Lightweight Semiformal Time Complexity Analysis for Purely Functional Data Structures

Nils Anders Danielsson

Chalmers/Nottingham

Teaser

$$(+\!\!+) : Seq\ a\ m \rightarrow Seq\ a\ n \rightarrow Thunk\ (1 + 2 * m)\ (Seq\ a\ (m + n))$$

Focus

- Purely functional (persistent) data structures.
- Complexity results valid for arbitrary usage patterns, not just single-threaded use.
- PFDSs which are efficient for all usage patterns often make essential use of laziness (call-by-need).
- Complexity analysis becomes subtle; many details to keep track of.
- This work: Type system and library which ensure that no details are forgotten.

Library

Library

- Types keep track of time complexity:

  $f : (n : \mathbb{N}) \rightarrow Thunk\ (1 + n)\ \mathbb{N}$

- In *dependently* typed language (Agda).

Meaning

$e : Thunk\ n_1\ (Thunk\ n_2 \ldots (Thunk\ n_k\ a) \ldots)$ means that it takes at most

$$n_1 + n_2 + \ldots + n_k$$

steps amortised time to evaluate e to WHNF, if this computation terminates.

Annotations

- Library based on user-inserted annotations:

  $\checkmark : \mathit{Thunk}\ n\ a \to \mathit{Thunk}\ (1+n)\ a$

- Every right-hand side should be ticked:

  $f\ (x :: xs) = \checkmark \ldots$

Machine assistance

- The library only checks correctness.
- Almost nothing inferred automatically.
- Recurrence equations have to be solved manually.

Example

Sequences

$$\begin{array}{l} \textbf{data}\ \mathit{Seq}\ (a : \star) : \mathbb{N} \to \star\ \textbf{where} \\ \quad \mathrm{nil}\ : \mathit{Seq}\ a\ 0 \\ \quad (::) : a \to \mathit{Seq}\ a\ n \to \mathit{Seq}\ a\ (1+n) \end{array}$$

Append

$$\begin{array}{l} (+\!\!+) : \mathit{Seq}\ a\ m \to \mathit{Seq}\ a\ n \\ \qquad\quad \to \mathit{Seq}\ a\ (m+n) \\ \mathrm{nil}\qquad\quad +\!\!+\ ys = ys \\ (x :: xs) +\!\!+\ ys = x :: (xs +\!\!+\ ys) \end{array}$$

Append

$$\begin{array}{l} (+\!\!+) : \mathit{Seq}\ a\ m \to \mathit{Seq}\ a\ n \\ \qquad\quad \to \mathit{Seq}\ a\ (m+n) \\ \mathrm{nil}\qquad\quad +\!\!+\ ys = \checkmark\ ys \\ (x :: xs) +\!\!+\ ys = \checkmark\ x :: (xs +\!\!+\ ys) \end{array}$$

$\checkmark : \mathit{Thunk}\ n\ a \to \mathit{Thunk}\ (1+n)\ a$

Append

$(+\!\!+) : \mathit{Seq}\ a\ m \to \mathit{Seq}\ a\ n$
$\quad \to \mathit{Thunk}\ (1 + m)\ (\mathit{Seq}\ a\ (m + n))$
$\mathrm{nil} \quad +\!\!+\ \mathit{ys} = {}^{\checkmark}\ \mathit{ys}$
$(x :: \mathit{xs})\ +\!\!+\ \mathit{ys} = {}^{\checkmark}\ x :: (\mathit{xs}\ +\!\!+\ \mathit{ys})$

${}^{\checkmark} : \mathit{Thunk}\ n\ a \to \mathit{Thunk}\ (1 + n)\ a$

Append

$(+\!\!+) : \mathit{Seq}\ a\ m \to \mathit{Seq}\ a\ n$
$\quad \to \mathit{Thunk}\ (1 + m)\ (\mathit{Seq}\ a\ (m + n))$
$\mathrm{nil} \quad +\!\!+\ \mathit{ys} = {}^{\checkmark}\ \mathit{return}\ \mathit{ys}$
$(x :: \mathit{xs})\ +\!\!+\ \mathit{ys} = {}^{\checkmark}$
$\quad \mathit{xs}\ +\!\!+\ \mathit{ys}\ \gg\!= \lambda \mathit{zs} \to$
$\quad \mathit{return}\ (x :: \mathit{zs})$

$\mathit{return} : a \to \mathit{Thunk}\ 0\ a$
$(\gg\!=)\ : \mathit{Thunk}\ m\ a \to (a \to \mathit{Thunk}\ n\ b)$
$\quad \to \mathit{Thunk}\ (m + n)\ b$

