Naturality, but not as you know it

Ralf Hinze

Department of Computer Science, University of Oxford
Wolfson Building, Parks Road, Oxford, OX1 3QD, England
ralf.hinze@cs.ox.ac.uk
http://www.cs.ox.ac.uk/ralf.hinze/

October 2013

Joint work with Fritz Henglein

0 Recap: Key-based sorting & searching

- *Idea:* employ the structure of sort keys *directly*.
- A hierarchy of operations:

$$
\begin{aligned}
&\mathit{sort} &&:: \mathit{Order}\,k \to [\,k \times v\,] \to [\,v\,] \\
&\mathit{discr} &&:: \mathit{Order}\,k \to [\,k \times v\,] \to [\,[\,v\,]\,] \\
&\mathit{trie} &&:: \mathit{Order}\,k \to [\,k \times v\,] \to \mathsf{Trie}\,k\,[\,v\,]
\end{aligned}
$$

- Keys and satellite data, ie values, are separated.
- An element of $\mathit{Order}\,K$ *represents* an order over the type K:

$$
\begin{aligned}
&\mathbf{data}\ \mathit{Order} :: * \to * \ \mathbf{where} \\
&\quad \mathit{OUnit} :: \mathit{Order}\,() \\
&\quad \mathit{OSum} :: \mathit{Order}\,k_1 \to \mathit{Order}\,k_2 \to \mathit{Order}\,(k_1 + k_2) \\
&\quad \mathit{OProd} :: \mathit{Order}\,k_1 \to \mathit{Order}\,k_2 \to \mathit{Order}\,(k_1 \times k_2) \\
&\quad \ldots
\end{aligned}
$$

0 Structure of correctness proofs

- Show that *sort* is correct.
- Relate *sort* and *discr*:

$$concat \cdot discr\ o = sort\ o$$

- This is nontrivial as *discr* and *sort* have different algorithmic strategies: MSD versus LSD.
- Relate *discr* and *trie*:

$$discr\ o = flatten \cdot trie\ o$$

- This is straightforward.

1 Recap: Naturality

- $\mathit{reverse} :: [\,a\,] \to [\,a\,]$ is a natural transformation:

  $$\mathit{map}\,h \cdot \mathit{reverse} = \mathit{reverse} \cdot \mathit{map}\,h$$

  for all $h :: A \to B$.
- (Parametricity implies naturality.)
- (We work in **Set**.)

1 Recap: Naturality

- Given $magic :: [a] \to [a]$ with

$$map\,h \cdot magic = magic \cdot map\,h$$

  for all $h :: A \to B$.
- What do we know about $magic$?

1 Recap: Naturality

- $magic :: [a] \to [a]$ is fully determined by its $\mathbb{N}$ instance.

$$
\begin{aligned}
& magic\ xs \\
= & \quad \{ \text{ introduce } ix : \mathbb{N} \to A \text{ so that } map\ ix\ [1 \mathinner{..} n] = xs \ \} \\
& magic\ (map\ ix\ [1 \mathinner{..} n]) \\
= & \quad \{ \ magic \text{ is natural } \} \\
& map\ ix\ (magic\ [1 \mathinner{..} n])
\end{aligned}
$$

1 Recap: Naturality

- Say, we suspect that $\mathit{magic} = \mathit{reverse}$.
- It suffices to show that $\mathit{magic}\,[1\,..\,n] = \mathit{reverse}\,[1\,..\,n]$.

$$
\begin{aligned}
& \mathit{magic}\ \mathit{xs} \\
= \quad & \{\ \text{see above}\ \} \\
& \mathit{map}\ \mathit{ix}\ (\mathit{magic}\,[1\,..\,n]) \\
= \quad & \{\ \text{proof obligation}\ \} \\
& \mathit{map}\ \mathit{ix}\ (\mathit{reverse}\,[1\,..\,n]) \\
= \quad & \{\ \mathit{reverse}\ \text{is natural}\ \} \\
& \mathit{reverse}\ (\mathit{map}\ \mathit{ix}\,[1\,..\,n]) \\
= \quad & \{\ \text{definition of}\ \mathit{ix}\ \} \\
& \mathit{reverse}\ \mathit{xs}
\end{aligned}
$$

1 Recap: Naturality

- What about $magic\,[\,] = [\,]$?
- (Intuitively, $magic :: [a] \to [a]$ *can*
    - rearrange elements,
    - delete elements,
    - duplicate elements,
- but it *cannot*
    - create elements.)

