Reasoning about Codata

Ralf Hinze

Computing Laboratory, University of Oxford
Wolfson Building, Parks Road, Oxford, OX1 3QD, England
ralf.hinze@comlab.ox.ac.uk
http://www.comlab.ox.ac.uk/ralf.hinze/

May 2009

Part 1

Prologue

1.0 Outline

1.1 What is codata?

- The dual of data, with an emphasis
- on observation rather than construction,
- process rather than value, and
- the indefinite rather than the finite.

1.2 For the mathematician . . .

- More elegant proofs through . . .
- a holistic or wholemeal approach,
- avoidance of index variables and subscripts,
- avoidance of case analysis,
- a compositional approach.

1.2 For the programmer . . .

- More elegant programs through . . .
- a holistic or wholemeal approach,
- avoidance of case analysis,
- a compositional approach,
- separation of concerns: production and termination. See also John Hughes' "Why Functional Programming Matters".

1.3 References

- J.J.M.M. Rutten. “Fundamental study — Behavioural differential equations: a coinductive calculus of streams, automata, and power series.” *Theoretical Computer Science*, (308):1–53, 2003.
- J.J.M.M. Rutten. “A coinductive calculus of streams”. *Math. Struct. in Comp. Science*, (15):93–147, 2005.
- Ralf Hinze. “Functional Pearl: Streams and Unique Fixed Points”. In Peter Thiemann, editor, *Proceedings of the 13th ACM Sigplan International Conference on Functional Programming (ICFP’08)*, September 22–24, 2008, Victoria, BC, Canada, pages 189–200. ACM Press, 2008.
- Ralf Hinze. “Scans and Convolutions — A Calculational Proof of Moessner’s Theorem”. In Sven-Bodo Scholz, editor, *Post-Proceedings of the 20th International Symposium on the Implementation and Application of Functional Languages (IFL’08)*, September 10–12, 2008, Hertfordshire, UK, *to appear*.

1.4 Overview

- Part 1: Prologue
- Part 2: Streams
- Part 3: Recurrences
- Part 4: Finite Calculus
- Part 5: Infinite Trees
- Part 6: Tabulation
- Part 7: Epilogue

Part 2

Streams

2.0 Outline

2.1 Streams

$$\langle 0, 1, 8, 27, 64, 125, 216, 343, 512, 729, 1000, 1331, \ldots \rangle$$

$$\langle 1, 3, 9, 27, 81, 243, 729, 2187, 6561, 19683, 59049, \ldots \rangle$$

$$\langle 0, 0, 1, 1, 2, 4, 3, 9, 4, 16, 5, 25, 6, 36, 7, 49, \ldots \rangle$$

$$\langle 0, 0, 2, 4, 8, 14, 24, 40, 66, 108, 176, 286, 464, 752, \ldots \rangle$$

$$\langle 0, 1, 2, 6, 15, 40, 104, 273, 714, 1870, 4895, 12816, \ldots \rangle$$

2.1 Datatype of streams

$$\mathbf{data}\ \mathsf{Stream}\ \alpha = \mathsf{Cons}\ \{\mathsf{head} :: \alpha, \mathsf{tail} :: \mathsf{Stream}\ \alpha\}$$

$$\begin{aligned}
&\mathbf{infixr}\ 5\ \prec \\
&(\prec)\quad :: \alpha \to \mathsf{Stream}\ \alpha \to \mathsf{Stream}\ \alpha \\
&a \prec s = \mathsf{Cons}\ a\ s
\end{aligned}$$

2.1 Fibonacci numbers

$$\mathrm{fib} = 0 \prec \mathrm{fib} + (1 \prec \mathrm{fib})$$

$$\begin{array}{cccccccccccccccccc}
 & 0 & 1 & 1 & 2 & 3 & 5 & 8 & 13 & 21 & 34 & \cdots & & & \mathit{fib} & & \\
 & + & + & + & + & + & + & + & + & + & + & \cdots & & & + & & \\
0 & 1 & 0 & 1 & 1 & 2 & 3 & 5 & 8 & 13 & 21 & \cdots & 0 & \prec & 1 & \prec & \mathit{fib} \\
\shortparallel & \shortparallel & \shortparallel & \shortparallel & \shortparallel & \shortparallel & \shortparallel & \shortparallel & \shortparallel & \shortparallel & \shortparallel & \cdots & \shortparallel & & & & \\
0 & 1 & 1 & 2 & 3 & 5 & 8 & 13 & 21 & 34 & 55 & \cdots & \mathit{fib} & & & &
\end{array}$$

2.2 Idioms or applicative functors

```
class Idiom $\phi$ where
  pure :: $\alpha \to \phi\ \alpha$
  ($\diamond$)  :: $\phi\ (\alpha \to \beta) \to (\phi\ \alpha \to \phi\ \beta)$
```

2.2 Streams are an idiom

```
instance Idiom Stream where
   pure $a = s$ where $s = a \prec s$
   $s \diamond t$     $= (\text{head}\ s)\ (\text{head}\ t) \prec (\text{tail}\ s) \diamond (\text{tail}\ t)$
```

2.2 Lifting

$$
\begin{aligned}
&\mathrm{repeat} && :: (\mathrm{Idiom}\ \phi) \Rightarrow \alpha \to \phi\ \alpha \\
&\mathrm{repeat}\ a && = \mathrm{pure}\ a \\[1em]
&\mathrm{map} && :: (\mathrm{Idiom}\ \phi) \Rightarrow (\alpha \to \beta) \to (\phi\ \alpha \to \phi\ \beta) \\
&\mathrm{map}\ f\ s && = \mathrm{pure}\ f \diamond s \\[1em]
&\mathrm{zip} && :: (\mathrm{Idiom}\ \phi) \Rightarrow (\alpha \to \beta \to \gamma) \to (\phi\ \alpha \to \phi\ \beta \to \phi\ \gamma) \\
&\mathrm{zip}\ g\ s\ t && = \mathrm{pure}\ g \diamond s \diamond t
\end{aligned}
$$

2.2 Arithmetic: a generic Num instance

```
instance (Idiom φ, Num α) ⇒ Num (φ α) where
  (+)            = zip (+)
  (−)            = zip (−)
  (∗ )           = zip (∗ )
  negate         = map negate    -- unary minus
  fromInteger i  = repeat (fromInteger i)
```

2.2 Natural numbers

$$\mathrm{nat} = 0 \prec \mathrm{nat} + 1$$

2.2 Natural numbers

$$\mathrm{nat} = 0 \prec \mathrm{nat} + 1$$

$$\mathrm{nat} = 0 \prec \mathrm{pure}\ (+) \diamond \mathrm{nat} \diamond \mathrm{pure}\ 1$$

$$\mathrm{nat} = 0 \prec \mathrm{map}\ (1+)\ \mathrm{nat}$$

$$
\begin{array}{ccccccccccccccc}
 & 0 & 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 & \cdots & & & \mathsf{nat} \\
 & + & + & + & + & + & + & + & + & + & + & \cdots & & & + \\
0 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & \cdots & 0 & \prec & 1 \\
\| & \| & \| & \| & \| & \| & \| & \| & \| & \| & \| & \cdots & \| & & \\
0 & 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 & 10 & \cdots & \mathsf{nat} & &
\end{array}
$$

2.2 Interactive session

$$\gg \quad \mathrm{fib}$$
$$\langle 0, 1, 1, 2, 3, 5, 8, 13, 21, 34, \ldots \rangle$$
$$\gg \quad \mathrm{nat} * \mathrm{nat}$$
$$\langle 0, 1, 4, 9, 16, 25, 36, 49, 64, 81, \ldots \rangle$$
$$\gg \quad \mathrm{fib}'^2 - \mathrm{fib} * \mathrm{fib}''$$
$$\langle 1, -1, 1, -1, 1, -1, 1, -1, 1, -1, \ldots \rangle$$
$$\gg \quad \mathrm{fib}'^2 - \mathrm{fib} * \mathrm{fib}'' == (-1)^{\mathrm{nat}}$$
$$\mathrm{True}$$
$$\gg \quad \mathrm{nat}^2$$
$$\langle 0, 1, 4, 9, 16, 25, 36, 49, 64, 81, \ldots \rangle$$

2.3 Interleaving

$\mathrm{infixr}\ 5\ \curlyvee$

$(\curlyvee) \quad :: \mathrm{Stream}\ \alpha \to \mathrm{Stream}\ \alpha \to \mathrm{Stream}\ \alpha$

