JOIN INVERSE CATEGORIES AND REVERSIBLE RECURSION

Nordic Workshop on Programming Theory 2015

Robin Kaarsgaard

October 21, 2015

DIKU, Department of Computer Science, University of Copenhagen
robin@di.ku.dk
http://www.di.ku.dk/~robin

WHO?

- Robin Kaarsgaard, PhD student at DIKU, Dept. of Computer Science, University of Copenhagen.
- Project: Logical Methods in Reversible Computing (category theory, type theory, logic, ...) – Dec. 2014 to Dec. 2017 (expected).
- Jointly advised by Robert Glück, Holger Bock Axelsen, Andrzej Filinski.

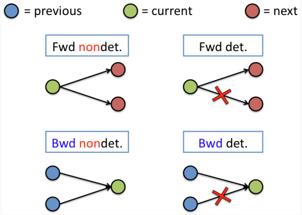
OVERVIEW

1. Reversible computing: What? Why?
2. Reversible functional programming
   - RFUN
   - Theseus (and Π^0)
3. Join inverse categories and reversible recursion
4. Concluding remarks

REVERSIBLE COMPUTING: WHAT? WHY?

WHAT IS REVERSIBLE COMPUTING?

- Reversible computing: The study of time invertible computations.
- Deterministic in both forward and backward directions.



- In a functional programming setting, reversible functions are injective.
- Note that totality is not required, nor necessarily desirable, in order to guarantee reversibility.

WHY REVERSIBLE COMPUTING?

- Originally motivated by the potential to reduce power consumption of computing processes, due to Landauer's principle: Irreversibility costs energy.
- Has since seen a number of applications independent of this property; personal favorites include
  - unified parser/pretty printer specifications and
  - fast parallel discrete event simulations.
- Plays an important role in quantum computing.

R. Landauer, "Irreversibility and heat generation in the computing process," *IBM Journal of Research and Development*, vol. 5, no. 3, pp. 261–269, 1961.

T. Rendel and K. Ostermann, "Invertible syntax descriptions: unifying parsing and pretty printing," *ACM SIGPLAN Notices*, vol. 45, no. 11, pp. 1–12, 2010.

M. Schordan, D. Jefferson, P. Barnes, *et al.*, "Reverse code generation for parallel discrete event simulation," in *Reversible Computation*, ser. LNCS, vol. 9138, 2015, pp. 95–110.

REVERSIBLE FUNCTIONAL PROGRAMMING

RFUN

$$
\begin{aligned}
&\mathit{fib}\ n \triangleq \textbf{case}\ n\ \textbf{of} \\
&\qquad Z \quad\ \ \to \langle S(Z), S(Z)\rangle \\
&\qquad S(m) \to \textbf{let}\ \langle x, y\rangle = \mathit{fib}\ m\ \textbf{in} \\
&\qquad\qquad\quad\ \textbf{let}\ z = \mathit{plus}\ \langle y, x\rangle\ \textbf{in}\ z
\end{aligned}
$$

$$
\begin{aligned}
&\mathit{plus}\ \langle x, y\rangle \triangleq \textbf{case}\ y\ \textbf{of} \\
&\qquad Z \quad\ \ \to \lfloor\langle x\rangle\rfloor \\
&\qquad S(u) \to \textbf{let}\ \langle x', u'\rangle = \mathit{plus}\ \langle x, u\rangle\ \textbf{in}\ \langle x', S(u')\rangle
\end{aligned}
$$

- Untyped first-order reversible functional programming language.
- Patterns are linear: All variables defined by a pattern must be used *exactly once*.
- Results of all function calls must be bound in a `let`-expression.

T. Yokoyama, H. B. Axelsen, and R. Glück, “Towards a reversible functional language,” in *Reversible Computation*, ser. LNCS, vol. 7165, 2012, pp. 14–29.

RFUN: RECURSION

- Recursion in RFUN is based on a call stack, as in irreversible functional programming.
- Recursive functions are inverted by inverting the body of the `let`, and replacing the recursive call with a call to the inverse.

T. Yokoyama, H. B. Axelsen, and R. Glück, “Towards a reversible functional language,” in *Reversible Computation*, ser. LNCS, vol. 7165, 2012, pp. 14–29.

