step semantics. Either include such a statement, or note its
absence as an area for future work.

In the section on expressing a SAT-solver in the language, it would be
helpful to say something about complexity---I presume the backtracking
required to implement fractional types means that it requires
exponential time.

Typo.

page 7, column 1, line 14: '(-b)+b)' has an extra close paren.

===========================================================================
ICFP 2012 Review #46F
Updated Wednesday 23 May 2012 9:59:57am CDT
---------------------------------------------------------------------------
Paper #46: The Two Dualities of Computation: Negative and Fractional
Types
---------------------------------------------------------------------------

Reviewer expertise: Y. Knowledgeable outsider

===== Comments for author =====

Below is a summary of the major technical problem of the paper, discussed during the PC meeting.

1. By definition of the syntax, types like b/0 (= b x 1/0) clearly exist, from which one can prove that the
categorical semantics collapse to a trivial one (as in Review B).

2. It is probably true (though not proved) that _values_ of types like b/0 do not exist, but that does not affect the
point 1 above.

3. If one (somehow) excludes types like b/0, then it would be hard (if not impossible) to establish the dualities

(for example, what is the multiplicative dual of 0, then?), which are the whole point of the paper.

4. The operational semantics might mean something (it is implemented, at least!) but _nothing_ is formally proved about it.

We hope that the authors find this information useful for further research.