Simulating Induction-Recursion for Partial Algorithms in Coq

Dominique Larchey-Wendling* (CNRS-LORIA, Nancy, France)
Jean-François Monin (Univ. Grenoble Alpes-VERIMAG, France)

`https://github.com/DmxLarchey/ite-normalisation`

Programming Systems Lab
Saarland University, March 2018

Standard Recursion in Coq

- Structural recursion, rec. calls on sub-terms

$$\texttt{fact}\ \texttt{O} = \texttt{S}\ \texttt{O} \qquad \texttt{fact}\ (\texttt{S}\ n) = \texttt{S}\ n \times \texttt{fact}\ n$$

- Well-founded recursion for $R : X \to X \to \texttt{Prop}$ (module `Wf`)

$\texttt{Inductive Acc}\ R\ x : \texttt{Prop} :=$

$\quad \mid \texttt{Acc\_intro} : (\forall y,\ R\ y\ x \to \texttt{Acc}\ R\ y) \to \texttt{Acc}\ R\ x.$

$\texttt{Fixpoint}\ f\ x\ (H_x : \texttt{Acc}\ R\ x)\ \{\texttt{struct}\ H_x\} := \ldots \quad f\ y\ T_y \quad \ldots$

- Must define R before f, prove $\texttt{Acc}\ R\ x$ and ensure $\boxed{T_y <_{\text{struct}} H_x}$
- Particular case, decreasing measure (using `lt_wf`):
  - $R\ x\ y$ is $m\ x < m\ y$ for some $m : X \to \texttt{nat}$
- This looks like strong constraints for general recursion...

Complicated recursive schemes

- No obvious structural recursion:

$$\texttt{min}\ f\ x = \texttt{if}\ f\ x = 0\ \texttt{then}\ x\ \texttt{else}\ \texttt{min}\ f\ (1 + x)$$

- Nor obvious termination:

$$\texttt{iter}_0\ f\ n = \texttt{if}\ n = 0\ \texttt{then}\ [\,]\ \texttt{else}\ n :: \texttt{iter}_0\ f\ (f\ n)$$

- Complicated termination proof ($succs : X \to \texttt{list}\ X$):

$$\begin{array}{lcll} \texttt{dfs}\ v\ [\,] & = & v & \\ \texttt{dfs}\ v\ (x :: l) & = & \texttt{dfs}\ v\ l & \text{if } x \in_{\texttt{l}} l \\ \texttt{dfs}\ v\ (x :: l) & = & \texttt{dfs}\ (x :: v)\ (succs\ x +\!\!+ l) & \text{if } x \notin_{\texttt{l}} l \end{array}$$

More complicated recursive schemes

- Nesting/mutual recursion:

  McCarthy F91 $\quad f\ x = \texttt{if}\ x > 100\ \texttt{then}\ x - 10\ \texttt{else}\ f\big(f(x+11)\big)$

  Knuth 1991 $\quad f\ x = \texttt{if}\ x > a\ \texttt{then}\ x - b\ \texttt{else}\ f^c(x+d)$

- Nesting and hard termination proof: $\mathbb{U}\ (m \cdot n)\ (m' \cdot n')$ is

$$\begin{cases} \texttt{None} & \text{if } \mathbb{U}\ m\ m' = \texttt{None} \\ \texttt{None} & \text{if } \mathbb{U}\ m\ m' = \texttt{Some}\ \rho \text{ and } \mathbb{U}\ (\rho\ n)\ (\rho\ n') = \texttt{None} \\ \texttt{Some}(\sigma \circ \rho) & \text{if } \mathbb{U}\ m\ m' = \texttt{Some}\ \rho \text{ and } \mathbb{U}\ (\rho\ n)\ (\rho\ n') = \texttt{Some}\ \sigma \end{cases}$$

Beyond Coq structural recursion

- Induction-Recursion (IR) (Bove&Capretta 05)
  - general but IR absent from Coq
  - domain predicates are *informative* (extraction)
- `Program Fixpoint` or Equations (Sozeau 2010)
  - structural recursion or decreasing measure
  - for total functions
- Partial and Nested in HOL (Krauss 09)
  - versatile but non-constructive
  - needs Hilbert's ϵ description operator
- We want *accurate extraction* as well

