Generalized general recursion

Thorsten Altenkirch

University of Nottingham
General recursion

\[ \text{gcd}' \in \text{Nat} \rightarrow \text{Nat} \rightarrow \text{Nat} \]

\[ \text{gcd}' \ m \ n \]
\[ \quad | \quad m = n = m \]
\[ \quad | \quad m < n = \text{gcd}' \ (m-n) \ n \]
\[ \quad | \quad n < m = \text{gcd}' \ m \ (n-m) \]
General recursion ... Paulson 86, Nordström 88

\[ f \in \Pi a \in A. (\Pi b \in A. (b < a) \rightarrow B) \rightarrow B \]

\[ \text{fix}(f) \in \Pi a \in A. (\text{Acc} < a) \rightarrow B \]

where \( \text{Acc} \) is defined inductively:

\[ \Pi b \in A. (b < a) \rightarrow \text{Acc} < b \]

\[ \text{Acc} < a \]
Better general recursion

Bove and Capretta
Define a specific termination predicate for each recursive function.

McBride and McKinna
Turn recursive programs into structurally recursive ones.
Better general recursion

Bove and Capretta
Define a specific termination predicate for each recursive function.
Better general recursion

Bove and Capretta
Define a specific termination predicate for each recursive function.

McBride and McKinna
Turn recursive programs into structurally recursive ones.
\textbf{nats}?

\[\text{nats} \in \text{Nat} \rightarrow [\text{Nat}]\]

\[\text{nats } n = n : (\text{nats } (n+1))\]
nats! 

\textbf{nats} cannot be defined by well-founded recursion.
nats !

- nats cannot be defined by well-founded recursion.
- nats can be defined using coiteration.
nats!

- nats cannot be defined by well-founded recursion.
- nats can be defined using coiteration.
- nats can be defined by guarded corecursion (Coquand 94).
merge ∈ [Nat] → [Nat] → [Nat]
merge (as @ (a:as')) (bs @ (b:bs'))
  | a<b = a:(merge as’ bs)
  | b<a = b:(merge as bs’)
  | a==b = a:(merge as’ bs’)

ham ∈ [Nat]
ham = 2 : (merge (map (λ i → 2*i) ham)
          (map (λ i → 3*i) ham))
**ham**

- **ham** cannot be defined by well-founded recursion.
ham cannot be defined by well-founded recursion.

It is not obvious how to use coiteration to define ham.
ham?

- ham cannot be defined by well-founded recursion.
- It is not obvious how to use coiteration to define ham.
- ham is not guarded!
primes ??

\[
s\text{ieve} \in \text{[Nat]} \to \text{[Nat]} \to \text{[Nat]}
\]
\[
s\text{ieve} \ (\text{ns} @ (n:\text{ns}')) \ (\text{ps} @ (p:\text{ps}'))
\]
\[
| \ n < p*p \quad = \ n:(\text{ieve} \ \text{ns}' \ \text{primes})
\]
\[
| \ \text{mod} \ n \ p \ == \ 0 \quad = \ \text{ieve} \ \text{ns}' \ \text{primes}
\]
\[
| \ \text{otherwise} \quad = \ \text{ieve} \ \text{ns} \ \text{ps}'
\]

\[
\text{primes} \in \text{[Nat]}
\]
\[
\text{primes} = 2 \ : \ (\text{ieve} \ (\text{nats} \ 3) \ \text{primes})
\]
Generalized general recursion

Generalizing well-founded recursion to coinductive domains
Fixpoints of contractive maps using converging equivalence relations (CERs)
Fixpoints of functions with coinductive codomains which are total even though they are not guarded.
Wellfounded recursion (general recursion) arises as a special case.
Developed in a classical setting (Isabelle, HOL).
Generalized general recursion

John Matthews (2001)

*Generalizing well-founded recursion to coinductive domains*
Generalized general recursion

- John Matthews (2001)
  Generalizing well-founded recursion to coinductive domains
- Fixpoints of contractive maps using converging equivalence relations (CERs) filtered limits.
Generalized general recursion

- John Matthews (2001)
  *Generalizing well-founded recursion to coinductive domains*
- Fixpoints of contractive maps using *converging equivalence relations* (CERs) \(\simeq\) filtered limits.
- Fixpoints of functions with coinductive codomains which are total even though they are not guarded.
Generalized general recursion

- John Matthews (2001)
  *Generalizing well-founded recursion to coinductive domains*

- Fixpoints of contractive maps using *converging equivalence relations* (CERs) $\simeq$ filtered limits.

- Fixpoints of functions with coinductive codomains which are total even though they are not guarded.