Append

$(+\!\!+) : \mathit{Seq}\ a\ m \to \mathit{Seq}\ a\ n$
$\quad \to \mathit{Thunk}\ (1 + 2 * m)\ (\mathit{Seq}\ a\ (m + n))$
$\mathrm{nil} \quad +\!\!+\ \mathit{ys} = {}^{\checkmark}\ \mathit{return}\ \mathit{ys}$
$(x :: \mathit{xs})\ +\!\!+\ \mathit{ys} = {}^{\checkmark}$
$\quad \mathit{xs}\ +\!\!+\ \mathit{ys}\ \gg\!= \lambda \mathit{zs} \to {}^{\checkmark}$
$\quad \mathit{return}\ (x :: \mathit{zs})$

$\mathit{return} : a \to \mathit{Thunk}\ 0\ a$
$(\gg\!=)\ : \mathit{Thunk}\ m\ a \to (a \to \mathit{Thunk}\ n\ b)$
$\quad \to \mathit{Thunk}\ (m + n)\ b$

Append

- Linear time to evaluate to WHNF?

  $(+\!\!+) : \ldots \to \mathit{Thunk}\ (1 + 2 * m)\ (\mathit{Seq}\ a\ (m + n))$

- *Seq* does not contain embedded *Thunk*s.
- Non-strict sequences also possible:

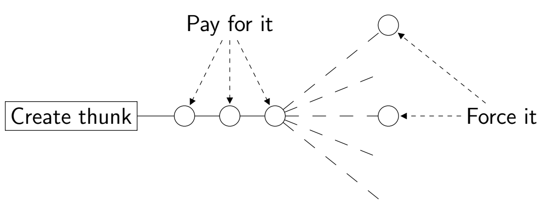
  data $S\ (a : \star)\ (c : \mathbb{N}) : \mathbb{N} \to \star$ **where**
  $\quad [\,]\ \ : S\ a\ c\ 0$
  $\quad (::)\ : a \to \mathit{Thunk}\ c\ (S\ a\ c\ n) \to S\ a\ c\ (1 + n)$

  $(+\!\!+) : S\ a\ c\ m \to S\ a\ c\ n$
  $\quad \to \mathit{Thunk}\ 2\ (S\ a\ (3 + c)\ (m + n))$

Essential laziness

Pay now, use later

- How can one take advantage of laziness (memoisation)?
- Let earlier operations pay for thunks which are forced later (perhaps several times):



Pay now, use later

- How can one take advantage of laziness (memoisation)?
- Let earlier operations pay for thunks which are forced later (perhaps several times):

$$\mathit{pay} : (m : \mathbb{N}) \to \mathit{Thunk}\ n\ a \to \mathit{Thunk}\ m\ (\mathit{Thunk}\ (n - m)\ a)$$

Summary

Library summary

$$\mathit{Thunk} : \mathbb{N} \to \star \to \star$$

$$\begin{aligned}
\checkmark &: \mathit{Thunk}\ n\ a \to \mathit{Thunk}\ (1 + n)\ a \\
\mathit{return} &: a \to \mathit{Thunk}\ 0\ a \\
(\gg\!=) &: \mathit{Thunk}\ m\ a \to (a \to \mathit{Thunk}\ n\ b) \\
&\quad \to \mathit{Thunk}\ (m + n)\ b \\
\mathit{pay} &: (m : \mathbb{N}) \to \mathit{Thunk}\ n\ a \\
&\quad \to \mathit{Thunk}\ m\ (\mathit{Thunk}\ (n - m)\ a)
\end{aligned}$$

Correctness

- Type system proved correct with respect to annotated big-step semantics (for toy language).
- Proof developed and checked using the Agda proof assistant.

Conclusions

- Simple library/type system for analysing time complexity of lazy functional programs.
- Well-defined semantics.
- Proved correct.
- Limitations:
  - Unstable type signatures: $\mathit{Thunk}\ (2 + 5 * n)\ a$.
  - Little support for aliasing.
- Applied to real-world examples.