Larry Paulson's normalization (1985)

$$
\begin{array}{l}
\texttt{let rec nm } e = \texttt{match } e \texttt{ with} \\
\quad \mid \alpha \qquad\qquad\qquad\qquad \to \alpha \\
\quad \mid \omega(\alpha, y, z) \qquad\qquad \to \omega(\alpha, \texttt{nm } y, \texttt{nm } z) \\
\quad \mid \omega(\omega(a, b, c), y, z) \to \texttt{nm}(\omega(a, \texttt{nm}(\omega(b, y, z)), \texttt{nm}(\omega(c, y, z))))
\end{array}
$$

- α is atomic expression, $\omega\ b\ x\ y = \texttt{if } b \texttt{ then } x \texttt{ else } y$
- Pros:
  - recurring example (Giesl 97, B&C 05...)
  - has nested recursion, but compact and meaningful
- Cons:
  - Termination proof is somewhat easy (by a measure)
  - Could be done by `Program Fixpoint`

Inductive-Recursive schemes (Bove&Capretta 05)

$$\texttt{Inductive}\ \ \Omega : \texttt{Set} := \alpha : \Omega \mid \omega : \Omega \to \Omega \to \Omega \to \Omega.$$

$$\begin{aligned}
&\texttt{Inductive}\ \ \mathbb{D} : \Omega \to \boxed{\texttt{Set}} := \\
&\quad \mid \texttt{d_nm_0} && : \mathbb{D}\ \alpha \\
&\quad \mid \texttt{d_nm_1}\ y\ z && : \mathbb{D}\ y \to \mathbb{D}\ z \to \mathbb{D}(\omega\ \alpha\ y\ z) \\
&\quad \mid \texttt{d_nm_2}\ a\ b\ c\ y\ z\ D_b\ D_c && : \mathbb{D}\big(\omega\ a\ (\texttt{nm}\ (\omega\ b\ y\ z)\ D_b)\ (\texttt{nm}\ (\omega\ c\ y\ z)\ D_c)\big) \\
& && \to \mathbb{D}\big(\omega\ (\omega\ a\ b\ c)\ y\ z\big)
\end{aligned}$$

$$\begin{aligned}
&\texttt{with Fixpoint}\ \ \texttt{nm}\ e\ (D_e : \mathbb{D}\ e)\ \{\texttt{struct}\ D_e\} : \Omega := \texttt{match}\ D_e\ \texttt{with} \\
&\quad \mid \texttt{d_nm_0} && \mapsto \alpha \\
&\quad \mid \texttt{d_nm_1}\ y\ z\ D_y\ D_z && \mapsto \omega\ \alpha\ (\texttt{nm}\ y\ D_y)\ (\texttt{nm}\ z\ D_z) \\
&\quad \mid \texttt{d_nm_2}\ a\ b\ c\ y\ z\ D_b\ D_c\ D_a && \mapsto \texttt{nm}\ \big(\omega\ a\ (\texttt{nm}\ (\omega\ b\ y\ z)\ D_b)\ (\texttt{nm}\ (\omega\ c\ y\ z)\ D_c)\big)\ D_a \\
&\texttt{end.}
\end{aligned}$$

- Coq does not have IR yet (but this might evolve shortly ?)
- The domain $\mathbb{D} : \Omega \to \texttt{Set}$ is *informative* (kept at extraction)

The graph method (Bove 09)

- a recursive scheme like `nm` induces a graph $\mathbb{G} : \Omega \to \Omega \to \texttt{Prop}$

$$\frac{}{\mathbb{G}\ \alpha\ \alpha} \qquad \frac{\mathbb{G}\ y\ n_y \qquad \mathbb{G}\ z\ n_z}{\mathbb{G}\ (\omega\ \alpha\ y\ z)\ (\omega\ \alpha\ n_y\ n_z)}$$

$$\frac{\mathbb{G}\ (\omega\ b\ y\ z)\ n_b \qquad \mathbb{G}\ (\omega\ c\ y\ z)\ n_c \qquad \mathbb{G}\ (\omega\ a\ n_b\ n_c)\ n_a}{\mathbb{G}\ (\omega\ (\omega\ a\ b\ c)\ y\ z)\ n_a}$$