2 Strong naturality

- $reverse :: [a] \to [a]$ satisfies a stronger property:

$$filter\ p \cdot reverse = reverse \cdot filter\ p$$

for all $p :: A \to \mathsf{Maybe}\ B$.

- (You may want to view p as a partial function.)
- $filter$ combines mapping and filtering.

$$\begin{array}{l}
filter :: (a \to \mathsf{Maybe}\ b) \to ([a] \to [b]) \\
filter\ p\ [\,] \qquad\quad = [\,] \\
filter\ p\ (x : xs) = \mathbf{case}\ p\ x\ \mathbf{of} \\
\qquad\qquad\qquad\qquad\quad Nothing \to \qquad filter\ p\ xs \\
\qquad\qquad\qquad\qquad\quad Just\ y \quad\ \to y : filter\ p\ xs
\end{array}$$

- (Also called $mapMaybe$.)

2 Strong naturality

- Given $magic :: [\,a\,] \to [\,a\,]$ with

  $$filter\,p \cdot magic = magic \cdot filter\,p$$

  for all $p :: A \to \mathsf{Maybe}\,B$.
- What do we know about $magic$?

2 Strong naturality

- What about $\mathit{magic}\,[\,] = [\,]$?
- Let $\varnothing$ be the totally undefined function, $\varnothing\, a = \mathit{Nothing}$, then

$$
\begin{aligned}
& \mathit{magic}\,[\,] \\
= {} & \quad \{\ \text{property of } \mathit{filter}\ \} \\
& \mathit{magic}\,(\mathit{filter}\,\varnothing\,[\,]) \\
= {} & \quad \{\ \mathit{magic} \text{ is strongly natural}\ \} \\
& \mathit{filter}\,\varnothing\,(\mathit{magic}\,[\,]) \\
= {} & \quad \{\ \text{property of } \mathit{filter}\ \} \\
& [\,]
\end{aligned}
$$

2 Permutation

- If $magic :: [\,a\,] \to [\,a\,]$ additionally satisfies

  $$magic\,[\,x\,] = [\,x\,]$$

  then it permutes its input!
- (For simplicity, we only consider inputs with no repeated elements.)

2 Permutation

- If $magic :: [\,a\,] \to [\,a\,]$ additionally satisfies

$$magic\,[\,x\,] = [\,x\,]$$

  then it permutes its input!
- (For simplicity, we only consider inputs with no repeated elements.)
- Let $\wr x \int$ be the partial function that maps x to x and is undefined otherwise, $\wr x \int\, a = \mathbf{if}\ x == a\ \mathbf{then}\ \mathit{Just}\ x\ \mathbf{else}\ \mathit{Nothing}$.
- Definition: $perm :: [\,a\,] \to [\,a\,]$ permutes its input if

$$\mathit{filter}\,\wr x \int \cdot\, perm = \mathit{filter}\,\wr x \int$$

  for all $x :: a$.

2 Permutation

- Here is the proof:

$$
\begin{aligned}
& \mathit{filter}\,\wr i \wr\,(\mathit{magic}\,[1\,..\,n]) \\
= \quad & \{\ \mathit{magic} \text{ is strongly natural }\} \\
& \mathit{magic}\,(\mathit{filter}\,\wr i \wr\,[1\,..\,n]) \\
= \quad & \{\ \text{definition of } \mathit{filter}\ \} \\
& \mathit{magic}\,(\mathbf{if}\ 1 \leqslant i \leqslant n\ \mathbf{then}\ [i]\ \mathbf{else}\ [\,]) \\
= \quad & \{\ \text{conditionals}\ \} \\
& \mathbf{if}\ 1 \leqslant i \leqslant n\ \mathbf{then}\ \mathit{magic}\,[i]\ \mathbf{else}\ \mathit{magic}\,[\,] \\
= \quad & \{\ \mathit{magic}\,[\,] = [\,] \text{ and assumption } \mathit{magic}\,[i] = [i]\ \} \\
& \mathbf{if}\ 1 \leqslant i \leqslant n\ \mathbf{then}\ [i]\ \mathbf{else}\ [\,] \\
= \quad & \{\ \text{definition of } \mathit{filter}\ \} \\
& \mathit{filter}\,\wr i \wr\,[1\,..\,n]
\end{aligned}
$$

2 Reversal

- If $magic :: [a] \to [a]$ additionally satisfies

  $$magic\,[x, y] = [y, x]$$

  then it reverses its input!