$\mathrm{s} \curlyvee \mathrm{t} = \mathrm{head}\ s \prec \mathrm{t} \curlyvee \mathrm{tail}\ s$

2.3 Binary numbers

$$\mathrm{bin} = 0 \prec 2 * \mathrm{bin} + 1 \curlyvee 2 * \mathrm{bin} + 2$$

$$\mathrm{nat} = \mathrm{bin}\,?$$

2.4 Recursion and iteration

```
recurse :: (α → α) → (α → Stream α)
recurse f a = s
   where s  = a ≺ map f s

iterate  :: (α → α) → (α → Stream α)
iterate f a        = loop a
   where loop x = x ≺ loop (f x)
```

2.5 Idiom laws

$$\begin{aligned}
\mathsf{pure}\ \mathsf{id} \diamond u &= u && \text{(identity)}\\
\mathsf{pure}\ (\cdot) \diamond u \diamond v \diamond w &= u \diamond (v \diamond w) && \text{(composition)}\\
\mathsf{pure}\ f \diamond \mathsf{pure}\ x &= \mathsf{pure}\ (f\ x) && \text{(homomorphism)}\\
u \diamond \mathsf{pure}\ x &= \mathsf{pure}\ (\lambda f \to f\ x) \diamond u && \\
& && \text{(interchange)}
\end{aligned}$$

2.5 Functor laws

$$\begin{aligned} \mathrm{map}\ \mathrm{id} &= \mathrm{id} \\ \mathrm{map}\ (f \cdot g) &= \mathrm{map}\ f \cdot \mathrm{map}\ g \end{aligned}$$

2.5 Proof: $\mathsf{nat} + 1 = \mathsf{map}\ (1+)\ \mathsf{nat}$

$$
\begin{aligned}
& \mathsf{nat} + 1 \\
={}& \quad \{\ \text{definition of } + \text{ and fromInteger}\ \} \\
& \mathsf{zip}\ (+)\ \mathsf{nat}\ (\mathsf{pure}\ 1) \\
={}& \quad \{\ \text{definition of zip}\ \} \\
& \mathsf{pure}\ (+) \diamond \mathsf{nat} \diamond \mathsf{pure}\ 1 \\
={}& \quad \{\ \text{Exercise 1.4}\ \} \\
& \mathsf{pure}\ (+) \diamond \mathsf{pure}\ 1 \diamond \mathsf{nat} \\
={}& \quad \{\ \text{homomorphism law}\ \} \\
& \mathsf{pure}\ (1+) \diamond \mathsf{nat} \\
={}& \quad \{\ \text{definition of map}\ \} \\
& \mathsf{map}\ (1+)\ \mathsf{nat}
\end{aligned}
$$

2.5 Lifting Lemma

Since the operations are lifted pointwise to streams, all the point-level identities hold for streams, as well.

2.5 Interleaving

$$\begin{aligned}
\mathrm{pure}\ a \curlyvee \mathrm{pure}\ a &= \mathrm{pure}\ a \\
(s_1 \diamond s_2) \curlyvee (t_1 \diamond t_2) &= (s_1 \curlyvee t_1) \diamond (s_2 \curlyvee t_2)
\end{aligned}$$

2.5 Abide law: geometrical interpretation

$$\begin{array}{ccc} s_1 & \diamond & s_2 \\ & \curlyvee & \\ t_1 & \diamond & t_2 \end{array} = \begin{array}{ccc} s_1 & & s_2 \\ \curlyvee & \diamond & \curlyvee \\ t_1 & & t_2 \end{array}$$

2.5 Recursion and iteration: fusion

$$\begin{array}{rcl}
\mathrm{map}\ h \cdot \mathrm{recurse}\ f_1 & = & \mathrm{recurse}\ f_2 \cdot h \\
& \Uparrow & \\
h \cdot f_1 & = & f_2 \cdot h \\
& \Downarrow & \\
\mathrm{map}\ h \cdot \mathrm{iterate}\ f_1 & = & \mathrm{iterate}\ f_2 \cdot h
\end{array}$$

2.5 Recursion-iteration lemma

$$\mathrm{recurse}\ f\ a \quad = \quad \mathrm{iterate}\ f\ a$$

2.6 Admissible equations

- Consider the equations

$$s_1 = \mathsf{tail}\ s_1$$

$$s_2 = \mathsf{head}\ s_2 \prec \mathsf{tail}\ s_2$$

- Both s_1 and s_2 loop in Haskell; viewed as stream equations they are ambiguous.
- An *admissible equation* has the form

$$x\ x_1\ \ldots\ x_n = h \prec t \ .$$

- The projection functions head and tail may only be applied to the arguments of x: $\mathsf{head}\ x_i$ or $\mathsf{tail}\ x_i$.

2.6 Unique solutions: $\mathrm{fix}\ \phi = x$

- Let $s = \phi\ s$ be an admissible equation.
- It has a unique solution, denoted by $\mathrm{fix}\ \phi$.
- Universal property:

$$\mathrm{fix}\ \phi = x \quad \iff \quad x = \phi\ x$$

2.6 Example: $\mathsf{nat} = 2 * \mathsf{nat} \curlyvee 2 * \mathsf{nat} + 1$

$$
\begin{aligned}
& 2 * \mathsf{nat} \curlyvee 2 * \mathsf{nat} + 1 \\
= \quad & \{ \text{ definition of nat } \} \\
& 2 * (0 \prec \mathsf{nat} + 1) \curlyvee 2 * \mathsf{nat} + 1 \\
= \quad & \{ \text{ arithmetic } \} \\
& (0 \prec 2 * \mathsf{nat} + 2) \curlyvee 2 * \mathsf{nat} + 1 \\
= \quad & \{ \text{ definition of } \curlyvee \} \\
& 0 \prec 2 * \mathsf{nat} + 1 \curlyvee 2 * \mathsf{nat} + 2 \\
= \quad & \{ \text{ arithmetic } \} \\
& 0 \prec (2 * \mathsf{nat} \curlyvee 2 * \mathsf{nat} + 1) + 1
\end{aligned}
$$

2.6 Unique solutions: $\mathrm{fix}\ \phi = \mathrm{fix}\ \psi$

- Let $s = \phi\ s$ and $t = \psi\ t$ be admissible equations.
- To prove $s = t$ there are at least four possibilities:

$$\begin{array}{lllll} \phi\ (\psi\ s) = \psi\ s & \Longrightarrow & \psi\ s = s & \Longrightarrow & s = t \\ \psi\ (\phi\ t) = \phi\ t & \Longrightarrow & \phi\ t = t & \Longrightarrow & s = t \end{array}$$

- Unfortunately, there is no success guarantee.

2.6 $\subset$-proofs

$$
\begin{array}{ll}
 & s \\
= & \{\ \text{why?}\ \} \\
 & \chi\ s \\
\subset & \{\ x = \chi\ x \text{ has a unique solution}\ \} \\
 & \chi\ t \\
= & \{\ \text{why?}\ \} \\
 & t
\end{array}
$$

2.6 Proof: Cassini's identity

$$
\begin{aligned}
& \mathit{fib}'^2 - \mathit{fib} * \mathit{fib}'' \\
= \quad & \{ \text{ definition of } \mathit{fib}'' \text{ and arithmetic } \} \\
& \mathit{fib}'^2 - (\mathit{fib} * \mathit{fib}' + \mathit{fib}^2) \\
= \quad & \{ \text{ definition of } \mathit{fib} \text{ and } \mathit{fib}' \ \} \\
& 1 \prec (\mathit{fib}''^2 - (\mathit{fib}' * \mathit{fib}'' + \mathit{fib}'^2)) \\
= \quad & \{ \ \mathit{fib}'' - \mathit{fib}' = \mathit{fib} \text{ and arithmetic } \} \\
& 1 \prec (-1) * (\mathit{fib}'^2 - \mathit{fib} * \mathit{fib}'') \\
\subset \quad & \{ \ x = 1 \prec (-1) * x \text{ has a unique solution } \} \\
& 1 \prec (-1) * (-1)^{\mathit{nat}} \\
= \quad & \{ \text{ definition of } \mathit{nat} \text{ and arithmetic } \} \\
& (-1)^{\mathit{nat}}
\end{aligned}
$$