- She defines the *informative* domain predicate $\mathbb{D}\ e := \{n \mid \mathbb{G}\ e\ n\}$
- $\mathbb{D} : \Omega \to \boxed{\texttt{Set}}$ is then used for $\texttt{nm}\ e\ (D_e : \mathbb{D}\ e) := \pi_1(D_e)$
  - and $\texttt{nm\_spec}\ e\ D_e : \mathbb{G}\ e\ (\texttt{nm}\ e\ D_e)$ as $\pi_2(D_e)$
- Problem: computational content of `nm` equals that of D_e
  - extracted code for $\texttt{nm}\ e$ depends on the proof term $D_e : \mathbb{D}\ e$
  - establishing $\mathbb{D}\ e$ must follow strict rules to get the good code

We obtain this simulated IR scheme

$$
\begin{array}{l}
\texttt{Inductive}\ \ \mathbb{D} : \Omega \to \boxed{\texttt{Prop}} := \\
\quad \mid \texttt{d\_nm\_0} \qquad\qquad\qquad\qquad\quad : \ \mathbb{D}\ \alpha \\
\quad \mid \texttt{d\_nm\_1}\ y\ z \qquad\qquad\qquad\quad : \ \mathbb{D}\ y \to \mathbb{D}\ z \to \mathbb{D}(\omega\ \alpha\ y\ z) \\
\quad \mid \texttt{d\_nm\_2}\ a\ b\ c\ y\ z\ D_b\ D_c \qquad : \ \mathbb{D}\big(\omega\ a\ (\texttt{nm}\ (\omega\ b\ y\ z)\ D_b)\ (\texttt{nm}\ (\omega\ c\ y\ z)\ D_c)\big) \\
\qquad\qquad\qquad\qquad\qquad\qquad\qquad\quad \to \mathbb{D}\big(\omega\ (\omega\ a\ b\ c)\ y\ z\big) \\
\texttt{with Fixpoint}\ \ \texttt{nm}\ e\ (D_e : \mathbb{D}\ e) : \Omega := \texttt{match}\ D_e\ \texttt{with} \\
\quad \mid \texttt{d\_nm\_0} \qquad\qquad\qquad\qquad\quad \mapsto \alpha \\
\quad \mid \texttt{d\_nm\_1}\ y\ z\ D_y\ D_z \qquad\qquad\quad \mapsto \omega\ \alpha\ (\texttt{nm}\ y\ D_y)\ (\texttt{nm}\ z\ D_z) \\
\quad \mid \texttt{d\_nm\_2}\ a\ b\ c\ y\ z\ D_b\ D_c\ D_a \mapsto \texttt{nm}\ \big(\omega\ a\ (\texttt{nm}\ (\omega\ b\ y\ z)\ D_b)\ (\texttt{nm}\ (\omega\ c\ y\ z)\ D_c)\big)\ D_a \\
\texttt{end.}
\end{array}
$$

- The domain $\mathbb{D} : \Omega \to \texttt{Prop}$ is *non-informative* (good extraction)
- $\texttt{nm} : \forall e,\ \mathbb{D}\ e \to \Omega$ is *proof-irrelevant*, i.e. $\texttt{nm}\ x\ D_1 = \texttt{nm}\ x\ D_2$
- We get constructors, a dep. elim. scheme and fixpoint eqs

The graph and the domain revisited

- Recall `nm` induces a graph $\mathbb{G} : \Omega \to \Omega \to \texttt{Prop}$

$$\frac{}{\mathbb{G}\ \alpha\ \alpha} \quad \cdots \quad \frac{\mathbb{G}\ (\omega\ b\ y\ z)\ n_b \quad \mathbb{G}\ (\omega\ c\ y\ z)\ n_c \quad \mathbb{G}\ (\omega\ a\ n_b\ n_c)\ n_a}{\mathbb{G}\ (\omega\ (\omega\ a\ b\ c)\ y\ z)\ n_a}$$