- Wellfounded recursion (general recursion) arises as a special case.
Generalized general recursion

- John Matthews (2001)
  *Generalizing well-founded recursion to coinductive domains*

- Fixpoints of contractive maps using *converging equivalence relations* (CERs) $\simeq$ filtered limits.

- Fixpoints of functions with coinductive codomains which are total even though they are not guarded.

- Wellfounded recursion (general recursion) arises as a special case.

- Developed in a classical setting (Isabelle,HOL).
Questions

Applicable in (extensional) Type Theory?

More interesting examples?

Practical? (i.e. better generalized general recursion)

Categorical semantics?

Discovered before?
Questions

• Applicable in (extensional) Type Theory?
Questions

- Applicable in (extensional) Type Theory?
- More interesting examples?
Questions

- Applicable in (extensional) Type Theory?
- More interesting examples?
- Practical? (i.e. *better generalized general recursion*)
Questions

- Applicable in (extensional) Type Theory?
- More interesting examples?
- Practical?
  (i.e. better generalized general recursion)
- Categorical semantics?
Questions

- Applicable in (extensional) Type Theory?
- More interesting examples?
- Practical?
  (i.e. better generalized general recursion)
- Categorical semantics?
- Discovered before?
nth \in [a] \rightarrow Nat \rightarrow a

nth (a:as) 0 = a
nth (a:as) (n+1) = nth as n
The stream CER

CER = Converging equivalence relations.
The stream CER

- CER = Converging equivalence relations.
- We define a CER on $[a]$
  (here Streams over $a$).
The stream CER

- CER = Converging equivalence relations.
- We define a CER on \([a]\) (here Streams over \(a\)).
- We define a family of equivalence relations

\[
i \in \text{Nat} \quad x, y \in [a] \\
x \approx_i y \in \text{Prop}
\]
The stream CER

- CER = Converging equivalence relations.
- We define a CER on \([a]\)
  (here Streams over \(a\)).
- We define a family of equivalence relations

\[
\begin{align*}
  i &\in \text{Nat} & x, y &\in [a] \\
  x &\approx_i y &\in \text{Prop}
\end{align*}
\]

- \[
  i \in \text{Nat} \quad x, y \in [a] \\
  x \approx_i y
\]

\[\iff \forall j \in \text{Nat}. (i < j) \rightarrow \text{nth} x j = \text{nth} y j\]

Generalized general recursion – p. 13/26
The stream CER ...
The stream CER . . .

chain

\[
\begin{array}{c}
i < j \\
x \approx_j y \\
\hline \\
x \approx_i y
\end{array}
\]
The stream CER ...

chain

\[i < j \quad x \approx_j y\]
\[\quad x \approx_i y\]

\[\perp \in [a] \quad \forall x \in [a].x \approx_0 \perp\]
The stream CER . . .

chain

\[ i < j \quad x \approx_j y \]
\[ x \approx_i y \]

0

global limit

\[ \bot \in [a] \quad \forall x \in [a].x \approx_0 \bot \]
The stream CER . . .

chain

\[
\begin{align*}
  i < j & \quad x \approx_j y \\
  & \quad x \approx_i y
\end{align*}
\]

0

\[
\bot \in [a] \quad \forall x \in [a].x \approx_0 \bot
\]

global limit

\[
\begin{align*}
  h \in \text{Nat} & \rightarrow [a] \quad \forall j < j'.h j \approx_j h j' \\
  \lim(h) \in [a] \\
  \forall i \in \text{Nat}.\lim(h) \approx_i h i \\
  (\forall i \in \text{Nat}.x \approx_i h i) & \rightarrow x = \lim(h)
\end{align*}
\]
CERs in general

A CER on a set $A$ is given by
CERs in general

A CER on a set $A$ is given by

- An index set $I$ with a well-founded relation $<$

$$
\frac{i, j \in I}{i < j \in \text{Prop}}
$$
CERs in general

A CER on a set $A$ is given by

- An index set $I$ with a well-founded relation $<$

\[ i, j \in I \quad \frac{i < j \in \text{Prop}}{} \]

- A collection of equivalence relations

\[ i \in I \quad x, y \in A \quad \frac{x \approx_i y \in \text{Prop}}{} \]
CERs in general ...
CERs in general...

chain

\[ \frac{i < j \quad x \approx_j y}{x \approx_i y} \]
CERs in general

\[ \frac{i < j \quad x \approx_j y}{x \approx_i y} \]

\[ h \in I \rightarrow A \quad \forall j < j' < i. h j \approx_j h j' \]

\[ \lim^i(h) \in A \]

\[ \forall k < i. \lim^i(h) \approx_k h k \]

\[ (\forall k < i. x \approx_k h k) \rightarrow x \approx_i \lim^i(h) \]
CERs in general . . .

