Foundational Extensible Corecursion

Jasmin Blanchette  Andrei Popescu  Dmitriy Traytel

Technische Universität München
Foundational Extensible Corecursion

Jasmin Blanchette  Andrei Popescu  Dmitriy Traytel

Inria  Loria  Middlesex University London  ETH Zürich
Am I Productive?
s = 0 : s
\[ s = 0 : s \]

primitive corecursion
s = 0 : tail s

tail evil
s = 0 : 1 : s
s = 0 : 1 : s

corecursion up to constructors
eo s = head s : eo (tail (tail s))
\[
eo s = \text{head } s : \neo (\text{tail } (\text{tail } s))
\]
s = 0 : 1 : eo s
s = 0 : 1 : eo s

eo evil
\( s \oplus t = (\text{head } s + \text{head } t) : (\text{tail } s \oplus \text{tail } t) \)
\[ s \oplus t = (\text{head } s + \text{head } t) : (\text{tail } s \oplus \text{tail } t) \]

primitive corecursion
s ⊗ t = (head s ∗ head t) : (tail s ⊗ t ⊕ s ⊗ tail t)
s ⊗ t = (head s ∗ head t) : (tail s ⊗ t ⊕ s ⊗ tail t)
corecursion up to ⊕
\[ s = (0 : 1 : s) \oplus (\emptyset : s) \]
s = (0 : 1 : s) ⊕ (0 : s)

corecursion up to constructors and ⊕
\( s \ n = \ if \ n > 0 \) \\
then \( s \ (n - 1) \oplus (\emptyset : s \ (n + 1)) \) \\
else \( 1 : s \ 1 \)
\[
\begin{align*}
  s \ n &= \text{if } n > 0 \\
  &\quad \text{then } s (n - 1) \oplus (0 : s (n + 1)) \\
  &\quad \text{else } 1 : s 1
\end{align*}
\]

mixed recursion/corecursion up to \(\oplus\)
Contribution

Foundational framework for defining all the green stuff and more
Contribution

Foundational framework for defining all the green stuff and more in an LCF-style proof assistant.
Contribution

Foundational framework for defining all the green stuff and more in an LCF-style proof assistant.

Burden on the user: prove \{ parametricity \text{ or } termination \} here and there.
Contribution

Foundational framework for defining all the green stuff and more in an LCF-style proof assistant.

Burden on the user: prove \{ parametricity or termination \} here and there. Most of the time: automatic.
LCF Philosophy: Reduce everything to a small trusted kernel

Kernel of \( \text{Isabelle HOL} \) \( \approx \) \{ simply typed lambda calculus + classical higher-order logic (axioms) + nonrecursive constant definition + nonrecursive type definition \}

Context
LCF Philosophy: Reduce everything to a small trusted kernel

Kernel of Isabelle/HOL $\approx$ 
- simply typed lambda calculus +
- classical higher-order logic (axioms) +
- nonrecursive constant definition +
- nonrecursive type definition

Our agenda make Isabelle/HOL a (co)recursion-friendly environment

LICS’12  ITP’14  IJCAR’14  ESOP’15  ICFP’15
Related Work

A lot
## Related Work

### Guarded Coprogramming/Proof Assistants

<table>
<thead>
<tr>
<th>System</th>
<th>Features</th>
</tr>
</thead>
<tbody>
<tr>
<td>Isabelle</td>
<td>primitive corecursion, corecursor</td>
</tr>
</tbody>
</table>

**FRP** (Krishnaswami & Benton) type system

- clocks (Atkey & McBride) type system
- guards (Clouston et al.) type system

Isabelle: corecursion up-to 

friendly operations smart corecursor mixed with recursion + wellfounded recursion
## Related Work

### Guarded Coprogramming/Proof Assistants

<table>
<thead>
<tr>
<th>Tool</th>
<th>Feature</th>
<th>Feature</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Isabelle</strong></td>
<td>primitive corecursion</td>
<td>corecursor</td>
</tr>
<tr>
<td><strong>Coq</strong></td>
<td>corecursion up-to constructors</td>
<td>built-in</td>
</tr>
</tbody>
</table>
## Related Work

Guarded Coprogramming/Proof Assistants

<table>
<thead>
<tr>
<th>System</th>
<th>Corecursion/Coinduction Features</th>
<th>Additional Features</th>
</tr>
</thead>
<tbody>
<tr>
<td>Isabelle</td>
<td>primitive corecursion</td>
<td>corecursor</td>
</tr>
<tr>
<td>Coq</td>
<td>corecursion up-to constructors</td>
<td>built-in</td>
</tr>
<tr>
<td>Agda</td>
<td>copatterns + sized types</td>
<td>built-in + type system</td>
</tr>
</tbody>
</table>
## Related Work