2 Reversal

- If $magic :: [a] \to [a]$ additionally satisfies

$$magic\,[x, y] = [y, x]$$

  then it reverses its input!
- Let $\wr x, y \wr$ be the partial function that maps x to x and y to y and is undefined otherwise.
- We have

$$xs = ys \iff \forall x\, y\, .\ filter\,\wr x, y \wr\, xs = filter\,\wr x, y \wr\, ys$$

2 Reversal

- Here is the proof:

$$
\begin{aligned}
& magic\,[1\,..\,n] = reverse\,[1\,..\,n] \\
\iff & \quad \{\ \text{see above}\ \} \\
& \forall\, i\, j\, .\ filter \wr i, j \wr\, (magic\,[1\,..\,n]) = filter \wr i, j \wr\, (reverse\,[1\,..\,n])
\end{aligned}
$$

- Assume $1 \leqslant i, j \leqslant n$, then

$$
\begin{aligned}
& filter \wr i, j \wr\, (magic\,[1\,..\,n]) = filter \wr i, j \wr\, (reverse\,[1\,..\,n]) \\
\iff & \quad \{\ \text{definition of } reverse \text{ and } filter\ \} \\
& filter \wr i, j \wr\, (magic\,[1\,..\,n]) = [j, i] \\
\iff & \quad \{\ magic \text{ is strongly natural}\ \} \\
& magic\,(filter \wr i, j \wr\, [1\,..\,n]) = [j, i] \\
\iff & \quad \{\ \text{definition of } filter\ \} \\
& magic\,[i, j] = [j, i]
\end{aligned}
$$

- The other cases are similar.

3 Categorically speaking

- *map* is the arrow part of a functor $\mathsf{List} : \mathbf{Set} \to \mathbf{Set}$.
- *reverse* is a natural transformation between List and List.
- What about *filter*?

3 Categorically speaking

- *map* is the arrow part of a functor $\mathsf{List} : \mathbf{Set} \to \mathbf{Set}$.
- *reverse* is a natural transformation between List and List.
- What about *filter*?
- *filter* is the arrow part of a functor $\mathsf{Filter} : \mathbf{Set}_{\mathsf{Maybe}} \to \mathbf{Set}$.
- $\mathbf{Set}_{\mathsf{Maybe}}$ is the Kleisli category of the monad Maybe.
- (You may want to view $\mathbf{Set}_{\mathsf{Maybe}}$ as the category of partial functions.)
- The object part of Filter is just $\mathsf{Filter}\,A = [A]$.
- *reverse* is a natural transformation between Filter and Filter.

3 Properties of *filter*

- *filter* is a monoid homomorphism:

$$\begin{aligned} \mathit{filter}\, p\, [\,] &= [\,] \\ \mathit{filter}\, p\, (\mathit{xs} \mathbin{+\!\!+} \mathit{ys}) &= \mathit{filter}\, p\, \mathit{xs} \mathbin{+\!\!+} \mathit{filter}\, p\, \mathit{ys} \end{aligned}$$

- *filter* preserves identity and composition:

$$\begin{aligned} \mathit{filter}\, \mathit{id} &= \mathit{id} \\ \mathit{filter}\, (p \cdot q) &= \mathit{filter}\, p \cdot \mathit{filter}\, q \end{aligned}$$

The arguments of *filter* live in the Kleisli category $\mathbf{Set}_{\mathsf{Maybe}}$.