### Chain

\[
\begin{align*}
  i < j & \quad x \approx_j y \\
  x \approx_i y \\
  h \in I \rightarrow A & \quad \forall j < j' < i. h j \approx_j h j'
\end{align*}
\]

### Local Limit

\[
\begin{align*}
  \lim^i(h) \in A \\
  \forall k < i. \lim^i(h) \approx_k h k \\
  (\forall k < i. x \approx_k h k) \rightarrow x \approx_i \lim^i(h)
\end{align*}
\]

### Global Limit

\[
\begin{align*}
  \lim(h) \in A \\
  \forall k \in I. \lim(h) \approx_k h k \\
  (\forall k \in I. x \approx_k h k) \rightarrow x = \lim(h)
\end{align*}
\]
Differences to Matthews

Matthews hasn't got the uniqueness conditions for limit and global limit.

\[(j < i) \not\vdash x \vdash (4)\]

Derivable from local limit.

\[(j \vdash x \vdash j \not\vdash y) \not\vdash x = y (6)\]

Derivable from global limit.
Differences to Matthews

Matthews hasn’t got the uniqueness conditions for limit and global limit.
Differences to Matthews

- Matthews hasn’t got the uniqueness conditions for limit and global limit.

\[(\forall j. \neg(j < i)) \rightarrow x \approx_i y \quad (4)\]
derivable from local limit.
Differences to Matthews

Matthews hasn’t got the uniqueness conditions for limit and global limit.

\[(\forall j. \neg(j < i)) \rightarrow x \approx_i y\]  \hspace{1cm} (4)
derivable from local limit.

\[(\forall j. x \approx_j y) \rightarrow x = y\]  \hspace{1cm} (6)
derivable from global limit.
A CER on \textbf{Nat} $\rightarrow$ [Nat]

This shows how to lift a CER on $B$ to $A!B$. 
A CER on $\text{Nat} \rightarrow [\text{Nat}]$

\[
\begin{array}{c}
i \in \text{Nat} \quad f, g \in \text{Nat} \rightarrow [a] \\
\begin{array}{c}
f \approx_i g \\
\iff \forall j \in \text{Nat}. (j < i) \rightarrow \\
\forall n \in \text{Nat}. \text{nth}(f n) j = \text{nth}(g n) j
\end{array}
\end{array}
\]

This shows how to lift a CER on $B$ to $A \rightarrow B$. 

Generalized general recursion – p. 18/26
A CER on $\text{Nat} \rightarrow \boxed{\text{Nat}}$

\[
\begin{align*}
  i & \in \text{Nat} & f, g & \in \text{Nat} \rightarrow \boxed{\text{a}} \\
  f \approx_i g \\
  \iff & \quad \forall j \in \text{Nat}. (j < i) \rightarrow \\
  & \quad \forall n \in \text{Nat}. \text{nth}(f \ n) \ j = \text{nth}(g \ n) \ j
\end{align*}
\]

This shows how to lift a CER on $B$ to $A \rightarrow B$. 

Contractive functions

Given a CER on $A$ a function $f \in A \rightarrow A$ is contractive, iff

$$\forall j < i. x \approx_j y \Rightarrow f x \approx_i f y$$
Contractive functions

Given a CER on $A$ a function $f \in A \rightarrow A$ is contractive, iff

$$\forall j < i. x \approx_j y \quad \Rightarrow \quad f x \approx_i f y$$

**Theorem (Matthews):** A contractive function $f \in A \rightarrow A$ has a unique fixpoint $\text{fix}(f) \in A$
Proof sketch

Define $h_i$ using well-founded recursion:

$$h_i = f(lim_{i} h)$$

and show that $h_i$ if $(h_i)$ then define $x(f) = lim(h)$
Proof sketch

Define $h \in I \rightarrow A$ using well founded recursion:

$$h_i = f(\lim^i h)$$
Proof sketch

Define $h \in I \rightarrow A$ using well founded recursion:

$$h_i = f(\lim^i h)$$

and show that

$$h_i \approx^i f(h_i)$$
Proof sketch

Define $h \in I \to A$ using well founded recursion:

$$h_i = f(\lim^i h)$$

and show that

$$h_i \approx_i f(h_i)$$

then define

$$\text{fix}(f) = \lim(h)$$
\[ f \in (\text{Nat} \rightarrow [\text{Nat}]) \rightarrow (\text{Nat} \rightarrow [\text{Nat}]) \]

\[ f \text{ nats} = n : (\text{nats} \ (n+1)) \]