THESEUS AND Π^0

```
treeUnwindf :: f:(Nat ↔ a) → Tree ↔ Tree * Tree + a
| Node t1 t2 ↔ Left (t1, t2)
| Leaf n     ↔ Right (f n)
```

- Typed first-order reversible functional programming language
- Supports parametrized maps, maps depending on other maps given at *compile time*.
- Patterns are linear *and exhaustive*, all functions are total.
- Compiles to the reversible combinator calculus Π^0.

W. J. Bowman, R. P. James, and A. Sabry, "Dagger traced symmetric monoidal categories and reversible programming," in *Reversible Computation*, ser. LNCS, vol. 7165, 2011, pp. 51–56.

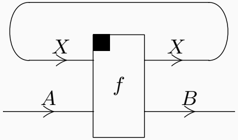
R. P. James and A. Sabry, "Theseus: A high level language for reversible computing," Work-in-progress report presented at Reversible Computation, 2014.

THESEUS AND Π^0: RECURSION VIA †-TRACE

- Recursion in Theseus (indirectly) and Π^0 (directly) is implemented via a reversible trace operator

$$\texttt{trace} : \texttt{a} + \texttt{x} \leftrightarrow \texttt{b} + \texttt{x} \;\rightarrow\; \texttt{a} \leftrightarrow \texttt{b}$$

- This is a trace in the categorical sense of traced monoidal categories (in fact, a †-trace).



P. Selinger, "A survey of graphical languages for monoidal categories," *Lecture Notes in Physics*, vol. 813, pp. 289–355, 2011.

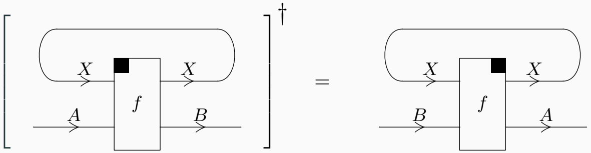
R. P. James and A. Sabry, "Theseus: A high level language for reversible computing," Work-in-progress report presented at Reversible Computation, 2014.

THESEUS AND Π^0: RECURSION VIA †-TRACE

- Recursion in Theseus (indirectly) and Π^0 (directly) is implemented via a reversible trace operator

$$\texttt{trace} : a + x \leftrightarrow b + x \rightarrow a \leftrightarrow b$$

- This is a trace in the categorical sense of traced monoidal categories (in fact, a †-trace).



P. Selinger, "A survey of graphical languages for monoidal categories," *Lecture Notes in Physics*, vol. 813, pp. 289–355, 2011.

R. P. James and A. Sabry, "Theseus: A high level language for reversible computing," Work-in-progress report presented at Reversible Computation, 2014.

JOIN INVERSE CATEGORIES AND REVERSIBLE RECURSION

MOTIVATION

- Wanted: Categorical model rich enough to capture...
  - *partial* injective functions (RFUN isn't total), and
  - the two *distinct* notions of reversible recursion from RFUN and Theseus
- Starting point: Giles' investigation of inverse categories as models of reversible functional programming.
- Inverse categories: Special case of restriction categories, categories with partiality.

B. G. Giles, "An investigation of some theoretical aspects of reversible computing," PhD thesis, University of Calgary, 2014.

INVERSE CATEGORIES

- A restriction category is a category where each $f : A \to B$ has a restriction idempotent $\overline{f} : A \to A$ (subject to axioms such as $f \circ \overline{f} = f$, and others).
- Partial order enriched; for parallel morphisms f and g,

$$f \leq g \quad \text{iff} \quad g \circ \overline{f} = f$$

- Partial isomorphism: A morphism $f : A \to B$ with a partial inverse $f^\dagger : B \to A$ such that $f^\dagger \circ f = \overline{f}$ and $f \circ f^\dagger = \overline{f^\dagger}$.
- Inverse category: Restriction category with only partial isomorphisms.

J. R. B. Cockett and S. Lack, “Restriction categories i: categories of partial maps,” *Theoretical Computer Science*, vol. 270, no. 2002, pp. 223–259, 2002.

JOIN INVERSE CATEGORIES

An inverse category is a join inverse category if it has

- a *restriction zero*, specifically all *zero morphisms* $0_{A,B} : A \to B$,
- a partial operation $\bigvee$ on all compatible subsets of all hom-sets, satisfying

  $$g \leq \bigvee_{f \in F} f \ \text{ if } g \in F\text{, and if } f \leq h \text{ for all } f \in F \text{ then } \bigvee_{f \in F} f \leq h$$

  and other axioms.
- We consider inverse categories with joins of countable sets.