2.6 Proof: iterate fusion

$$
\begin{array}{ll}
& \mathsf{map}\ h\ (\mathsf{iterate}\ f_1\ a) \\
= & \quad \{\ \text{definition of iterate and map}\ \} \\
& h\ a \prec \mathsf{map}\ h\ (\mathsf{iterate}\ f_1\ (f_1\ a)) \\
\subset & \quad \{\ x\ a = h\ a \prec x\ (f_1\ a)\ \text{has a unique solution}\ \} \\
& h\ a \prec \mathsf{iterate}\ f_2\ (h\ (f_1\ a)) \\
= & \quad \{\ \text{assumption: } h \cdot f_1 = f_2 \cdot h\ \} \\
& h\ a \prec \mathsf{iterate}\ f_2\ (f_2\ (h\ a)) \\
= & \quad \{\ \text{definition of iterate}\ \} \\
& \mathsf{iterate}\ f_2\ (h\ a)
\end{array}
$$

2.6 Proof: recursion-iteration lemma

We show that $\mathit{iterate}\ f\ a$ is the unique solution of $x = a \prec \mathit{map}\ f\ x$.

$$
\begin{aligned}
& \mathit{iterate}\ f\ a \\
= \quad & \{\ \text{definition of iterate}\ \} \\
& a \prec \mathit{iterate}\ f\ (f\ a) \\
= \quad & \{\ \text{iterate fusion law: } h = f_1 = f_2 = f\ \} \\
& a \prec \mathit{map}\ f\ (\mathit{iterate}\ f\ a)
\end{aligned}
$$

Consequently, $\mathit{nat} = \mathit{iterate}\ (1+)\ 0$.

2.6 Summary

- Stream is a co-inductive datatype.
- Stream is an idiom.
- Recursion and iteration.
- Admissible equations have unique solutions.

Part 3

Recurrences

3.0 Outline

3.1 Example: tower of Hanoï

$$\begin{aligned} \mathcal{T}_0 &= 0 \\ \mathcal{T}_{n+1} &= 2 * \mathcal{T}_n + 1 \end{aligned}$$

$$t = 0 \prec 2 * t + 1$$

3.1 Recurrences as streams

$$\begin{aligned}\mathcal{F}_0 &= k \\ \mathcal{F}_{n+1} &= f\ (\mathcal{F}_n)\end{aligned}$$

$$s = k \prec map\ f\ s$$

3.1 A one-to-one correspondence

$$\mathrm{Stream}\ \alpha \;\cong\; \mathrm{Nat} \to \alpha$$

3.1 Tabulation

$$\mathbf{data}\ \mathsf{Nat} = 0 \mid \mathsf{Nat} + 1$$

$$\begin{aligned}
&\mathsf{tabulate} :: (\mathsf{Nat} \to \alpha) \to \mathsf{Stream}\ \alpha \\
&\mathsf{tabulate}\ f = f\ 0 \prec \mathsf{tabulate}\ (f \cdot (+1))
\end{aligned}$$

$$\begin{aligned}
&\mathsf{lookup} :: \mathsf{Stream}\ \alpha \to (\mathsf{Nat} \to \alpha) \\
&\mathsf{lookup}\ s\ 0 &&= \mathsf{head}\ s \\
&\mathsf{lookup}\ s\ (n+1) &&= \mathsf{lookup}\ (\mathsf{tail}\ s)\ n
\end{aligned}$$

NB. tabulate and lookup are *natural transformations*.

3.1 Laws: naturality properties

$$\begin{aligned}\mathrm{map}\ f \cdot \mathrm{tabulate} &= \mathrm{tabulate} \cdot (f\ \cdot) \\ (f\ \cdot) \cdot \mathrm{lookup} &= \mathrm{lookup} \cdot \mathrm{map}\ f\end{aligned}$$

NB. $(f\ \cdot)$ is the mapping function of the functor $\alpha \rightarrow$.

$$\begin{aligned}\mathrm{map}\ f\ (\mathrm{tabulate}\ g) &= \mathrm{tabulate}\ (f \cdot g) \\ f \cdot \mathrm{lookup}\ t &= \mathrm{lookup}\ (\mathrm{map}\ f\ t)\end{aligned}$$

3.1 Laws: isomorphisms

$$\begin{aligned} \mathit{lookup} \cdot \mathit{tabulate} &= \mathit{id} \\ \mathit{tabulate} \cdot \mathit{lookup} &= \mathit{id} \end{aligned}$$

3.1 Reminder: initial algebras

$$\begin{aligned}
&\mathrm{fold} :: (\alpha \to \alpha) \to \alpha \to (\mathrm{Nat} \to \alpha)\\
&\mathrm{fold}\ s\ z\ 0 \qquad\quad = z\\
&\mathrm{fold}\ s\ z\ (n+1) = s\ (\mathrm{fold}\ s\ z\ n)
\end{aligned}$$

3.1 Reminder: universal property of fold

$$h = \mathrm{fold}\ s\ z \iff h\ 0 = z \ \land\ h \cdot (+1) = s \cdot h$$

3.1 Proof: $\mathsf{tabulate}\ (\mathsf{fold}\ s\ z) = \mathsf{iterate}\ s\ z$

$$
\begin{aligned}
& \mathsf{tabulate}\ (\mathsf{fold}\ s\ z) \\
= \quad & \{\ \text{definition of tabulate}\ \} \\
& \mathsf{fold}\ s\ z\ 0 \prec \mathsf{tabulate}\ (\mathsf{fold}\ s\ z \cdot (+1)) \\
= \quad & \{\ \text{computation rules: } \mathsf{fold}\ s\ z\ 0 = z \\
& \quad \text{and } \mathsf{fold}\ s\ z \cdot (+1) = s \cdot \mathsf{fold}\ s\ z\ \} \\
& z \prec \mathsf{tabulate}\ (s \cdot \mathsf{fold}\ s\ z) \\
= \quad & \{\ \text{naturality of tabulate}\ \} \\
& z \prec \mathsf{map}\ s\ (\mathsf{tabulate}\ (\mathsf{fold}\ s\ z))
\end{aligned}
$$

3.1 Proof: tabulate id = nat

$$
\begin{aligned}
& \mathsf{tabulate}\ \mathsf{id} \\
= \quad & \{\ \text{reflection law: } \mathsf{fold}\ (+1)\ 0 = \mathsf{id}\ \} \\
& \mathsf{tabulate}\ (\mathsf{fold}\ (+1)\ 0) \\
= \quad & \{\ \text{see above}\ \} \\
& \mathsf{iterate}\ (+1)\ 0 \\
= \quad & \{\ \text{recursion-iteration lemma}\ \} \\
& \mathsf{nat}
\end{aligned}
$$

3.2 Example: Fibonacci numbers

$$\begin{aligned}\mathcal{F}_0 &= 0\\ \mathcal{F}_1 &= 1\\ \mathcal{F}_{n+2} &= \mathcal{F}_n + \mathcal{F}_{n+1}\end{aligned}$$

$$\mathsf{fib} = 0 \prec 1 \prec \mathsf{fib} + \mathsf{tail}\ \mathsf{fib}$$

3.2 Environment idiom: $\alpha \rightarrow$

$$
\begin{aligned}
&\textbf{instance}\ \mathrm{Idiom}\ (\alpha \rightarrow)\ \textbf{where}\\
&\quad \mathrm{pure}\ a = \lambda x \rightarrow a\\
&\quad f \diamond g \quad = \lambda x \rightarrow (f\ x)\ (g\ x)
\end{aligned}
$$

NB. pure is the K combinator and $\diamond$ is the S combinator.