4 Key-based sorting: correctness

- *sort* is correct:
  - *sort o* is strongly natural:

    $$\mathit{filter}\,p \cdot \mathit{sort}\,o = \mathit{sort}\,o \cdot \mathit{filter}\,(\mathit{id} \times p)$$

  - *sort o* produces a permutation of the input values:

    $$\mathit{sort}\,o\,[\,(k, v)\,] = [\,v\,]$$

  - values are output in non-decreasing order of their keys:

    $$\mathit{sort}\,o\,[\,(a, i), (b, j)\,] = [\,i, j\,] \iff \mathit{leq}\,o\,a\,b$$

  $\mathit{leq} :: \mathit{Order}\,k \to (k \to k \to \mathbb{B})$ interprets an order representation.

4 Key-based sorting: correctness

- Relate *sort* and *discr*:
  - *discr* commutes with *strong* natural transformations: if *perm* is strongly natural, $filter\ p \cdot perm = perm \cdot filter\ (id \times p)$, then

$$map\ perm \cdot discr\ o = discr\ o \cdot perm \cdot map\ swap \ :\ [\,K \times (A \times V)\,] \to [\,[\,V\,]\,]$$

  - This is straightforward then:

$$concat \cdot discr\ o = sort\ o$$

5 Summary

- Strong naturality: an amusing twist on naturality.
- Key-based sorting: strong naturality, coupled with preserving singletons and correct sorting of two-element lists, corresponds to Henglein's *consistent permutativity*, which characterizes stable sorting functions.
- For the details see:
  Henglein, F., Hinze, R.: *Sorting and Searching by Distribution: From Generic Discrimination to Generic Tries.* In Shan, C., ed.: Proc. 11th Asian Symposium on Programming Languages and Systems (APLAS), (December 2013).

6 Generic key-based sorting

- $sort\ o$ takes a list of key-value pairs and returns the values in non-decreasing order of their associated keys.

$$
\begin{aligned}
&sort :: Order\ k \to [\,k \times v\,] \to [\,v\,]\\
&sort\ o \qquad\qquad\qquad\ \ [\,] = [\,]\\
&sort\ (OUnit) \qquad\quad rel = map\ val\ rel\\
&sort\ (OSum\ o_1\ o_2)\ rel = sort\ o_1\ (filter\ froml\ rel)\\
&\qquad\qquad\qquad\qquad\qquad +\!\!+\ sort\ o_2\ (filter\ fromr\ rel)\\
&sort\ (OProd\ o_1\ o_2)\ rel = sort\ o_1\ (sort\ o_2\ (map\ curryr\ rel))
\end{aligned}
$$

- LSD strategy:

$$
\begin{aligned}
&curryr :: (k_1 \times k_2) \times v \to k_2 \times (k_1 \times v)\\
&curryr\ ((k_1, k_2), v) = (k_2, (k_1, v))
\end{aligned}
$$

6 Generic key-based discrimination

- *discr o* returns a list of non-empty lists of values, where the inner lists group values whose keys are equivalent.

$$
\begin{array}{lll}
\mathit{discr} :: \mathit{Order}\,k \to [\,k \times v\,] \to [\,[\,v\,]\,] & & \\
\mathit{discr}\;o & [\,] & = [\,] \\
\mathit{discr}\;o & [\,(k, v)\,] & = [\,[\,v\,]\,] \\
\mathit{discr}\;\mathit{OUnit} & \mathit{rel} & = [\,\mathit{map}\;\mathit{val}\;\mathit{rel}\,] \\
\mathit{discr}\;(\mathit{OSum}\;o_1\;o_2) & \mathit{rel} & = \mathit{discr}\;o_1\;(\mathit{filter}\;\mathit{froml}\;\mathit{rel}) \\
 & & \quad +\!\!\!+\; \mathit{discr}\;o_2\;(\mathit{filter}\;\mathit{fromr}\;\mathit{rel}) \\
\mathit{discr}\;(\mathit{OProd}\;o_1\;o_2) & \mathit{rel} & = \mathit{concat}\;(\mathit{map}\;(\mathit{discr}\;o_2) \\
 & & \qquad (\mathit{discr}\;o_1\;(\mathit{map}\;\mathit{curryl}\;\mathit{rel})))
\end{array}
$$

- MSD strategy:

$$
\begin{array}{l}
\mathit{curryl} :: (k_1 \times k_2) \times v \to k_1 \times (k_2 \times v) \\
\mathit{curryl}\;((k_1, k_2), v) = (k_1, (k_2, v))
\end{array}
$$