X. Guo, "Products, joins, meets, and ranges in restriction categories," PhD thesis, University of Calgary, 2012.

JOIN INVERSE CATEGORIES AS CPO-CATEGORIES

- Observation: The underlying sets for all ω-chains are compatible.
- Idea: Given an ω-chain $\{f_i\}_{i\in\omega}$, define $\sup\{f_i\}_{i\in\omega} = \bigvee_{i\in\omega} f_i$.
- Consequence (by Kleene's fixed point theorem): Every monotone and continuous morphism scheme of the form $f : \mathrm{Hom}_{\mathscr{C}}(A,B) \to \mathrm{Hom}_{\mathscr{C}}(A,B)$ has a least fixed point $\mathrm{fix}\, f : A \to B$.
    - Morphism schemes in general look a whole lot like parametrized maps à la Theseus...

JOIN INVERSE CATEGORIES AS CPO-CATEGORIES

- Insight: The family of morphism schemes defined by $\mathsf{inv}_{A,B}(f) = f^\dagger$ is monotone, continuous, and an isomorphism with inverse $\mathsf{inv}_{B,A}$ in each component.
- Every monotone and continuous morphism scheme of the form $f : \mathrm{Hom}_{\mathscr{C}}(A,B) \to \mathrm{Hom}_{\mathscr{C}}(A,B)$ has a *fixed point adjoint* $f_\ddagger : \mathrm{Hom}_{\mathscr{C}}(B,A) \to \mathrm{Hom}_{\mathscr{C}}(B,A)$ such that $(\mathrm{fix}\, f)^\dagger = \mathrm{fix}\, f_\ddagger$.
    - Trick: Define $f_\ddagger = \mathsf{inv}_{A,B} \circ f \circ \mathsf{inv}_{B,A}$.
- This is precisely recursion à la RFUN!

JOIN INVERSE CATEGORIES AS UNIQUE DECOMPOSITION CATEGORIES

- Unique decomposition categories (UDCs) are categories with...
  - a partial sum operator Σ on countable families of parallel morphisms, and
  - a sum-like monoidal tensor $\cdot \oplus \cdot$

  both subject to certain axioms.
- Result (Haghverdi): Given the existence of certain sums, UDCs have a (uniform) trace.
- Idea: Define $\sum_{i \in I} f_i = \bigvee_{i \in I} f_i$, and get the sum-like monoidal tensor via a join-preserving disjointness tensor (Giles).

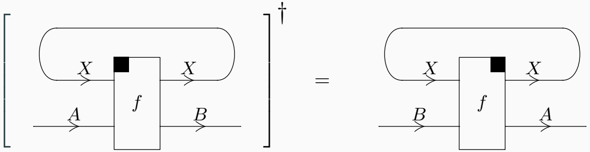
E. Haghverdi, "A categorical approach to linear logic, geometry of proofs and full completeness," PhD thesis, Carlton University and University of Ottawa, 2000, pp. 1–239.

B. G. Giles, "An investigation of some theoretical aspects of reversible computing," PhD thesis, University of Calgary, 2014.

JOIN INVERSE CATEGORIES AS UNIQUE DECOMPOSITION CATEGORIES

- Result: Not just a trace operator, but one satisfying the †-trace condition

$$\mathrm{Tr}^X_{A,B}(f)^\dagger = \mathrm{Tr}^X_{B,A}(f^\dagger)$$



for all $f : A \oplus X \to B \oplus X$.

- Reversible recursion à la Theseus and Π^0!

P. Selinger, “A survey of graphical languages for monoidal categories,” *Lecture Notes in Physics*, vol. 813, pp. 289–355, 2011.

CONCLUDING REMARKS

WHAT I DID NOT COVER

- All of the gory details!
    - A few more are in the abstract – for the rest, just ask!
- Using Adámek's fixed point theorem, Guo's join completion theorem, and a few lemmas, we can also show faithful embedding in algebraically ω-compact category: This models isorecursive data types à la Theseus.

CONCLUSION

- By viewing join inverse categories as CPO-categories, we get
    - fixed points of morphism schemes, modelling reversible recursion à la RFUN.
- Additionally assuming the existence of a join-preserving disjointness tensor, we get
    - a †-trace operator for modelling reversible tail recursion à la Theseus and Π^0.
- Next up:
    - Use these insights to inform language design.
    - Compact closed inverse categories – relation to partiality in quantum computing?
    - Suggestions? Talk to me!

Thank you!