3.2 Idiom homomorphism

The natural transformation

$$h :: \phi\ \alpha \to \psi\ \alpha$$

is an *idiom homomorphism* iff

$$\begin{aligned} h\ (\mathrm{pure}\ a) &= \mathrm{pure}\ a \\ h\ (x \diamond y) &= h\ x \diamond h\ y\ . \end{aligned}$$

3.2 Tabulation

The natural transformations tabulate and lookup are idiom homomorphisms between Stream and Nat $\rightarrow$.

$$\begin{aligned} \mathrm{tabulate}\ (\mathrm{pure}\ a) &= \mathrm{pure}\ a \\ \mathrm{tabulate}\ (x \diamond y) &= \mathrm{tabulate}\ x \diamond \mathrm{tabulate}\ y \end{aligned}$$

$$\begin{aligned} \mathrm{lookup}\ (\mathrm{pure}\ a) &= \mathrm{pure}\ a \\ \mathrm{lookup}\ (x \diamond y) &= \mathrm{lookup}\ x \diamond \mathrm{lookup}\ y \end{aligned}$$

3.2 Derivation of fib

$$
\begin{aligned}
& \mathsf{tabulate}\ \mathcal{F} \\
= \quad & \{\ \text{definition of tabulate}\ \} \\
& \mathcal{F}_0 \prec \mathcal{F}_1 \prec \mathsf{tabulate}\ (\mathcal{F} \cdot (+2)) \\
= \quad & \{\ \text{definition of } \mathcal{F}\ \} \\
& 0 \prec 1 \prec \mathsf{tabulate}\ (\mathcal{F} + \mathcal{F} \cdot (+1)) \\
= \quad & \{\ \text{tabulate is an idiom homomorphism}\ \} \\
& 0 \prec 1 \prec \mathsf{tabulate}\ \mathcal{F} + \mathsf{tabulate}\ (\mathcal{F} \cdot (+1)) \\
= \quad & \{\ \text{definition of tabulate}\ \} \\
& 0 \prec 1 \prec \mathsf{tabulate}\ \mathcal{F} + \mathsf{tail}\ (\mathsf{tabulate}\ \mathcal{F})
\end{aligned}
$$

3.3 Example: Dijkstra's fusc function

Dijkstra's fusc sequence (EWD570 and EWD578).

$$\begin{aligned}\mathcal{S}_1 &= 1\\ \mathcal{S}_{2*n} &= \mathcal{S}_n\\ \mathcal{S}_{2*n+1} &= \mathcal{S}_n + \mathcal{S}_{n+1}\end{aligned}$$

$$\mathrm{fusc} = 1 \prec \mathrm{fusc} \curlyvee \mathrm{fusc} + \mathrm{tail\ fusc}$$

3.3 Recurrences as streams

$$\begin{aligned}
\mathcal{F}_0 &= k \\
\mathcal{F}_{2*n+1} &= f\ (\mathcal{F}_n) \\
\mathcal{F}_{2*n+2} &= g\ (\mathcal{F}_n)
\end{aligned}$$

$$s = k \prec \mathrm{map}\ f\ s \curlyvee \mathrm{map}\ g\ s$$

3.3 Example: most significant bit

$$\mathrm{msb} = 1 \prec 2 * \mathrm{msb} \curlyvee 2 * \mathrm{msb}$$

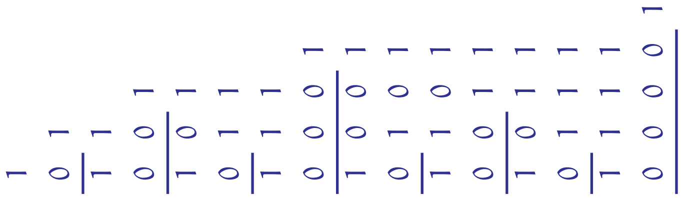
3.3 Example: 1s-counting sequence

$$\mathrm{ones} = \mathrm{ones} \curlyvee \mathrm{ones} + 1$$

$$\begin{aligned} \mathrm{ones} &= 0 \prec \mathrm{ones}' \\ \mathrm{ones}' &= 1 \prec \mathrm{ones}' \curlyvee \mathrm{ones}' + 1 \end{aligned}$$

3.3 Example: binary carry sequence

$$
\begin{array}{ccccc}
1 & & & & \\
0 & 1 & & & \\
\cline{1-1}
1 & 1 & & & \\
0 & 0 & 1 & & \\
\cline{1-2}
1 & 0 & 1 & & \\
0 & 1 & 1 & & \\
\cline{1-1}
1 & 1 & 1 & & \\
0 & 0 & 0 & 1 & \\
\cline{1-3}
1 & 0 & 0 & 1 & \\
0 & 1 & 0 & 1 & \\
\cline{1-1}
1 & 1 & 0 & 1 & \\
0 & 0 & 1 & 1 & \\
\cline{1-2}
1 & 0 & 1 & 1 & \\
0 & 1 & 1 & 1 & \\
\cline{1-1}
1 & 1 & 1 & 1 & \\
0 & 0 & 0 & 0 & 1 \\
\cline{1-4}
\end{array}
$$



$$\mathsf{carry} = 0 \curlyvee \mathsf{carry} + 1$$

$$\mathsf{carry} = 0 \prec \mathsf{carry} + 1 \curlyvee 0$$

3.3 Interactive session

$\gg$ msb

$\langle 1, 2, 2, 4, 4, 4, 4, 8, 8, 8, 8, 8, 8, 8, 8, 16, \ldots \rangle$

$\gg$ ones

$\langle 0, 1, 1, 2, 1, 2, 2, 3, 1, 2, 2, 3, 2, 3, 3, 4, \ldots \rangle$

$\gg$ carry

$\langle 0, 1, 0, 2, 0, 1, 0, 3, 0, 1, 0, 2, 0, 1, 0, 4, \ldots \rangle$

3.3 Summary

- Streams tabulate functions from the naturals.
- tabulate and lookup are idiom homomorphisms.
- Using $\prec$ and $\curlyvee$ we can express many recurrences.

Part 4

Finite Calculus

4.0 Outline

4.1 Finite difference

$$\begin{aligned} &\Delta &&:: (\mathsf{Num}\ \alpha) \Rightarrow \mathsf{Stream}\ \alpha \to \mathsf{Stream}\ \alpha \\ &\Delta\ s &&= \mathsf{tail}\ s - s \end{aligned}$$

4.1 Interactive session

$\gg \quad \Delta\, 2^{\mathrm{nat}}$

$\langle 1, 2, 4, 8, 16, 32, 64, 128, 256, 512, 1024, 2048, 4096, 8192, 16384, \ldots \rangle$

$\gg \quad \Delta\, \mathrm{carry}$

$\langle 1, -1, 2, -2, 1, -1, 3, -3, 1, -1, 2, -2, 1, -1, 4, -4, \ldots \rangle$

$\gg \quad \Delta\, \mathrm{nat}^3$

$\langle 1, 7, 19, 37, 61, 91, 127, 169, 217, 271, 331, 397, 469, 547, 631, 721, \ldots \rangle$

$\gg \quad 3 * \mathrm{nat}^2$

$\langle 0, 3, 12, 27, 48, 75, 108, 147, 192, 243, 300, 363, 432, 507, 588, 675, \ldots \rangle$

4.1 A *new* power

$$\Delta\,(\mathrm{nat}^{\underline{n+1}}) \quad = \quad (\mathrm{repeat}\; n+1) * \mathrm{nat}^{\underline{n}}$$

4.1 Derivation

$$
\begin{aligned}
& \Delta\ (\mathrm{nat}^{\underline{n+1}}) \\
={} & \quad \{\ \text{definition of } \Delta\ \} \\
& \mathrm{tail}\ (\mathrm{nat}^{\underline{n+1}}) - \mathrm{nat}^{\underline{n+1}} \\
={} & \quad \{\ \text{definition of nat}\ \} \\
& (\mathrm{nat} + 1)^{\underline{n+1}} - \mathrm{nat}^{\underline{n+1}} \\
={} & \quad \{\ \text{requirements: } x * (x-1)^{\underline{n}} = x^{\underline{n+1}} = x^{\underline{n}} * (x - n)\ \} \\
& (\mathrm{nat} + 1) * \mathrm{nat}^{\underline{n}} - \mathrm{nat}^{\underline{n}} * (\mathrm{nat} - \mathrm{repeat}\ n) \\
={} & \quad \{\ \text{arithmetic}\ \} \\
& (\mathrm{repeat}\ n + 1) * \mathrm{nat}^{\underline{n}}
\end{aligned}
$$

4.1 Falling factorial powers

$$\begin{aligned} x^{\underline{0}} &= 1 \\ x^{\underline{n+1}} &= x * (x - 1)^{\underline{n}} \end{aligned}$$

4.1 Interactive session

$\gg \quad \Delta\ (\mathrm{nat}^{\underline{3}})$
$\langle 0, 0, 6, 18, 36, 60, 90, 126, 168, 216, 270, 330, 396, 468, 546, 630, \ldots\rangle$