**Observation:** \( f \) is contractive.
\( f \in [\text{Nat}] \rightarrow [\text{Nat}] \)
\( f \text{ ham} = 2 : (\text{merge} \ (\text{map} \ (\lambda \ i \rightarrow 2*i) \ \text{ham}) \ \\
(\text{map} \ (\lambda \ i \rightarrow 3*i) \ \text{ham})) \)

**Observation:** \( f \) is contractive.

**Lemma:**

\[
\frac{h \approx_i h'}{\text{map} \ g \ h \approx_i \text{map} \ g \ h'}
\]

**Lemma:**

\[
\frac{h \approx_i h' \quad g \approx_i g'}{\text{merge} \ h \ g \approx_i \text{merge} \ h' \ g'}
\]
primes

\[ \text{sieve} \in [\text{Nat}] \rightarrow [\text{Nat}] \rightarrow [\text{Nat}] \]
\[ \text{sieve} \ (\text{ns} \ @ \ (n:\text{ns}')) \ (\text{ps} \ @ \ (p:\text{ps}')) \]
\[ \begin{array}{c}
| \ n < p*p \quad = \ n: (\text{sieve} \ \text{ns}' \ \text{primes}) \\
| \ \text{mod n p == 0} \quad = \ \text{sieve} \ \text{ns}' \ \text{primes} \\
| \ \text{otherwise} \quad = \ \text{sieve} \ \text{ns} \ \text{ps}' \\
\end{array} \]

\[ \text{primes} \in [\text{Nat}] \]
\[ \text{primes} = 2 : (\text{sieve} \ (\text{nats} \ 3) \ \text{primes}) \]

Left as an exercise.
Wellfounded recursion

Given: $A \rightarrow B$ where $(A; <)$ is well-ordered.

We define a CER on $A \rightarrow B$:

$a \in A \rightarrow \{f; g\} \in A \rightarrow B \ni f, g$ if $a \in x < f(x) = g(x)$.

Local and global limits:

$\lim (h a) = a : ha$.
Wellfounded recursion

Given:

\[ A \rightarrow B \]

where \((A, <)\) is well-ordered.
Wellfounded recursion

- **Given:**

\[ A \rightarrow B \]

where \( (A, <) \) is well-ordered.

- **We define a CER on** \( A \rightarrow B \):

\[
\frac{a \in A \quad f, g \in A \rightarrow B}{f \approx g \iff \forall x < a. f(x) = g(x)}
\]
Wellfounded recursion

Given:

\[ A \rightarrow B \]

where \((A, <)\) is well-ordered.

We define a CER on \( A \rightarrow B \):

\[
\begin{array}{c}
 a \in A \\
 f, g \in A \rightarrow B \\
 f \approx g \iff \forall x < a. f x = g x
\end{array}
\]

Local and global limits:

\[
\lim(h) = \lambda a. h a a
\]
Wellfounded recursion ...

\[ f \in (A \rightarrow B) \rightarrow (A \rightarrow B) \]
Wellfounded recursion . . .

\[ f \in (A \to B) \to (A \to B) \]

\( f \) contractive:

\[
\forall a < b. h \approx_a h' \Rightarrow fh \approx_b fh'
\]
Wellfounded recursion...

\( f \in (A \to B) \to (A \to B) \)

\( f \) contractive:

\[
\forall a < b. h \approx_a h' \\
\implies f h \approx_b f h'
\]

means

\[
\forall x < a < b. h x = h' x \\
\implies \forall x < b. f h x = f h' x
\]
Wellfounded recursion . . .

\[ f \in (A \to B) \to (A \to B) \]

\( f \) contractive:

\[ \forall a < b. h \approx_a h' \]

\[ fh \approx_b fh' \]

means

\[ \forall x < a < b. h x = h' x \]

\[ \forall x < b. f h x = f h' x \]

that \( f \) uses \( h \) only on smaller arguments.
Wellfounded recursion . . .

\[ f \in (A \to B) \to (A \to B) \]

\( f \) contractive:

\[ \forall a < b. h \approx_a h' \]

\[ fh \approx_b fh' \]

means

\[ \forall x < a < b. h x = h' x \]

\[ \forall x < b. f h x = f h' x \]

that \( f \) uses \( h \) only on smaller arguments.

**Hence**: Contractive \( \implies \) Wellfounded.
Applicable in (extensional) Type Theory?
More interesting examples?
Practical? (i.e. *better generalized general recursion*)
Categorical semantics?
Discovered before?