Chasing Bottoms

A Case Study in Program Verification in the Presence of Partial and Infinite Values

Nils Anders Danielsson Patrik Jansson

Chalmers

Context

- Cover project.
- Verification of real Haskell programs.
- Haskell non-strict $\Rightarrow$ partial and infinite values relatively common.
  1. Lazily generate infinite structure.
  2. Use selected parts.

Case study

$$\mathbf{data}\ T = L \mid B\ T\ T$$

$$\begin{aligned}
&pretty' :: T \to String \\
&pretty'\ L \qquad\quad = \texttt{"L"} \\
&pretty'\ (B\ l\ r) = \texttt{"B"} +\!\!+ pretty'\ l +\!\!+ pretty'\ r
\end{aligned}$$

$$\begin{aligned}
&pretty :: (T, String) \to String \\
&pretty\ (t, cs) = pretty'\ t +\!\!+ cs
\end{aligned}$$

$$pretty\ (B\ L\ (B\ L\ L), \texttt{"x"}) = \texttt{"BLBLLx"}$$

Case study

$$
\begin{aligned}
&parse :: String \to (T, String)\\
&parse\ (\texttt{'L'} : cs) = (L, cs)\\
&parse\ (\texttt{'B'} : cs) = (B\ l\ r, cs'')\\
&\quad \textbf{where}\ (l, cs') \ = parse\ cs\\
&\qquad\qquad (r, cs'') = parse\ cs'
\end{aligned}
$$

Properties

For total, finite input:

- $parse \circ pretty = id :: (T, String) \to (T, String)$
- $pretty \circ parse \sqsubseteq id :: String \to String$

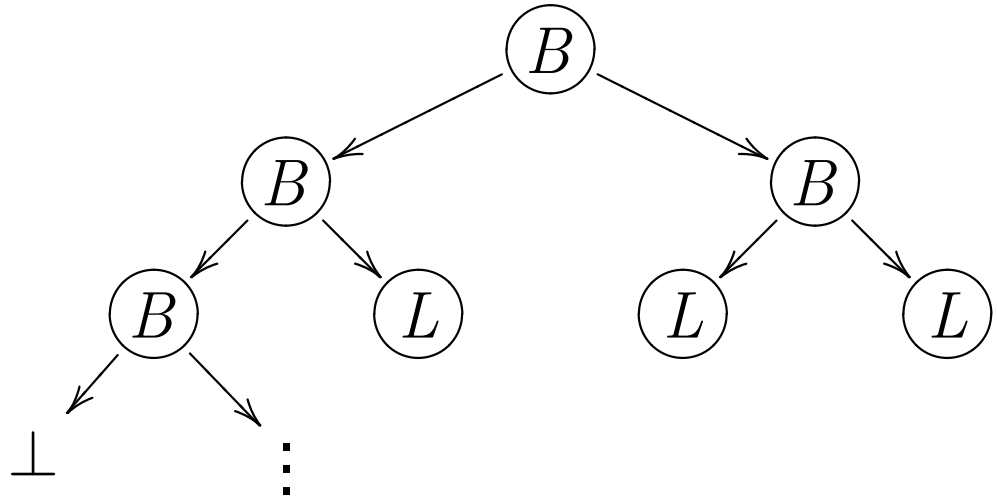
(Embedding-projection pair.)

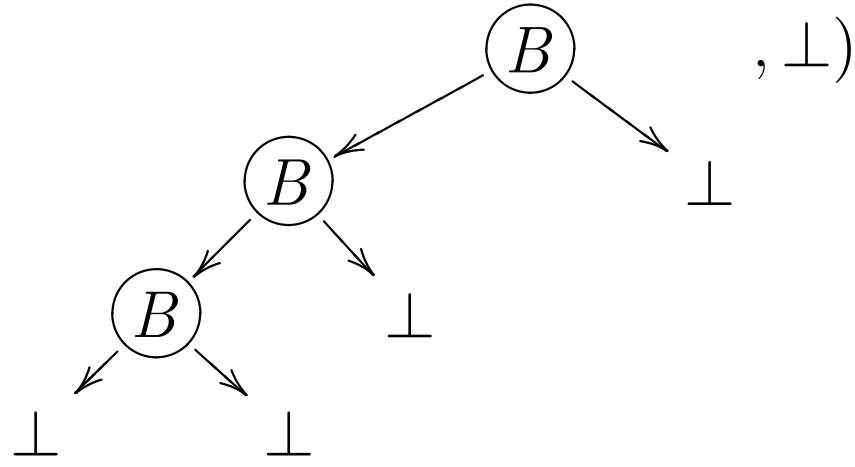
Properties

For arbitrary input:

- $parse \circ pretty = strictify \sqsubseteq id :: (T, String) \to (T, String)$
- $pretty \circ parse \sqsubseteq id :: String \to String$

Properties

- $p = ($  $, \text{"x"})$

- $strictify\ p = ($  $, \bot)$

Properties

$$
\begin{aligned}
&strictify :: (T, a) \to (T, a) \\
&strictify\ (t, a) = t\ `seq`\ (t', tTotal\ `seq`\ a) \\
&\quad \textbf{where}\ (t', tTotal) = strictify'\ t
\end{aligned}
$$

$$
\begin{aligned}
&strictify' :: T \to (T, ()) \\
&strictify'\ L \qquad\quad = (L, ()) \\
&strictify'\ (B\ l\ r) = (B\ l'\ (lTotal\ `seq`\ r'), lTotal\ `seq`\ rTotal) \\
&\quad \textbf{where}\ (l',\ lTotal) = strictify'\ l \\
&\qquad\qquad\quad (r', rTotal) = strictify'\ r
\end{aligned}
$$

seq

$$seq :: a \to b \to b$$
$$seq\ \bot\ b = \bot$$
$$seq\ a\ \ b = b \quad \text{-- if } a \neq \bot$$