$\gg \quad 3 * \mathrm{nat}^{\underline{2}}$
$\langle 0, 0, 6, 18, 36, 60, 90, 126, 168, 216, 270, 330, 396, 468, 546, 630, \ldots\rangle$

4.1 Powers and falling factorial powers

$$\begin{aligned}
x^{0} &= x^{\underline{0}} \\
x^{1} &= x^{\underline{1}} \\
x^{2} &= x^{\underline{2}} + x^{\underline{1}} \\
x^{3} &= x^{\underline{3}} + 3 * x^{\underline{2}} + x^{\underline{1}} \\
x^{4} &= x^{\underline{4}} + 6 * x^{\underline{3}} + 7 * x^{\underline{2}} + x^{\underline{1}}
\end{aligned}$$

$$\begin{aligned}
x^{\underline{0}} &= x^{0} \\
x^{\underline{1}} &= x^{1} \\
x^{\underline{2}} &= x^{2} - x^{1} \\
x^{\underline{3}} &= x^{3} - 3 * x^{2} + 2 * x^{1} \\
x^{\underline{4}} &= x^{3} - 6 * x^{2} + 11 * x^{1} - 6 * x^{1}
\end{aligned}$$

4.1 Laws

$$\begin{aligned}
\Delta\ (\mathsf{tail}\ s) &= \mathsf{tail}\ (\Delta\ s) \\
\Delta\ (a \prec s) &= \mathsf{head}\ s - a \prec \Delta\ s \\
\Delta\ (s \curlyvee t) &= (t - s) \curlyvee (\mathsf{tail}\ s - t) \\
\Delta\ c &= 0 \\
\Delta\ (c * s) &= c * \Delta\ s \\
\Delta\ (s + t) &= \Delta\ s + \Delta\ t \\
\Delta\ (s * t) &= s * \Delta\ t + \Delta\ s * \mathsf{tail}\ t \\
\Delta\ c^{\mathsf{nat}} &= (c - 1) * c^{\mathsf{nat}} \\
\Delta\ (\mathsf{nat}^{\underline{n+1}}) &= (\mathsf{repeat}\ n + 1) * \mathsf{nat}^{\underline{n}}
\end{aligned}$$

4.1 Proof: product rule

$$
\begin{array}{ll}
 & \Delta\ (s * t) \\
= & \quad \{\ \text{definition of } \Delta \text{ and } *\ \} \\
 & \mathsf{tail}\ s * \mathsf{tail}\ t - s * t \\
= & \quad \{\ \text{arithmetic}\ \} \\
 & s * \mathsf{tail}\ t - s * t + \mathsf{tail}\ s * \mathsf{tail}\ t - s * \mathsf{tail}\ t \\
= & \quad \{\ \text{distributivity}\ \} \\
 & s * (\mathsf{tail}\ t - t) + (\mathsf{tail}\ s - s) * \mathsf{tail}\ t \\
= & \quad \{\ \text{definition of } \Delta\ \} \\
 & s * \Delta\ t + \Delta\ s * \mathsf{tail}\ t
\end{array}
$$

4.2 The right-inverse of Δ: Σ

$$
\begin{aligned}
& \Delta\ (\Sigma\ s) = s \\
\Longleftrightarrow\quad & \{\ \text{definition of } \Delta\ \} \\
& \mathsf{tail}\ (\Sigma\ s) - \Sigma\ s = s \\
\Longleftrightarrow\quad & \{\ \text{arithmetic}\ \} \\
& \mathsf{tail}\ (\Sigma\ s) = \Sigma\ s + s
\end{aligned}
$$

4.2 Summation

$$\begin{aligned}&\Sigma \quad :: \; (\mathrm{Num}\ \alpha) \Rightarrow \mathrm{Stream}\ \alpha \rightarrow \mathrm{Stream}\ \alpha\\ &\Sigma\ s = \mathrm{t}\ \mathbf{where}\ \mathrm{t} = 0 \prec \mathrm{t} + s\end{aligned}$$

$$
\begin{array}{ccccccccccccccc}
 & t_0 & t_1 & t_2 & t_3 & t_4 & t_5 & t_6 & t_7 & t_8 & t_9 & \cdots & & & t \\
 & + & + & + & + & + & + & + & + & + & + & \cdots & & & + \\
0 & s_0 & s_1 & s_2 & s_3 & s_4 & s_5 & s_6 & s_7 & s_8 & s_9 & \cdots & 0 & \prec & s \\
\| & \| & \| & \| & \| & \| & \| & \| & \| & \| & \| & \cdots & \| & & \\
t_0 & t_1 & t_2 & t_3 & t_4 & t_5 & t_6 & t_7 & t_8 & t_9 & t_{10} & \cdots & t & &
\end{array}
$$

4.2 Interactive session

$\gg\ \ \Sigma\ (0 \curlyvee 1)$

$\langle 0, 0, 1, 1, 2, 2, 3, 3, 4, 4, 5, 5, 6, 6, 7, 7, \ldots \rangle$

$\gg\ \ \Sigma\ (2 * \mathsf{nat} + 1)$

$\langle 0, 1, 4, 9, 16, 25, 36, 49, 64, 81, 100, 121, 144, 169, 196, 225, \ldots \rangle$

$\gg\ \ \Sigma\ \mathsf{carry}$

$\langle 0, 0, 1, 1, 3, 3, 4, 4, 7, 7, 8, 8, 10, 10, 11, 11, \ldots \rangle$

$\gg\ \ \Sigma\ (\mathsf{nat}\ \hat{}\ 2)$

$\langle 0, 0, 1, 5, 14, 30, 55, 91, 140, 204, 285, 385, 506, 650, 819, 1015, \ldots \rangle$

$\gg\ \ \Sigma\ (\mathsf{nat} * 2\ \hat{}\ \mathsf{nat})$

$\langle 0, 0, 2, 10, 34, 98, 258, 642, 1538, 3586, 8194, 18434, 40962, \ldots \rangle$

4.2 Summation by happenstance

$$t = 0 \prec t + s \iff \Sigma\, s = t$$

4.2 Proof: $\Sigma\,\mathsf{fib} = \mathsf{fib}' - 1$

$$\begin{aligned}
& \mathsf{fib} = 0 \prec \mathsf{fib} + (1 \prec \mathsf{fib}) \\
\Longleftrightarrow \quad & \{\ \text{summation by happenstance}\ \} \\
& \Sigma\,(1 \prec \mathsf{fib}) = \mathsf{fib} \\
\Longleftrightarrow \quad & \{\ \text{summation law, see next slide}\ \} \\
& 0 \prec 1 + \Sigma\,\mathsf{fib} = \mathsf{fib} \\
\Longrightarrow \quad & \{\ s_1 = s_2 \Longrightarrow \mathsf{tail}\ s_1 = \mathsf{tail}\ s_2\ \} \\
& 1 + \Sigma\,\mathsf{fib} = \mathsf{fib}' \\
\Longleftrightarrow \quad & \{\ \text{arithmetic}\ \} \\
& \Sigma\,\mathsf{fib} = \mathsf{fib}' - 1
\end{aligned}$$

4.2 Laws

$$
\begin{aligned}
\Sigma\,(\mathrm{tail}\ s) &= \mathrm{tail}\,(\Sigma\ s) - \mathrm{repeat}\,(\mathrm{head}\ s)\\
\Sigma\,(a \prec s) &= 0 \prec \mathrm{repeat}\ a + \Sigma\ s\\
\Sigma\,(s \curlyvee t) &= (\Sigma\ s + \Sigma\ t) \curlyvee (s + \Sigma\ s + \Sigma\ t)\\
\Sigma\ c &= c * \mathrm{nat}\\
\Sigma\,(c * s) &= c * \Sigma\ s\\
\Sigma\,(s + t) &= \Sigma\ s + \Sigma\ t\\
\Sigma\,(s * \Delta\ t) &= s * t - \Sigma\,(\Delta\ s * \mathrm{tail}\ t) - \mathrm{repeat}\,(\mathrm{head}\,(s * t))\\
\Sigma\ c^{\mathrm{nat}} &= c^{\mathrm{nat}-1} \,/\, (c - 1)\\
\Sigma\,(\mathrm{nat}^{\underline{n}}) &= \mathrm{nat}^{\underline{n+1}} \,/\, (\mathrm{repeat}\ n + 1)
\end{aligned}
$$

4.2 Fundamental Theorem

$$\mathrm{t} = \Delta\, s \iff \Sigma\, \mathrm{t} = s - \mathrm{repeat}\ (\mathrm{head}\ s)$$