- $\mathbb{G}$ is functional: $\texttt{g\_nm\_fun} : \forall\, e\ n_1\ n_2,\ \mathbb{G}\ e\ n_1 \to \mathbb{G}\ e\ n_2 \to n_1 = n_2$
- we define the *non-informative* domain $\boxed{\mathbb{D}\ e := \exists n, \mathbb{G}\ e\ n}$
- inductive structure on $\mathbb{D} : \Omega \to \texttt{Prop}$ provided by the `nm` scheme
  - for $e \in \mathbb{D}$, $m(e) =$ number of rec. calls to compute `nm` e
  - well-founded relation of $\mathbb{D}$: $R\ x\ y$ iff $m(x) < m(y)$
  - Problem: defining $m : \mathbb{D} \to \texttt{nat}$ as hard as defining $\texttt{nm} : \mathbb{D} \to \Omega$

The inductive structure of $\mathbb{D} : \Omega \to \texttt{Prop}$ (i)

- Non-informative capture by accessibility predicates (`Wf`)
- The relation $<_{\mathbb{R}} : \Omega \to \Omega \to \texttt{Prop}$ relates calls to sub-calls

$$\frac{}{y <_{\mathbb{R}} \omega\ \alpha\ y\ z} \qquad \frac{}{z <_{\mathbb{R}} \omega\ \alpha\ y\ z} \qquad \frac{}{\omega\ b\ y\ z <_{\mathbb{R}} \omega\ (\omega\ a\ b\ c)\ y\ z}$$

$$\frac{\mathbb{G}\ (\omega\ b\ y\ z)\ n_b \qquad \mathbb{G}\ (\omega\ c\ y\ z)\ n_c}{\omega\ a\ n_b\ n_c <_{\mathbb{R}} \omega\ (\omega\ a\ b\ c)\ y\ z} \qquad \frac{}{\omega\ c\ y\ z <_{\mathbb{R}} \omega\ (\omega\ a\ b\ c)\ y\ z}$$

- $\mathbb{D} = \texttt{fun}\ e \mapsto \exists n, \mathbb{G}\ e\ n$ is well-founded: $\forall e, \mathbb{D}\ e \to \texttt{Acc} <_{\mathbb{R}} e$
- We define `nm_rec` by structural induction on A_e:

  $\texttt{Fixpoint nm\_rec}\ e\ (A_e : \texttt{Acc} <_{\mathbb{R}} e)\ \{\texttt{struct}\ A_e\} : \{n \mid \mathbb{G}\ e\ n\} := \ldots$

The inductive structure of $\mathbb{D} : \Omega \to \texttt{Prop}$ (ii)

- Capture $\mathbb{D}$ as bar inductive predicate

$$\frac{}{\mathbb{G}\ \alpha\ \alpha} \qquad \frac{\mathbb{G}\ y\ n_y \qquad \mathbb{G}\ z\ n_z}{\mathbb{G}\ (\omega\ \alpha\ y\ z)\ (\omega\ \alpha\ n_y\ n_z)}$$

$$\frac{\mathbb{G}\ (\omega\ b\ y\ z)\ n_b \qquad \mathbb{G}\ (\omega\ c\ y\ z)\ n_c \qquad \mathbb{G}\ (\omega\ a\ n_b\ n_c)\ n_a}{\mathbb{G}\ (\omega\ (\omega\ a\ b\ c)\ y\ z)\ n_a}$$

