Generalized general recursion

Thorsten Altenkirch

University of Nottingham

General recursion

```
gcd′ ∈ Nat → Nat → Nat

gcd′ m n
   | m==n = m
   | m<n = gcd′ (m-n) n
   | n<m = gcd′ m (n-m)
```

General recursion . . .

Paulson 86, Nordström 88

$$\frac{f \in \Pi a \in A.(\Pi b \in A.(b < a) \rightarrow B) \rightarrow B}{\mathsf{fix}(f) \in \Pi a \in A.(\mathsf{Acc} \ < \ a) \rightarrow B}$$

where Acc is defined inductively:

$$\frac{\Pi b \in A.(b < a) \rightarrow \mathsf{Acc} \ < \ b}{\mathsf{Acc} \ < \ a}$$

Better general recursion

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.

nats?

$\texttt{nats} \in \texttt{Nat} \rightarrow \texttt{[Nat]}$

```
nats n = n : (nats (n+1))
```

`nats`!

- `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).

ham ?

```
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` ?

- `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 ??

```
sieve ∈ [Nat] → [Nat] → [Nat]
sieve (ns @ (n:ns')) (ps @ (p:ps'))
    | n < p*p      = n:(sieve ns' primes)
    | mod n p == 0 = sieve ns' primes
    | otherwise    = sieve ns ps'

primes ∈ [Nat]
primes = 2 : (sieve (nats 3) primes)
```

Generalized general recursion

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) $\approx$ filtered limits.

Generalized general recursion

- John Matthews (2001)
  Generalizing well-founded recursion to coinductive domains
- Fixpoints of contractive maps using *converging equivalence relations* (CERs) $\approx$ fi ltered 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) ≈ 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) $\approx$ 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

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

```
nth ∈ [a] → Nat → 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 defi ne a CER on `[a]`
  (here Streams over a).

The stream CER

- CER = Converging equivalence relations.
- We defi ne a CER on `[a]`
  (here Streams over a).
- We defi ne a family of equivalence relations

$$\frac{i \in \texttt{Nat} \qquad x, y \in \texttt{[a]}}{x \approx_i y \in \mathbf{Prop}}$$

The stream CER

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

$$\frac{i \in \texttt{Nat} \qquad x, y \in \texttt{[a]}}{x \approx_i y \in \mathbf{Prop}}$$

- $$\frac{i \in \texttt{Nat} \qquad x, y \in \texttt{[a]}}{x \approx_i y}$$

$$\iff \quad \forall j \in \texttt{Nat}.(i < j) \rightarrow \texttt{nth}\, x\, j = \texttt{nth}\, y\, j$$

The stream CER . . .

chain

The stream CER . . .

chain

$$\frac{i < j \qquad x \approx_j y}{x \approx_i y}$$

The stream CER ...

chain

$$\frac{i < j \qquad x \approx_j y}{x \approx_i y}$$

0

$$\bot \in [\mathsf{a}] \qquad \forall x \in [\mathsf{a}]. x \approx_0 \bot$$

The stream CER . . .

chain

$$\frac{i < j \qquad x \approx_j y}{x \approx_i y}$$

0

$$\perp \in [\mathsf{a}] \qquad \forall x \in [\mathsf{a}].x \approx_0 \perp$$

global limit

The stream CER . . .

chain

$$\frac{i < j \qquad x \approx_j y}{x \approx_i y}$$

0

$$\bot \in \texttt{[a]} \qquad \forall x \in \texttt{[a]}. x \approx_0 \bot$$

global limit

$$\frac{h \in \texttt{Nat} \to \texttt{[a]} \qquad \forall j < j'. h\, j \approx_j h\, j'}{\begin{array}{l} \texttt{lim}(h) \in \texttt{[a]} \\ \forall i \in \texttt{Nat}. \texttt{lim}(h) \approx_i h\, i \\ (\forall i \in \texttt{Nat}. x \approx_i h\, i) \to x = \texttt{lim}(h) \end{array}}$$

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 $<$

$$\frac{i, j \in I}{i < j \in \textbf{Prop}}$$

CERs in general

A CER on a set A is given by

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

$$\frac{i, j \in I}{i < j \in \textbf{Prop}}$$

- A collection of equivalence relations

$$\frac{i \in I \qquad x, y \in A}{x \approx_i y \in \textbf{Prop}}$$

CERs in general . . .

CERs in general . . .

chain $$\frac{i < j \qquad x \approx_j y}{x \approx_i y}$$

CERs in general . . .

chain $$\frac{i < j \qquad x \approx_j y}{x \approx_i y}$$

$$\frac{h \in I \to A \qquad \forall j < j' < i. h\, j \approx_j h\, j'}{\begin{array}{l} \mathtt{lim}^i(h) \in A \\ \forall k < i. \mathtt{lim}^i(h) \approx_k h\, k \\ (\forall k < i. x \approx_k h\, k) \to x \approx_i \mathtt{lim}^i(h) \end{array}}$$

local limit

CERs in general . . .

chain
$$\frac{i < j \qquad x \approx_j y}{x \approx_i y}$$

local limit
$$\frac{h \in I \to A \qquad \forall j < j' < i.h\,j \approx_j h\,j'}{\begin{array}{c}\mathtt{lim}^i(h) \in A \\ \forall k < i.\mathtt{lim}^i(h) \approx_k h\,k \\ (\forall k < i.x \approx_k h\,k) \to x \approx_i \mathtt{lim}^i(h)\end{array}}$$

global limit
$$\frac{h \in I \to A \qquad \forall j < j'.h\,j \approx_j h\,j'}{\begin{array}{c}\mathtt{lim}(h) \in A \\ \forall k \in I.\mathtt{lim}(h) \approx_k h\,k \\ (\forall k \in I.x \approx_k h\,k) \to x = \mathtt{lim}(h)\end{array}}$$

Differences to Matthews

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.
- $(\forall j.\neg(j < i)) \to x \approx_i y$ (4) derivable from local limit.