4.2 Proof: summation by parts

Let $c = \mathsf{repeat}\ (\mathsf{head}\ (s * t))$, then

$$
\begin{aligned}
& s * \Delta\ t + \Delta\ s * \mathsf{tail}\ t = \Delta\ (s * t) \\
\iff & \quad \{ \text{ Fundamental Theorem } \} \\
& \Sigma\ (s * \Delta\ t + \Delta\ s * \mathsf{tail}\ t) = s * t - c \\
\iff & \quad \{\ \Sigma \text{ is linear } \} \\
& \Sigma\ (s * \Delta\ t) + \Sigma\ (\Delta\ s * \mathsf{tail}\ t) = s * t - c \\
\iff & \quad \{ \text{ arithmetic } \} \\
& \Sigma\ (s * \Delta\ t) = s * t - \Sigma\ (\Delta\ s * \mathsf{tail}\ t) - c \quad .
\end{aligned}
$$

4.2 Derivation: square pyramidal numbers

$$
\begin{aligned}
& \Sigma\, \mathit{nat}^{2} \\
= \quad & \{\ \text{converting to falling factorial powers}\ \} \\
& \Sigma\, (\mathit{nat}^{\underline{2}} + \mathit{nat}^{\underline{1}}) \\
= \quad & \{\ \text{summation laws}\ \} \\
& \frac{1}{3} * \mathit{nat}^{\underline{3}} + \frac{1}{2} * \mathit{nat}^{\underline{2}} \\
= \quad & \{\ \text{converting to ordinary powers}\ \} \\
& \frac{1}{3} * (\mathit{nat}^{3} - 3 * \mathit{nat}^{2} + 2 * \mathit{nat}) + \frac{1}{2} * (\mathit{nat}^{2} - \mathit{nat}) \\
= \quad & \{\ \text{arithmetic}\ \} \\
& \frac{1}{6} * (\mathit{nat} - 1) * \mathit{nat} * (2 * \mathit{nat} - 1)
\end{aligned}
$$

4.2 Derivation: $\Sigma\,(\mathsf{nat} * 2^{\mathsf{nat}})$

$$
\begin{aligned}
& \Sigma\,(\mathsf{nat} * 2^{\mathsf{nat}}) \\
= \quad & \{\ \Delta\, 2^{\mathsf{nat}} = 2^{\mathsf{nat}}\ \} \\
& \Sigma\,(\mathsf{nat} * \Delta\, 2^{\mathsf{nat}}) \\
= \quad & \{\ \text{summation by parts}\ \} \\
& \mathsf{nat} * 2^{\mathsf{nat}} - \Sigma\,(\Delta\,\mathsf{nat} * \mathsf{tail}\ 2^{\mathsf{nat}}) \\
= \quad & \{\ \Delta\,\mathsf{nat} = 1\text{, and definition of } \mathsf{nat}\ \} \\
& \mathsf{nat} * 2^{\mathsf{nat}} - 2 * \Sigma\, 2^{\mathsf{nat}} \\
= \quad & \{\ \text{summation law}\ \} \\
& \mathsf{nat} * 2^{\mathsf{nat}} - 2 * (2^{\mathsf{nat}} - 1) \\
= \quad & \{\ \text{arithmetic}\ \} \\
& (\mathsf{nat} - 2) * 2^{\mathsf{nat}} + 2
\end{aligned}
$$

4.2 Running time of the binary increment

Amortised running time of the binary increment: $\Sigma\,\mathit{carry}\,/\,\mathit{nat}$.

$\gg\ \ \Sigma\,\mathit{carry}$
$\langle 0, 0, 1, 1, 3, 3, 4, 4, 7, 7, 8, 8, 10, 10, 11, 11, \ldots\rangle$

$\gg\ \ \mathit{nat} - \Sigma\,\mathit{carry}$
$\langle 0, 1, 1, 2, 1, 2, 2, 3, 1, 2, 2, 3, 2, 3, 3, 4, \ldots\rangle$

$\gg\ \ \mathit{nat} - \Sigma\,\mathit{carry} \geqslant 0$
True

4.2 1s-counting sequence, again

$$\mathrm{ones} = 0 \prec \mathrm{ones} + 1 - \mathrm{carry}$$

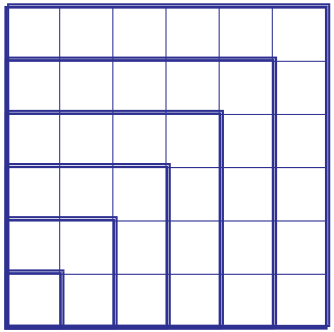
4.2 Proof: $\Sigma\ \mathsf{carry} = \mathsf{nat} - \mathsf{ones}$

$$
\begin{aligned}
& \mathsf{ones} = 0 \prec \mathsf{ones} + (1 - \mathsf{carry}) \\
\iff & \quad \{\ \text{summation by happenstance}\ \} \\
& \Sigma\ (1 - \mathsf{carry}) = \mathsf{ones} \\
\iff & \quad \{\ \text{arithmetic}\ \} \\
& \Sigma\ \mathsf{carry} = \mathsf{nat} - \mathsf{ones}
\end{aligned}
$$

4.3 Moessner's theorem: $n = 2$

$$\begin{array}{ccccccccccccc}
1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 & 10 & 11 & 12 & \ldots \\
1 & & 3 & & 5 & & 7 & & 9 & & 11 & & \ldots \\
1 & & 4 & & 9 & & 16 & & 25 & & 36 & & \ldots
\end{array}$$

4.3 A geometric proof



4.3 Moessner's theorem: $n = 3$

$$\begin{array}{ccccccccccccc}
1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 & 10 & 11 & 12 & \ldots \\
1 & 2 & & 4 & 5 & & 7 & 8 & & 10 & 11 & & \ldots \\
1 & 3 & & 7 & 12 & & 19 & 27 & & 37 & 48 & & \ldots \\
1 & & & 7 & & & 19 & & & 37 & & & \ldots \\
1 & & & 8 & & & 27 & & & 64 & & & \ldots
\end{array}$$

4.3 Proof: $n = 2$

$$\begin{array}{lcl} s_1 & \Upsilon & s_2 \\ s_1 + \Sigma\, s_1 & & \\ & & \\ 2 * nat^{\underline{1}} + 1 & \Upsilon & 2 * nat^{\underline{1}} + 2 \\ nat^{\underline{2}} + 3 * nat^{\underline{1}} + 1 & & \\ & & \\ 2 * nat + 1 & \Upsilon & 2 * nat + 2 \\ (nat + 1)^2 & & \end{array}$$

4.3 Proof: $n = 3$

$$\begin{array}{lclcl}
s_1 & \curlyvee_3 & s_2 & \curlyvee_3 & s_3 \\
s_1 + \Sigma\,(s_1 + s_2) & \curlyvee_2 & s_1 + s_2 + \Sigma\,(s_1 + s_2) & & \\
\multicolumn{5}{l}{s_1 + \Sigma\,(s_1 + s_2) + \Sigma\,(s_1 + \Sigma\,(s_1 + s_2))}
\end{array}$$

$$\begin{array}{lclcl}
3 * nat^{\underline{1}} + 1 & \curlyvee_3 & 3 * nat^{\underline{1}} + 2 & \curlyvee_3 & 3 * nat^{\underline{1}} + 3 \\
3 * nat^{\underline{2}} + 6 * nat^{\underline{1}} & \curlyvee_2 & 3 * nat^{\underline{2}} + 9 * nat^{\underline{1}} + 3 & & \\
\multicolumn{5}{l}{nat^{\underline{3}} + 6 * nat^{\underline{2}} + 7 * nat^{\underline{1}} + 1}
\end{array}$$

$$\begin{array}{lclcl}
3 * nat\ 1 + 1 & \curlyvee_3 & 3 * nat\ 1 + 2 & \curlyvee_3 & 3 * nat\ 1 + 3 \\
3 * nat^2 + 3 * nat & \curlyvee_2 & 3 * nat^2 + 6 * nat + 3 & & \\
\multicolumn{5}{l}{(nat + 1)^3}
\end{array}$$

4.3 Summary

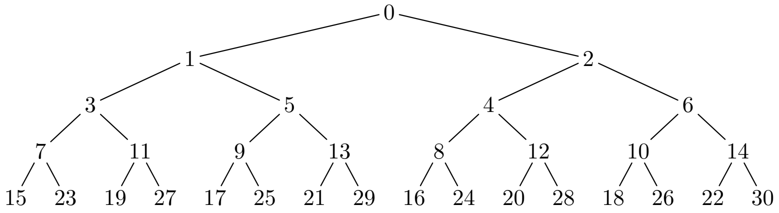
- Finite calculus serves as an elegant application of stream calculus.
- Avoidance of index variables and subscripts.