Extra slides

Equality proofs

$(+\!\!+) : S\ a\ m \to S\ a\ n$
$\quad \to Thunk\ (1 + 2 * m)\ (S\ a\ (m + n))$
$\text{nil} \quad +\!\!+\ ys = {}^{\checkmark} return\ ys$
$x ::_m xs +\!\!+\ ys = {}^{\checkmark}$
$\quad cast\ (lemma\ m)$
$\quad\quad (xs +\!\!+\ ys \gg\!= \lambda zs \to {}^{\checkmark}$
$\quad\quad\quad return\ (x :: zs))$

$lemma : (m : \mathbb{N}) \to (1 + 2 * m) + 1 \equiv 2 * (1 + m)$

Library summary

$Thunk : \mathbb{N} \to \star \to \star$

${}^{\checkmark} \quad : Thunk\ n\ a \to Thunk\ (1 + n)\ a$
$return\ : a \to Thunk\ 0\ a$
$(\gg\!=)\ : Thunk\ m\ a \to (a \to Thunk\ n\ b)$
$\quad \to Thunk\ (m + n)\ b$
$pay\ : (m : \mathbb{N}) \to Thunk\ n\ a$
$\quad \to Thunk\ m\ (Thunk\ (n - m)\ a)$

$force\ : Thunk\ n\ a \to a$

Library implementation

$Thunk\ n\ a = a$

${}^{\checkmark}x \quad = x$
$return\ x \quad = x$
$x \gg\!= f \quad = f\ x$

$pay\ \_\ x \quad = x$

$force\ x \quad = x$

Essential laziness

$\textbf{data}\ Queue\ (a : \star) : \star\ \textbf{where}$
$\quad \text{empty} : \quad Queue\ a$
$\quad \text{cons}_{10} : a \to Queue\ (a \times a) \quad \to Queue\ a$
$\quad \text{cons}_{11} : a \to Queue\ (a \times a) \to a \to Queue\ a$

$snoc : Queue\ a \to a \to Queue\ a$
$snoc\ \text{empty} \quad x_1 = \text{cons}_{10}\ x_1\ \text{empty}$
$snoc\ (\text{cons}_{10}\ x_1\ xs_2) \quad x_3 = \text{cons}_{11}\ x_1\ xs_2\ x_3$
$snoc\ (\text{cons}_{11}\ x_1\ xs_2\ x_3)\ x_4 =$
$\quad \text{cons}_{10}\ x_1\ (snoc\ xs_2\ (x_3, x_4))$

Essential laziness

$\textbf{data}\ Queue\ (a : \star) : \star\ \textbf{where}$
$\quad \text{cons}_{10} : a \to Queue\ (a \times a) \to Queue\ a$

$snoc : Queue\ a \to a \to Thunk\ ?\ (Queue\ a)$
$snoc\ \text{empty} \quad x_1 = {}^{\checkmark} \text{cons}_{10}\ x_1\ \text{empty}$
$snoc\ (\text{cons}_{10}\ x_1\ xs_2) \quad x_3 = {}^{\checkmark} \text{cons}_{11}\ x_1\ xs_2\ x_3$
$snoc\ (\text{cons}_{11}\ x_1\ xs_2\ x_3)\ x_4 = {}^{\checkmark}$
$\quad \text{cons}_{10}\ x_1\ (snoc\ xs_2\ (x_3, x_4))$

Essential laziness

$\textbf{data}\ Queue\ (a : \star) : \star\ \textbf{where}$
$\quad \text{cons}_{10} : a \to Queue\ (a \times a) \to Queue\ a$

$snoc : Queue\ a \to a \to Thunk\ ?\ (Queue\ a)$
$snoc\ \text{empty} \quad x_1 = {}^{\checkmark} return\ (\text{cons}_{10}\ x_1\ \text{empty})$
$snoc\ (\text{cons}_{10}\ x_1\ xs_2) \quad x_3 = {}^{\checkmark} return\ (\text{cons}_{11}\ x_1\ xs_2\ x_3)$
$snoc\ (\text{cons}_{11}\ x_1\ xs_2\ x_3)\ x_4 = {}^{\checkmark}$
$\quad return\ (\text{cons}_{10}\ x_1\ (snoc\ xs_2\ (x_3, x_4)))$

Essential laziness

$\textbf{data}\ \mathit{Queue}\ (a : \star) : \star\ \textbf{where}$
$\quad \mathsf{cons}_{10} : a \to \mathit{Thunk}\ ?\ (\mathit{Queue}\ (a \times a)) \to \mathit{Queue}\ a$

$\mathit{snoc} : \mathit{Queue}\ a \to a \to \mathit{Thunk}\ ?\ (\mathit{Queue}\ a)$
$\mathit{snoc}\ \mathsf{empty}\qquad x_1 = {}^{\checkmark}$
$\quad \mathit{return}\ (\mathsf{cons}_{10}\ x_1\ (\mathit{return}\ \mathsf{empty}))$
$\mathit{snoc}\ (\mathsf{cons}_{10}\ x_1\ xs_2)\quad x_3 = {}^{\checkmark}\ \mathit{return}\ (\mathsf{cons}_{11}\ x_1\ xs_2\ x_3)$
$\mathit{snoc}\ (\mathsf{cons}_{11}\ x_1\ xs_2\ x_3)\ x_4 = {}^{\checkmark}$
$\quad \mathit{return}\ (\mathsf{cons}_{10}\ x_1\ (\mathit{snoc}\ xs_2\ (x_3, x_4)))$