Differences to Matthews

- Matthews hasn't got the uniqueness conditions for limit and global limit.
- $(\forall j.\neg(j < i)) \to x \approx_i y$ (4)
  derivable from local limit.
- $(\forall j.x \approx_j y) \to x = y$ (6)
  derivable from global limit.

A CER on `Nat` $\longrightarrow$ `[Nat]`

A CER on $\texttt{Nat} \rightarrow [\texttt{Nat}]$

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

A CER on $\texttt{Nat} \to [\texttt{Nat}]$

$$\frac{i \in \texttt{Nat} \qquad f, g \in \texttt{Nat} \to [\texttt{a}]}{\begin{array}{l} f \approx_i g \\ \quad \Longleftrightarrow \quad \forall j \in \texttt{Nat}.(j < i) \to \\ \qquad\qquad \forall n \in \texttt{Nat}.\texttt{nth}\,(f\,n)\,j = \texttt{nth}\,(g\,n)\,j \end{array}}$$

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

Contractive functions

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

$$\frac{\forall j < i . x \approx_j y}{f\, x \approx_i f\, y}$$

Contractive functions

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

$$\frac{\forall j < i . x \approx_j y}{f\ x \approx_i f\ y}$$

Theorem (Matthews): A contractive function $f \in A \to A$ has a unique fixpoint $\mathrm{fix}(f) \in A$

Proof sketch

Proof sketch

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

$$h\,i = f(\lim^{i} h)$$

Proof sketch

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

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

and show that

$$h\,i \approx_i f(h\,i)$$

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

$$\mathrm{fix}(f) = \lim(h)$$

nats

`f`$\in$ `(Nat` $\longrightarrow$ `[Nat])` $\longrightarrow$ `(Nat` $\longrightarrow$ `[Nat])`

```
f nats = n : (nats (n+1))
```

Observation: f is contractive.

ham

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

Observation: f is contractive.

Lemma:

$$\frac{h \approx_i h'}{\mathtt{map}\, g\, h \approx_i \mathtt{map}\, g\, h'}$$

Lemma:

$$\frac{h \approx_i h' \qquad g \approx_i g'}{\mathtt{merge}\, h\, g \approx_i \mathtt{merge}\, h'\, g'}$$

primes

```
sieve ∈ [Nat] → [Nat] → [Nat]
sieve (ns @ (n:ns')) (ps @ (p:ps'))
    | n < p*p       = n:(sieve ns' primes)
    | mod n p == 0 = sieve ns' primes
    | otherwise     = sieve ns ps'

primes ∈ [Nat]
primes = 2 : (sieve (nats 3) primes)
```

Left as an exercise.

Wellfounded recursion

Wellfounded recursion

- **Given**:

$$A \to B$$

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

Wellfounded recursion

- **Given**:

$$A \longrightarrow B$$

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

- We define a CER on $A \longrightarrow B$:

$$\frac{a \in A \quad f, g \in A \longrightarrow B}{f \approx g \iff \forall x < a. f\, x = g\, x}$$

Wellfounded recursion

- **Given**:

$$A \to B$$

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

- We define a CER on $A \to B$:

$$\frac{a \in A \quad f, g \in A \to B}{f \approx g \iff \forall x < a. f\, x = g\, x}$$

- Local and global limits:

$$\mathtt{lim}(h) = \lambda a. h\, a\, a$$

Wellfounded recursion . . .

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

Wellfounded recursion ...

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

f contractive:

$$\frac{\forall a < b. h \approx_a h'}{f\,h \approx_b f\,h'}$$

Wellfounded recursion . . .

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

f contractive:

$$\frac{\forall a < b . h \approx_a h'}{f\,h \approx_b f\,h'}$$

means

$$\frac{\forall x < a < b . h\,x = h'\,x}{\forall x < b . f\,h\,x = f\,h'\,x}$$

Wellfounded recursion . . .

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

f contractive:

$$\frac{\forall a < b . h \approx_a h'}{f\, h \approx_b f\, h'}$$

means

$$\frac{\forall x < a < b . h\, x = h'\, x}{\forall x < b . f\, h\, x = f\, h'\, x}$$

that f uses h only on smaller arguments.

Wellfounded recursion . . .

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

f contractive:

$$\frac{\forall a < b . h \approx_a h'}{f\, h \approx_b f\, h'}$$

means

$$\frac{\forall x < a < b . h\, x = h'\, x}{\forall x < b . f\, h\, x = f\, h'\, x}$$

that f uses h only on smaller arguments.

Hence : Contractive $\implies$ Wellfounded.

Back to Questions

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