Part 5

Infinite Trees

5.0 Outline

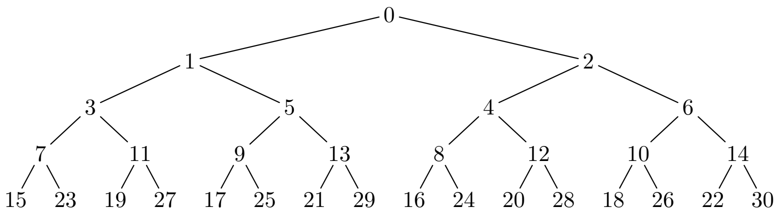
5.1 Infinite trees



5.1 Datatype of infinite trees

$$\mathbf{data}\ \mathrm{Tree}\ \alpha = \mathrm{Node}\ \{\mathrm{root} :: \alpha, \mathrm{left} :: \mathrm{Tree}\ \alpha, \mathrm{right} :: \mathrm{Tree}\ \alpha\}$$

5.1 Binary numbers



$$\mathrm{bin} = \mathrm{Node}\ 0\ (2 * \mathrm{bin} + 1)\ (2 * \mathrm{bin} + 2)$$

5.2 Trees are an idiom

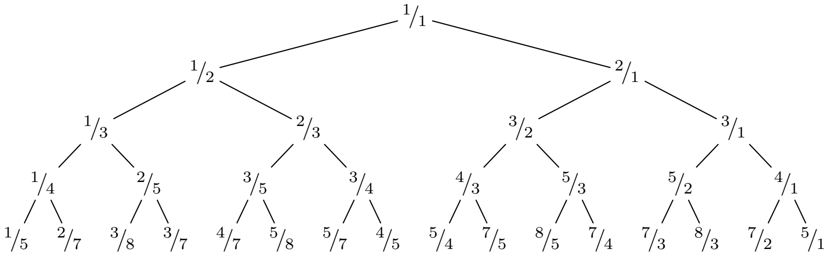
```
instance Idiom Tree where
  pure a = t where t = Node a t t
  t ⋄ u   = Node ((root t) (root u)) (left  t ⋄ left  u)
                                     (right t ⋄ right u)
```

5.2 Example: positive numbers

$$\mathrm{pos} = \mathrm{Node}\ 1\ (2 * \mathrm{pos} + 0)\ (2 * \mathrm{pos} + 1)$$

$$\mathrm{bin} + 1 = \mathrm{pos}$$

5.2 Example: Stern-Brocot tree



$$\mathrm{stern} = \mathrm{Node}\ 1\ (1\ /\ (1\ /\ \mathrm{stern} + 1))\ (\mathrm{stern} + 1)$$

5.2 Example: binary sequences

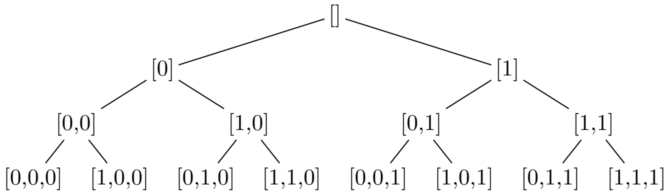
$$\mathrm{lbits} = \mathrm{Node}\ [\,]\ (\mathrm{map}\ ([0]{+\!\!+})\ \mathrm{lbits})\ (\mathrm{map}\ ([1]{+\!\!+})\ \mathrm{lbits})$$

$$\mathrm{rbits} = \mathrm{Node}\ [\,]\ (\mathrm{map}\ ({+\!\!+}[0])\ \mathrm{rbits})\ (\mathrm{map}\ ({+\!\!+}[1])\ \mathrm{rbits})$$

5.3 Recursion and iteration

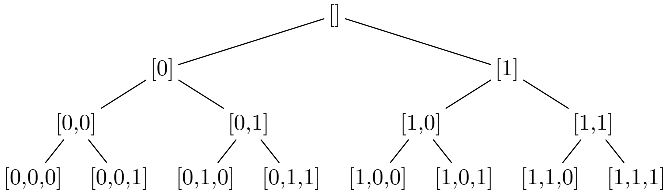
```
recurse :: (α → α) → (α → α) → (α → Tree α)
recurse f g a = t
  where t     = Node a (map f t) (map g t)

iterate :: (α → α) → (α → α) → (α → Tree α)
iterate f g a = loop a
  where loop x = Node x (loop (f x)) (loop (g x))
```

5.3 Example: bit strings, iteratively



$$\mathrm{rbits} = \mathrm{iterate}\ ([0] \mathbin{+\!\!+})\ ([1] \mathbin{+\!\!+})\ []$$

5.3 Example: bit strings, recursively



$$\mathrm{lbits} = \mathrm{recurse}\ ([0]{+\!\!+})\ ([1]{+\!\!+})\ []$$

5.4 A one-to-one correspondence

$$\mathrm{Tree}\ \alpha \quad \cong \quad \mathrm{Bin} \to \alpha$$

5.4 Tabulation

```
data Bin = Nil | One Bin | Two Bin

tabulate :: (Bin → α) → Tree α
tabulate f = Node (f Nil) (tabulate (f · One)) (tabulate (f · Two))

lookup :: Tree α → (Bin → α)
lookup t Nil       = root t
lookup t (One b) = lookup (left   t) b
lookup t (Two b) = lookup (right t) b
```

5.4 Facts

- tabulate and lookup are natural transformations.
- They are mutually inverse.
- tabulate (fold one two nil) = iterate one two nil.
- tabulate id = bin where

$$\text{bin} = \text{Node Nil (map One bin) (map Two bin)} \quad .$$

5.5 Another one-to-one correspondence

$$\mathrm{Stream}\ \alpha \quad \cong \quad \mathrm{Tree}\ \alpha$$

$$\mathrm{Nat} \quad \cong \quad \mathrm{Bin}$$

5.5 Linearising a tree

$$\begin{aligned} &\mathrm{stream} :: \mathrm{Tree}\ \alpha \rightarrow \mathrm{Stream}\ \alpha \\ &\mathrm{stream}\ t = \mathrm{root}\ t \prec \mathrm{stream}\ (\mathrm{left}\ t) \curlyvee \mathrm{stream}\ (\mathrm{right}\ t) \end{aligned}$$

$$\begin{aligned} &\mathrm{stream} :: \mathrm{Tree}\ \alpha \rightarrow \mathrm{Stream}\ \alpha \\ &\mathrm{stream}\ t = \mathrm{root}\ t \prec \mathrm{stream}\ (\mathrm{chop}\ t) \end{aligned}$$

$$\begin{aligned} &\mathrm{chop} :: \mathrm{Tree}\ \alpha \rightarrow \mathrm{Tree}\ \alpha \\ &\mathrm{chop}\ t = \mathrm{Node}\ (\mathrm{root}\ (\mathrm{left}\ t))\ (\mathrm{right}\ t)\ (\mathrm{chop}\ (\mathrm{left}\ t)) \end{aligned}$$

5.5 Constructing a tree

$$\begin{aligned}
&\mathsf{tree} &&:: \mathsf{Stream}\ \alpha \rightarrow \mathsf{Tree}\ \alpha\\
&\mathsf{tree}\ s &&= \mathsf{Node}\ (\mathsf{head}\ s)\ (\mathsf{tree}\ (\mathsf{even}\ (\mathsf{tail}\ s)))\ (\mathsf{tree}\ (\mathsf{odd}\ (\mathsf{tail}\ s)))
\end{aligned}$$

$$\begin{aligned}
&\mathsf{even}, \mathsf{odd} &&:: \mathsf{Stream}\ \alpha \rightarrow \mathsf{Stream}\ \alpha\\
&\mathsf{even}\ s &&= \mathsf{head}\ s \prec \mathsf{odd}\ \ (\mathsf{tail}\ s)\\
&\mathsf{odd}\ s &&= \phantom{\mathsf{head}\ s \prec{}} \mathsf{even}\ (\mathsf{tail}\ s)
\end{aligned}$$

$$\begin{aligned} \mathsf{tree}\ (a \prec l \curlyvee r) &= \mathsf{Node}\ a\ (\mathsf{tree}\ l)\ (\mathsf{tree}\ r) \\ a \prec \mathsf{stream}\ l \curlyvee \mathsf{stream}\ r &= \mathsf{stream}\ (\mathsf{Node}\ a\ l\ r) \end{aligned}$$