Essential laziness

$\textbf{data}\ \mathit{Queue}\ (a : \star) : \star\ \textbf{where}$
$\quad \mathsf{cons}_{10} : a \to \mathit{Thunk}\ ?\ (\mathit{Queue}\ (a \times a)) \to \mathit{Queue}\ a$

$\mathit{snoc} : \mathit{Queue}\ a \to a \to \mathit{Thunk}\ ?\ (\mathit{Queue}\ a)$
$\mathit{snoc}\ \mathsf{empty}\qquad x_1 = {}^{\checkmark}$
$\quad \mathit{return}\ (\mathsf{cons}_{10}\ x_1\ (\mathit{return}\ \mathsf{empty}))$
$\mathit{snoc}\ (\mathsf{cons}_{10}\ x_1\ xs_2)\quad x_3 = {}^{\checkmark}$
$\quad xs_2 \ggg= \lambda xs_2' \to {}^{\checkmark}$
$\quad \mathit{return}\ (\mathsf{cons}_{11}\ x_1\ xs_2'\ x_3)$
$\mathit{snoc}\ (\mathsf{cons}_{11}\ x_1\ xs_2\ x_3)\ x_4 = {}^{\checkmark}$
$\quad \mathit{return}\ (\mathsf{cons}_{10}\ x_1\ (\mathit{snoc}\ xs_2\ (x_3, x_4)))$

Essential laziness

$\textbf{data}\ \mathit{Queue}\ (a : \star) : \star\ \textbf{where}$
$\quad \mathsf{cons}_{10} : a \to \mathit{Thunk}\ ?\ (\mathit{Queue}\ (a \times a)) \to \mathit{Queue}\ a$

$\mathit{snoc} : \mathit{Queue}\ a \to a \to \mathit{Thunk}\ ?\ (\mathit{Queue}\ a)$
$\mathit{snoc}\ \mathsf{empty}\qquad x_1 = {}^{\checkmark}$
$\quad \mathit{return}\ (\mathsf{cons}_{10}\ x_1\ (\mathit{return}\ \mathsf{empty}))$
$\mathit{snoc}\ (\mathsf{cons}_{10}\ x_1\ xs_2)\quad x_3 = {}^{\checkmark}$
$\quad xs_2 \ggg= \lambda xs_2' \to {}^{\checkmark}$
$\quad \mathit{return}\ (\mathsf{cons}_{11}\ x_1\ xs_2'\ x_3)$
$\mathit{snoc}\ (\mathsf{cons}_{11}\ x_1\ xs_2\ x_3)\ x_4 = {}^{\checkmark}$
$\quad \mathit{pay}\ ?\ (\mathit{snoc}\ xs_2\ (x_3, x_4)) \ggg= \lambda xs_{234} \to {}^{\checkmark}$
$\quad \mathit{return}\ (\mathsf{cons}_{10}\ x_1\ xs_{234})$

Essential laziness

$\textbf{data}\ \mathit{Queue}\ (a : \star) : \star\ \textbf{where}$
$\quad \mathsf{cons}_{10} : a \to \mathit{Thunk}\ 2\ (\mathit{Queue}\ (a \times a)) \to \mathit{Queue}\ a$

$\mathit{snoc} : \mathit{Queue}\ a \to a \to \mathit{Thunk}\ 4\ (\mathit{Queue}\ a)$
$\mathit{snoc}\ \mathsf{empty}\qquad x_1 = {}^{\checkmark\checkmark\checkmark\checkmark}$
$\quad \mathit{return}\ (\mathsf{cons}_{10}\ x_1\ ({}^{\checkmark\checkmark}\mathit{return}\ \mathsf{empty}))$
$\mathit{snoc}\ (\mathsf{cons}_{10}\ x_1\ xs_2)\quad x_3 = {}^{\checkmark}$
$\quad xs_2 \ggg= \lambda xs_2' \to {}^{\checkmark}$
$\quad \mathit{return}\ (\mathsf{cons}_{11}\ x_1\ xs_2'\ x_3)$
$\mathit{snoc}\ (\mathsf{cons}_{11}\ x_1\ xs_2\ x_3)\ x_4 = {}^{\checkmark}$
$\quad \mathit{pay}\ 2\ (\mathit{snoc}\ xs_2\ (x_3, x_4)) \ggg= \lambda xs_{234} \to {}^{\checkmark}$
$\quad \mathit{return}\ (\mathsf{cons}_{10}\ x_1\ xs_{234})$