Caveats

- η-equality is not valid.
  - $\bot \neq \lambda x.\bot$
- No surjective pairing.
  - $(fst\ \bot, snd\ \bot) = (\bot, \bot) \neq \bot$

Caveats

- Typical $Monad$ instances not monads.
  - $\bot \gg\!= return = \lambda x.\bot \neq \bot$
- Surprising (?) pattern matching semantics.
  - $\lambda True\ x.x \neq \lambda True.\lambda x.x$
  - $\mathbf{let}\ (x, y) = p\ \mathbf{in}\ f\ (x, y)$ equals $f\ p$ iff $p \neq \bot$ or $f\ (\bot, \bot) = f\ \bot$

Testing partial values

- $isBottom :: a \to Bool$

  1. Force evaluation of argument.
  2. Catch any exceptions (of the right kind).

- Several flavours of bottom:
  - Non-termination.
  - $error$ "..."
  - Pattern match failure.

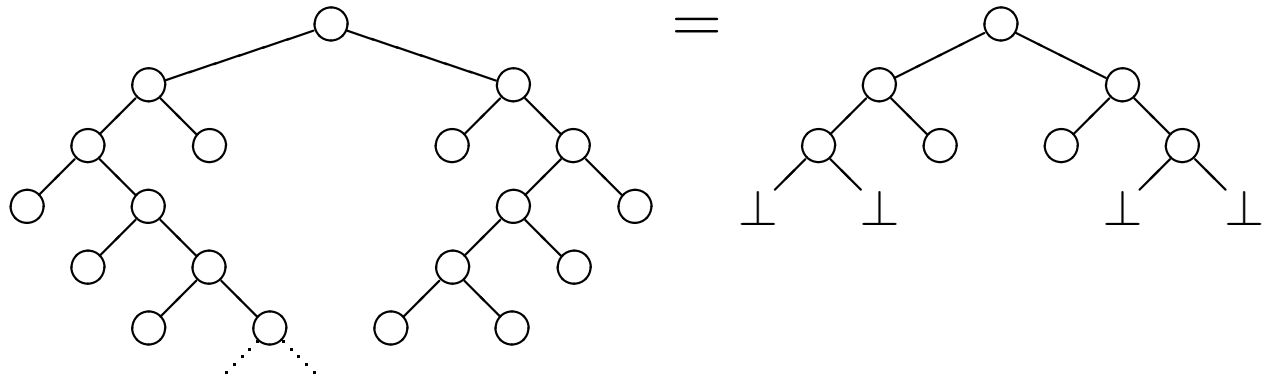
  Indistinguishable within pure subset of Haskell.

- ($isBottom :: a \to IO\ Bool$)

Testing partial values

- $\hat{=}, \hat{\sqsubseteq} :: Data\ a \Rightarrow a \rightarrow a \rightarrow Bool$
- Nice together with QuickCheck.

Testing infinite values

- $approx :: Nat \to T \to T$
- $approx\ 3$ 

- The approximation lemma:
  - $t_1 = t_2$ iff $\forall n \in \mathbb{N}.\ approx\ n\ t_1 = approx\ n\ t_2$

Testing partial and infinite values

- $pretty \circ parse \sqsubseteq id$

$$forAll\ string\ (\lambda cs.$$
$$approx\ n\ ((pretty \circ parse)\ cs) \mathbin{\hat{\sqsubseteq}} approx\ n\ (id\ cs))$$

- $parse \circ pretty = strictify$

$$forAll\ pair\ (\lambda p.$$
$$approxPair\ n\ ((parse \circ pretty)\ p) \mathbin{\hat{=}}$$
$$approxPair\ n\ (strictify\ p))$$
$$\textbf{where}\ approxPair\ n\ (t, cs) = (approx\ n\ t, approx\ (2\hat{}n)\ cs)$$

Proofs

- Fixpoint induction.
- The approximation lemma.
- Coinduction.
- Fusion.

Proofs

To prove

$$pretty \circ parse \sqsubseteq id$$

the following inductive hypothesis was used:

$$\forall m \in \mathbb{N}.\ pp_m \sqsubseteq id.$$

Here

$$\begin{aligned} pp_m\ cs = {} & pretty'\ t_1 +\!\!+ pretty'\ t_2 +\!\!+ \ldots +\!\!+ pretty'\ t_m +\!\!+ cs_m \\ \textbf{where}\ (t_1,\ cs_1) &= parse\ cs \\ (t_2,\ cs_2) &= parse\ cs_1 \\ &\vdots \\ (t_m, cs_m) &= parse\ cs_{m-1} \end{aligned}$$

Conclusions

- Testing reasonably easy, but test case coverage has to be checked.
- Proving tricky.
- Approximate semantics might be nice.
  - $\lambda x.\bot \rightsquigarrow \bot$
- Library available from `http://www.cs.chalmers.se/~nad/`.