- $\mathbb{D} : \Omega \to \texttt{Prop}$ inductively defined:

$$\frac{}{\mathbb{D}\ \alpha} \qquad \frac{\mathbb{D}\ y \qquad \mathbb{D}\ z}{\mathbb{D}\ (\omega\ \alpha\ y\ z)} \qquad \frac{\begin{array}{c}\mathbb{D}\ (\omega\ b\ y\ z) \qquad \mathbb{D}\ (\omega\ c\ y\ z) \\ \forall n_b\, n_c, \mathbb{G}\ (\omega\ b\ y\ z)\ n_b \to \mathbb{G}\ (\omega\ c\ y\ z)\ n_c \to \mathbb{D}\ (\omega\ a\ n_b\ n_c)\end{array}}{\mathbb{D}\ (\omega\ (\omega\ a\ b\ c)\ y\ z)}$$

- We define `nm_rec` by structural induction on D_e:

$$\texttt{Fixpoint nm\_rec}\ e\ (D_e : \mathbb{D}\ e)\ \{\texttt{struct}\ D_e\} : \{n \mid \mathbb{G}\ e\ n\} := \ldots$$

The definition of nm_rec : $\forall e,\ \mathbb{D}\ e \to \{n \mid \mathbb{G}\ e\ n\}$

```
Let nm_rec : ∀e, 𝔻 e → {n | 𝔾 e n}.
refine(fix loop e D_e {struct D_e} :=
  match e as e' return 𝔻 e' → {n | 𝔾 e' n} with
    | α                 ↦ fun D ↦     exist _ α 𝔾₀?
    | ω α y z           ↦ fun D ↦ let (n_y, H_y) := loop y 𝕋_y? in
                                  let (n_z, H_z) := loop z 𝕋_z?
                                  in  exist _ (ω α n_y n_z) 𝔾₁?
    | ω (ω a b c) y z ↦ fun D ↦ let (n_b, H_b) := loop (ω b y z) 𝕋_b? in
                                  let (n_c, H_c) := loop (ω c y z) 𝕋_c? in
                                  let (n_a, H_a) := loop (ω a n_b n_c) 𝕋_a?
                                  in  exist _ n_a 𝔾₂?
  end D_e); simpl in *.
Proof. of certificates 𝕋_y?, 𝕋_z?, 𝕋_b?, 𝕋_c?, 𝕋_a? and post-conditions 𝔾₀?, 𝔾₁?, 𝔾₂? Qed.
```

Proof obligations (non-informative part)

- Post-conditions by the constructors of $\mathbb{G}$

$$\mathbb{G}_0^? \;//\; \ldots \vdash \mathbb{G}\ \alpha\ \alpha$$
$$\mathbb{G}_1^? \;//\; \ldots, H_y : \mathbb{G}\ y\ n_y, H_z : \mathbb{G}\ z\ n_z \vdash \mathbb{G}\ (\omega\ \alpha\ y\ z)\ (\omega\ \alpha\ n_y\ n_z)$$
$$\mathbb{G}_2^? \;//\; \ldots, H_b : \mathbb{G}\ (\omega\ b\ y\ z)\ n_b, H_c : \mathbb{G}\ (\omega\ c\ y\ z)\ n_c, \ldots$$
$$\ldots H_a : \mathbb{G}\ (\omega\ a\ n_b\ n_c)\ n_a \vdash \mathbb{G}\ (\omega\ (\omega\ a\ b\ c)\ y\ z)\ n_a$$

- Termination certificates by `inversion` lemmas

$$\mathbb{T}_y^? \;//\; \ldots, D : \mathbb{D}\ (\omega\ \alpha\ y\ z) \vdash \mathbb{D}\ y$$
$$\mathbb{T}_b^? \;//\; \ldots, D : \mathbb{D}\ (\omega\ (\omega\ a\ b\ c)\ y\ z) \vdash \mathbb{D}\ (\omega\ b\ y\ z)$$
$$\mathbb{T}_a^? \;//\; \ldots, D : \mathbb{D}\ (\omega\ (\omega\ a\ b\ c)\ y\ z), H_b : \mathbb{G}\ (\omega\ b\ y\ z)\ n_b, \ldots$$
$$\ldots H_c : \mathbb{G}\ (\omega\ c\ y\ z)\ n_c \vdash \mathbb{D}\ (\omega\ a\ n_b\ n_c)$$