5.5 Interactive session

$\gg$ stream (iterate $(\lambda n \rightarrow 2 * n + 1)$ $(\lambda n \rightarrow 2 * n + 2)$ 0)
$\langle 0, 1, 2, 3, 5, 4, 6, 7, 11, 9, 13, 8, 12, 10, 14, 15, \ldots \rangle$

$\gg$ stream (recurse $(\lambda n \rightarrow 2 * n + 1)$ $(\lambda n \rightarrow 2 * n + 2)$ 0)
$\langle 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, \ldots \rangle$

5.6 Recursion and iteration: fusion

$$\begin{array}{ccc} \mathsf{map}\ h \cdot \mathsf{recurse}\ f_1\ g_1 & = & \mathsf{recurse}\ f_2\ g_2 \cdot h \\ & \Uparrow & \\ h \cdot f_1 = f_2 \cdot h & \wedge & h \cdot g_1 = g_2 \cdot h \\ & \Downarrow & \\ \mathsf{map}\ h \cdot \mathsf{iterate}\ f_1\ g_1 & = & \mathsf{iterate}\ f_2\ g_2 \cdot h \end{array}$$

5.6 Monoids

```
class Monoid $\alpha$ where
  $\epsilon$   :: $\alpha$
  $(\cdot)$ :: $\alpha \to \alpha \to \alpha$
```

5.6 Recursion-iteration lemma

$$\mathrm{recurse}\ (a\cdot)\ (b\cdot)\ \epsilon \quad = \quad \mathrm{iterate}\ (\cdot a)\ (\cdot b)\ \epsilon$$

NB. One is the *bit-reversal permutation tree* of the other.

5.6 Summary

- Tree is a co-inductive datatype.
- Tree is an idiom.
- Recursion and iteration.
- Admissible equations have unique solutions!
- Trees tabulate functions from the binary numbers.
- tabulate and lookup are idiom homomorphisms!

Part 6

Tabulation

6.1 Recap: streams

$$\mathrm{Nat} \to \gamma \quad \cong \quad \mathrm{Stream}\,\gamma$$

$$(\mu\,\alpha\,.\,1 + \alpha) \to \quad \cong \quad \nu\,\tau\,.\,\mathrm{Id} \mathbin{\dot{\times}} \tau$$

6.1 Recap: infinite trees

$$\mathrm{Bin} \to \gamma \quad \cong \quad \mathrm{Tree}\,\gamma$$

$$(\mu\,\alpha\,.\,1 + \alpha + \alpha) \to \quad \cong \quad \nu\,\tau\,.\,\mathrm{Id} \mathbin{\dot{\times}} \tau \mathbin{\dot{\times}} \tau$$

6.1 Laws of exponentials

$$\begin{aligned}
0 \to \gamma &\cong 1 \\
1 \to \gamma &\cong \gamma \\
(\alpha + \beta) \to \gamma &\cong (\alpha \to \gamma) \times (\beta \to \gamma) \\
(\alpha \times \beta) \to \gamma &\cong \alpha \to (\beta \to \gamma)
\end{aligned}$$

$$\begin{aligned}
0 \to &\cong \mathrm{K}\ 1 \\
1 \to &\cong \mathrm{Id} \\
(\alpha + \beta) \to &\cong (\alpha \to) \mathbin{\dot{\times}} (\beta \to) \\
(\alpha \times \beta) \to &\cong (\alpha \to) \cdot (\beta \to)
\end{aligned}$$

6.1 Tabulation

$$
\begin{array}{lcl@{\qquad\qquad}lcl}
F(\boldsymbol{\alpha}) & = & \alpha_i & H(\boldsymbol{\tau}) & = & \tau_i \\
F(\boldsymbol{\alpha}) & = & 0 & H(\boldsymbol{\tau}) & = & K\ 1 \\
F(\boldsymbol{\alpha}) & = & 1 & H(\boldsymbol{\tau}) & = & Id \\
F(\boldsymbol{\alpha}) & = & F_1(\boldsymbol{\alpha}) + F_2(\boldsymbol{\alpha}) & H(\boldsymbol{\tau}) & = & H_1(\boldsymbol{\tau}) \mathbin{\dot{\times}} H_2(\boldsymbol{\tau}) \\
F(\boldsymbol{\alpha}) & = & F_1(\boldsymbol{\alpha}) \times F_2(\boldsymbol{\alpha}) & H(\boldsymbol{\tau}) & = & H_1(\boldsymbol{\tau}) \cdot H_2(\boldsymbol{\tau}) \\
F(\boldsymbol{\alpha}) & = & \mu\, \alpha\ .\ F_1(\boldsymbol{\alpha}, \alpha) & H(\boldsymbol{\tau}) & = & \nu\, \tau\ .\ H_1(\boldsymbol{\tau}, \tau)
\end{array}
$$

6.1 Tabulation Theorem

$$\mathrm{F}(\alpha_1, \ldots, \alpha_n) \to \;\cong\; \mathrm{T}\,(\alpha_1 \to, \ldots, \alpha_n \to)$$

6.1 Example: integers

$$\mathbf{data}\ \mathrm{Int} = \mathrm{Neg}\ \mathrm{Nat} \mid \mathrm{Pos}\ \mathrm{Nat}$$

$$\mathrm{Int} \rightarrow \quad \cong \quad \mathrm{Stream}\ \dot{\times}\ \mathrm{Stream}$$

6.1 Example: pairs of naturals

$$(\mathrm{Nat} \times \mathrm{Nat}) \rightarrow \quad \cong \quad \mathrm{Stream} \cdot \mathrm{Stream}$$

6.1 Proofs

$F_1(\boldsymbol{\alpha}) + F_2(\boldsymbol{\alpha})$	case analysis	pairs	$H_1(\boldsymbol{\tau}) \dot{\times} H_2(\boldsymbol{\tau})$
$F_1(\boldsymbol{\alpha}) \times F_2(\boldsymbol{\alpha})$	pairs	nested proofs	$H_1(\boldsymbol{\tau}) \cdot H_2(\boldsymbol{\tau})$
$\mu\, \alpha \,.\, F_1(\boldsymbol{\alpha}, \alpha)$	induction	co-induction	$\nu\, \tau \,.\, H_1(\boldsymbol{\tau}, \tau)$

6.1 Applications

- Memoisation or tabulation of functions.
- Finite version: tries and generalised tries.
- Deforested: generic sorting and grouping.

6.1 References

- Richard H. Connelly and F. Lockwood Morris. “A generalization of the trie data structure”, *Math. Struct. in Comp. Science*, (5):381–418, 1995.
- Ralf Hinze. “Memo functions, polytypically!”, in Johan Jeuring, editor, *Proceedings of the 2nd Workshop on Generic Programming, Ponte de Lima, Portugal*, pages 17–32, 2000. The proceedings appeared as a technical report of Universiteit Utrecht, UU-CS-2000-19.
- Thorsten Altenkirch. “Representations of first order function types as terminal coalgebras”. In *Typed Lambda Calculi and Applications (TLCA 2001)*, LNCS 2044, pages 8–21, 2001.
- Fritz Henglein. “Generic discrimination: sorting and paritioning unshared data in linear time”. In Peter Thiemann, editor, *Proceedings of the 13th ACM Sigplan International Conference on Functional Programming (ICFP’08)*, September 22–24, 2008, Victoria, BC, Canada, pages 91–102. ACM Press, 2008.

Part 7

Epilogue

7.0 Summary

- Streams and infinite trees are tabulations.
- Idioms and idiom homomorphisms.
- Holistic or wholemeal approach to proving and programming.

7.0 What else?

- Equality of ADTs or objects.

$$\mathbf{data}\ \mathsf{Obj}\ \mathsf{F} = \exists\ \sigma\ .\ \mathsf{Obj}\ ((\sigma \to \mathsf{F}\ \sigma) \times \sigma)$$