Guarded Coprogramming/Proof Assistants

<table>
<thead>
<tr>
<th>System</th>
<th>Support</th>
<th>Features</th>
</tr>
</thead>
<tbody>
<tr>
<td>Isabelle</td>
<td>primitive corecursion</td>
<td>corecursion friendly operations</td>
</tr>
<tr>
<td>Coq</td>
<td>corecursion up-to constructors</td>
<td>built-in</td>
</tr>
<tr>
<td>Agda</td>
<td>copatterns + sized types</td>
<td>built-in + type system</td>
</tr>
<tr>
<td></td>
<td>- FRP (Krishnaswami &amp; Benton, ...)</td>
<td>type system</td>
</tr>
<tr>
<td></td>
<td>- clocks (Atkey &amp; McBride)</td>
<td>type system</td>
</tr>
<tr>
<td></td>
<td>- guards (Clouston et al.)</td>
<td>type system</td>
</tr>
</tbody>
</table>
## Related Work

**Guarded Coprogramming/Proof Assistants**

<table>
<thead>
<tr>
<th>System</th>
<th>Features</th>
</tr>
</thead>
<tbody>
<tr>
<td>Isabelle</td>
<td>primitive corecursion</td>
</tr>
<tr>
<td>Coq</td>
<td>corecursion up-to constructors</td>
</tr>
<tr>
<td>Agda</td>
<td>copatterns + sized types</td>
</tr>
<tr>
<td></td>
<td>FRP (Krishnaswami &amp; Benton, ...)</td>
</tr>
<tr>
<td></td>
<td>clocks (Atkey &amp; McBride)</td>
</tr>
<tr>
<td></td>
<td>guards (Clouston et al.)</td>
</tr>
<tr>
<td>Isabelle’</td>
<td>corecursion up-to <em>friendly</em> operations mixed with recursion</td>
</tr>
<tr>
<td></td>
<td>smart corecursor</td>
</tr>
<tr>
<td></td>
<td>+ wellfounded recursion</td>
</tr>
</tbody>
</table>

*Built-in features are marked in blue.*
Primitive Corecursor

codatatype $Stream = Int : Stream$
codatatype $Stream = Int : Stream$

$Stream \cong \text{gfp} (Int \times -)$

$corec^P :: (A \to Int \times A) \to A \to Stream$
Primitive Corecursor

codatatype $Stream = \text{Int} : Stream$

- $Stream \simeq \text{gfp} (\text{Int} \times - )$
- $\text{corec}^P :: (A \rightarrow \text{Int} \times A) \rightarrow A \rightarrow Stream$

$\text{primcorec} s \oplus t = (\text{head} s + \text{head} t) : (\text{tail} s \oplus \text{tail} t)$
Primitive Corecursor

codatatype $Stream = \text{Int} : Stream$

$Stream \simeq \text{gfp} (\text{Int} \times -)$

$corec^P :: (A \rightarrow \text{Int} \times A) \rightarrow A \rightarrow Stream$

$\text{primcorec } s \oplus t = (\text{head } s + \text{head } t) : (\text{tail } s \oplus \text{tail } t)$

$s \oplus t = \text{corec}^P (\lambda (s,t). ((\text{head } s + \text{head } t), (\text{tail } s, \text{tail } t))) (s,t)$
Primitive Corecursor

\[
codatatype \, C = \cdots
\]

\[
- \, C \simeq \text{gfp} \, F
\]

\[
- \, \text{corec}^P \, :: \,(A \to F \, A) \to A \to C
\]

\[
\text{primcorec} \, f \, x = \cdots
\]

\[
- \, f \, x = \text{corec}^P \, (\lambda (x). \cdots ) (x)
\]

(Assuming \( F \) is a bounded natural functor)
Smart Corecursor

corec^P :: (A \rightarrow F A) \rightarrow A \rightarrow C
Smart Corecursor

\[ \text{corec}^P :: (A \rightarrow F A) \rightarrow A \rightarrow C \]

\[ \text{corec}_0^S :: (A \rightarrow \square (F (\square A))) \rightarrow A \rightarrow C \]
Smart Corecursor

\[
corec^P :: (A \rightarrow F A) \rightarrow A \rightarrow C
\]

\[
corec^S_0 :: (A \rightarrow \square (F (\square A))) \rightarrow A \rightarrow C
\]

\[
corec^S_1 :: (A \rightarrow \oplus (F (\oplus A))) \rightarrow A \rightarrow C
\]
**Smart Corecursor**

\[
\text{corec}^P :: (A \rightarrow F A) \rightarrow A \rightarrow C
\]

\[
\text{corec}^S_0 :: (A \rightarrow \Box (F (\Box A))) \rightarrow A \rightarrow C
\]

\[
\text{corec}^S_1 :: (A \rightarrow \bigoplus (F (\bigoplus A))) \rightarrow A \rightarrow C
\]

\[
\text{corec} s \otimes t = (\text{head } s \ast \text{head } t) : (\text{tail } s \otimes t \oplus s \otimes \text{tail } t)
\]

\[
\eta((\text{head } s \ast \text{head } t), \eta(\text{tail } s, t) \bigoplus \eta(s, \text{tail } t))(s, t)
\]

\[
\bigoplus :: \bigoplus A \rightarrow \bigoplus A \rightarrow \bigoplus A
\]

\[
\eta :: A \rightarrow \bigoplus A
\]
Smart Corecursor

corec^P :: (A \to F A) \to A \to C

corec_0^S :: (A \to \text{[gray]} (F (\text{[gray]} A))) \to A \to C

corec_1^S :: (A \to \text{[white]} (F (\text{[white]} A))) \to A \to C

corec_2^S :: (A \to \text{[green]} (F (\text{[green]} A))) \to A \to C

corec \otimes t = (\text{head } s \ast \text{head } t) : (\text{tail } s \otimes t \oplus s \otimes \text{tail } t)

= s \otimes t = \text{corec}_1^S (\lambda(s, t).