- beware of $\boxed{\text{structural decrease}}$ in termination certificates

Reification of the graph predicate $\mathbb{G} : \Omega \to \Omega \to \texttt{Prop}$

- with `nm_rec` we obtain a *reifier* of the graph $\mathbb{G}$

$$(\exists n, \mathbb{G}\ e\ n) \to \left(\frac{\texttt{Acc} <_{\mathbb{R}} e}{\mathbb{D}\ e} \right) \to \{n \mid \mathbb{G}\ e\ n\}$$

- hence we can show: $\exists n, \mathbb{G}\ e\ n$ iff $\texttt{Acc} <_{\mathbb{R}} e$ iff $\mathbb{D}\ e$
- we define $\texttt{nm}\ e\ D_e := \pi_1(\texttt{nm\_rec}\ e\ D_e)$
- we get $\texttt{nm\_spec}\ e\ D_e : \mathbb{G}\ e\ (\texttt{nm}\ e\ D_e)$ as $\pi_2(\texttt{nm\_rec}\ e\ D_e)$
  - with simulated constructors for $\mathbb{D}$
  - with dependent elimination scheme for $\mathbb{D}$
  - proof irrelevance for `nm`, fixpoints equations for `nm`

Extraction independent of the domain

- In $\texttt{nm}\ e\ (D_e : \mathbb{D}\ e) : \Omega$, extraction erases the term $D_e : \mathbb{D}\ e : \texttt{Prop}$
- Hence `Extraction nm` gives the intended term:

$$
\begin{array}{l}
\texttt{let rec nm } e = \texttt{match } e \texttt{ with} \\
\quad \mid \alpha \qquad\qquad\quad \to \alpha \\
\quad \mid \omega(x,y,z) \quad \to \texttt{match } x \texttt{ with} \\
\qquad \mid \alpha \qquad\qquad \to \omega(\alpha, \texttt{nm } y, \texttt{nm } z) \\
\qquad \mid \omega(a,b,c) \to \texttt{nm}(\omega(a, \texttt{nm}(\omega(b,y,z)), \texttt{nm}(\omega(c,y,z))))
\end{array}
$$

- the proof of $D_e : \mathbb{D}\ e$ can be provided:
  - after both $\mathbb{D} : \Omega \to \texttt{Prop}$ and $\texttt{nm} : \forall e,\ \mathbb{D}\ e \to \Omega$ are defined
  - by any means necessary w/o consequences on extracted code

Dependent elimination scheme for $\mathbb{D} : \Omega \to \texttt{Prop}$

$\texttt{Fact d\_nm\_0} : \mathbb{D}\ \alpha.$

$\texttt{Fact d\_nm\_1} : \forall\, y\ z,\ \mathbb{D}\ y \to \mathbb{D}\ z \to \mathbb{D}\,(\omega\ \alpha\ y\ z).$

$\texttt{Fact d\_nm\_2} : \forall\, a\ b\ c\ y\ z\ \big(D_b : \mathbb{D}\,(\omega\ b\ y\ z)\big)\ \big(D_c : \mathbb{D}\,(\omega\ c\ y\ z)\big),$
$\mathbb{D}\,\big(\omega\ a\ (\texttt{nm}\ \_\ D_b)\ (\texttt{nm}\ \_\ D_c)\big) \to \mathbb{D}\,\big(\omega\ (\omega\ a\ b\ c)\ y\ z\big).$

$\texttt{Variables}\ \big(P \quad : \forall e,\ \mathbb{D}\ e \to \texttt{Type}\big)$
$\big(HP_i\ : \forall\, e\ D_1\ D_2,\ P\ e\ D_1 \to P\ e\ D_2\big)$
$\big(HP_0 : P\ \alpha\ \texttt{d\_nm\_0}\big)$
$\big(HP_1 : \forall\, y\ z\ D_y\ (\_ : P\ \_\ D_y)\ D_y\ (\_ : P\ \_\ D_z), P\ \_\ (\texttt{d\_nm\_1}\ y\ z\ D_y\ D_z)\big)$
$\big(HP_2 : \forall\, a\ b\ c\ y\ z\ D_b\ (\_ : P\ \_\ D_b)\ D_c\ (\_ : P\ \_\ D_c)\ \ldots$
$\ldots\ D_a\ (\_ : P\ \_\ D_a), P\ \_\ (\texttt{d\_nm\_2}\ a\ b\ c\ y\ z\ D_b\ D_c\ D_a)\big)$