\eta((\text{head } s \ast \text{head } t), \eta(\text{tail } s, t) \oplus \eta(s, \text{tail } t))) (s, t)

= \oplus :: \text{[white]} A \to \text{[white]} A \to \text{[white]} A

= \eta :: A \to \text{[white]} A
\( \otimes :: C \to C \to C \) has to be friendly

A friendly function can destroy one constructor to produce at least one constructor.
\(\otimes :: C \rightarrow C \rightarrow C\) has to be *friendly*

\[\exists\text{ parametric }\rho\otimes :: (A \times F A) \rightarrow (A \times F A) \rightarrow F (\mathcal{G} A)\ \text{s.t.}\]
\[s\otimes t = \cdots (\rho\otimes (\cdots (s, t)))\]
$\otimes :: C \to C \to C$ has to be friendly

\exists \text{ parametric } \rho_{\otimes} :: (A \times F A) \to (A \times F A) \to F (\bigotimes A) \text{ s.t. }
\begin{align*}
s \otimes t = \cdots (\rho_{\otimes} (\cdots (s, t)))
\end{align*}

\begin{align*}
\rho_{\otimes} :: (A \times (\text{Int} \times A)) & \to (A \times (\text{Int} \times A)) \to (\text{Int} \times \bigotimes A) \\
\rho_{\otimes} (s, hs, ts) (t, ht, tt) & = (hs \ast ht, \eta ts \bigotimes \eta t \bigoplus \eta s \bigotimes \eta tt)
\end{align*}
In the paper

In the meantime

Coq constructor + Agda constructor · arbitrary (manual proofs)

Isabelle friendly · constructor · friendly ∗ (auto proofs)

Thanks for listening! Questions?
In the paper

Figure 5: A new friendly operation \( \xi \)

Figure 6: Mixed fixpoint
In the paper

The following Isabelle-like theory fragment gives a flavor of the envisioned functionality from the user’s point of view:

```plaintext
codatatype Stream A = SCons (head: A) (tail: Stream A)
corec (friendly) ⊕: Stream → Stream → Stream
  .xs ⊕ ys = SCons (head xs + head ys) (tail xs ⊕ tail ys)
corec (friendly) ⊗: Stream → Stream → Stream
  .xs ⊗ ys = SCons (head xs × head ys)
    ((xs ⊗ tail ys) ⊕ (tail xs ⊗ ys))
```
In the paper

The following Isabelle-like theory fragment gives a flavor of the envisioned functionality from the user’s point of view:

\[
\text{codatatype Stream } A = \text{SCons (head: A) (tail: Stream A)}
\]

\[
\text{corec (friendly) } \oplus : \text{Stream } \rightarrow \text{Stream } \rightarrow \text{Stream}
\]

\[
\text{xs } \oplus \text{ ys } = \text{SCons (head xs + head ys) (tail xs } \oplus \text{ tail ys)}
\]

\[
\text{corec (friendly) } \otimes : \text{Stream } \rightarrow \text{Stream } \rightarrow \text{Stream}
\]

\[
\text{xs } \otimes \text{ ys } = \text{SCons (head xs } \times \text{ head ys)}
\]

\[
((\text{xs } \otimes \text{ tail ys) } \oplus (\text{tail xs } \otimes \text{ ys}))
\]

In the meantime

```
codatatype 'a llist = Nil | Cons (head: 'a) (tail: "a llist") (infix "::")
```

```
corec zeros where "zeros = 0::zeros"
corec zero_ones where "zero_ones = 0::zero_ones"
corec evens where "evens s = head s :: evens (tail (tail s))"
corec fib where "fib = 0::fib + 0::fib"
corec (friendly) pow (infix "^") 70 where "S o t = (head s :: head t) :: (tail s } \oplus \text{ tail t)}
```

Figure 5: A new friendly operation \( e \).

Figure 6: Mixed instance.

```
corecUU_1list_v1
(\lambda(s, t).
  VLeaf_list_v1
  (Inr (head s * head t, 0::VLeaf_list_v1
  (Sig_list_v1
    (Inr (VLeaf_list_v1 (Inr (tail s, t)), VLeaf_list_v1 (Inr (s, tail t))) ) ) ) )
```

Thanks for listening!

Questions?
In the paper

- Coq: constructor+
- Agda: constructor+ · arbitrary (manual proofs)
- Isabelle: friendly* · constructor · friendly* (auto proofs)

Thanks for listening!

Questions?
What is $s_1$?

$s_1 = \text{if } n > 0 \text{ then } s_{n - 1} \oplus (0 : s_{n + 1}) \text{ else } 1 : s_1$