$\texttt{Theorem}\quad \texttt{d\_nm\_rect} : \forall e,\ \mathbb{D}\ e \to P\ e\ D_e.$

- Hypothesis HP_i restricts to proof-irrelevant properties

Proof-irrelevance and fixpoint equations

$\texttt{Fact nm\_pirr} : \forall\, e\ D_1\ D_2,\ \texttt{nm}\ e\ D_1 = \texttt{nm}\ e\ D_2.$

$\texttt{Fact nm\_fix\_0} : \texttt{nm}\ \alpha\ \texttt{d\_nm\_0} = \alpha.$

$\texttt{Fact nm\_fix\_1} : \forall\, y\ z\ D_y\ D_z,$

$\texttt{nm}\ (\omega\ \alpha\ y\ z)\ (\texttt{d\_nm\_1}\ y\ z\ D_y\ D_z)$

$= \omega\ \alpha\ (\texttt{nm}\ y\ D_y)\ (\texttt{nm}\ z\ D_z).$

$\texttt{Fact nm\_fix\_2} : \forall\, a\ b\ c\ y\ z\ D_b\ D_c\ D_a,$

$\texttt{nm}\ (\omega\ (\omega\ a\ b\ c)\ y\ z)\ (\texttt{d\_nm\_2}\ a\ b\ c\ y\ z\ D_b\ D_c\ D_a)$

$= \texttt{nm}\ (\omega\ a\ (\texttt{nm}\ (\omega\ b\ y\ z)\ D_b)\ (\texttt{nm}\ (\omega\ c\ y\ z)\ D_c))\ D_a.$

- With `nm_pirr`, `d_nm_rect` is sufficient for partial correction:
  - `nm_normal` : $\forall\, e\ (D_e : \mathbb{D}\ e),\ \texttt{normal}\ (\texttt{nm}\ e\ D_e)$
  - `nm_equiv` : $\forall\, e\ (D_e : \mathbb{D}\ e),\ e \simeq_\Omega \texttt{nm}\ e\ D_e$
  - `nm_dec` : $\forall\, e\ (D_e : \mathbb{D}\ e),\ |\texttt{nm}\ e\ D_e| \leqslant |e|$ ($|e|$ defined next slide)

Totality of $\mathbb{D}$ / Termination of `nm`

- As in Giesl 97, we define $|\cdot| : \Omega \to \texttt{nat}$ as:
  - $|\alpha| = 1$ and $|\omega\ x\ y\ z| = |x| \times (1 + |y| + |z|)$
- We show $\texttt{d\_nm\_total} : \forall e,\ \mathbb{D}\ e$ by induction on $|e|$
  - we use $\texttt{nm\_dec} : \forall\, e\ (D_e : \mathbb{D}\ e),\ |\texttt{nm}\ e\ D_e| \leqslant |e|$
  - $|\omega\ x\ y\ z| \leqslant |\omega\ x'\ y'\ z'|$ when $|x| \leqslant |x'|$, $|y| \leqslant |y'|$, $|z| \leqslant |z'|$
  - and $|\omega\ u\ y\ z| < |\omega\ v\ y\ z|$ when $|u| < |v|$
  - and $|y| < |\omega\ x\ y\ z|$ and $|z| < |\omega\ x\ y\ z|$
  - and $|\omega\ a\ (\omega\ b\ y\ z)\ (\omega\ c\ y\ z)| < |\omega\ (\omega\ a\ b\ c)\ y\ z|$
- Partial correction / termination independently of definition

$$\texttt{paulson\_nm} : \forall e : \Omega,\ \{n_e : \Omega \mid e \simeq_\Omega n_e \wedge \texttt{normal}\ e\}$$

Other Example, Knuth F91 (1991) (i)

$$\texttt{let rec } f\ x \;\; = \texttt{if } x \leqslant a \texttt{ then } f^c(x+d) \texttt{ else } x-b$$

$$\texttt{and } f^n\ x = \texttt{if } n = 0 \texttt{ then } x \texttt{ else } f^{n-1}(f\ x)$$

$$\texttt{Inductive df} : \texttt{nat} \to \texttt{Prop} :=$$

$$\mid \texttt{df\_0}\ x \qquad\qquad : \; a < x \to \texttt{df}\ x$$

$$\mid \texttt{df\_1}\ x \qquad\qquad : \; x \leqslant a \to \texttt{df}^*\ c\ (x+d)$$

$$\texttt{with} \qquad \texttt{df}^* : \texttt{nat} \to \texttt{nat} \to \texttt{Prop} :=$$

$$\mid \texttt{df\_0}^*\ x \qquad\qquad : \; \texttt{df}^*\ 0\ x$$

$$\mid \texttt{df\_1}^*\ n\ x\ (D_x : \texttt{df}\ x) \;\; : \; \texttt{df}^*\ n\ (f\ x\ D_x) \to \texttt{df}^*\ (\texttt{S}\ n)\ x$$

$$\texttt{with Fixpoint } f\ x\ (D : \texttt{df}\ x) : \texttt{nat} := \texttt{match } D \texttt{ with}$$

$$\mid \texttt{df\_0}\ x\ H_x \qquad\qquad \mapsto x - b$$

$$\mid \texttt{df\_1}\ x\ H_x\ D_c \qquad\qquad \mapsto f^*\ c\ (x+d)\ D_c \qquad\qquad \texttt{end}$$

$$\texttt{with Fixpoint } f^*\ n\ x\ (D : \texttt{df}^*\ n\ x) : \texttt{nat} := \texttt{match } D \texttt{ with}$$

$$\mid \texttt{df\_0}^*\ x \qquad\qquad \mapsto x$$

$$\mid \texttt{df\_1}^*\ n\ x\ D_x\ D_n \qquad\qquad \mapsto f^*\ n\ (f\ x\ D_x)\ D_n \qquad\qquad \texttt{end.}$$

Other Example, Knuth F91 (1991) (ii)

- We ignore the case $c = 0$ (there is no recursion)
- For $d \leqslant b(c-1)$, f is partial:
  - For any $\mathbb{G} : \texttt{nat} \to \texttt{nat} \to \texttt{Prop}$ closed under

$$\frac{a < x}{\mathbb{G}\ x\ (x-b)} \qquad \frac{x \leqslant a \quad \mathbb{G}^c\ (x+d)\ y}{\mathbb{G}\ x\ y}$$

  - Graph induction: $\texttt{Thm}\ \texttt{f\_graph\_ind} : \forall\, x\ D_x, \mathbb{G}\ x\ (f\ x\ D_x)$
  - use it for $\mathbb{G}\ x\ y$ iff $a < x \wedge y = x - b$
  - much simpler than Knuth proof, stronger characterization
- For $b(c-1) < d$, f is total and solution of:
  - $f\ x = \texttt{if}\ x \leqslant a\ \texttt{then}\ f\ (x + d - b(c-1))\ \texttt{else}\ x - b$

Other Examples

- DFS, unification: difficult termination argument
- Huet&Hullot list reversal: mutual induction
- Matrix diagonal (dependently typed)
- minimization and μ-recursive functions (a "compact" interpreter)

Perspectives

- complete the bestiary
- automation, what is the best user interface (IR or graph)?

Depth First Search (nice but not nested)

$$\begin{array}{l}
\texttt{Inductive ddfs} : \texttt{list } X \to \texttt{list } X \to \texttt{Prop} := \\
\quad | \ \texttt{ddfs_0 } v \qquad\quad : \texttt{ddfs } v \texttt{ nil} \\
\quad | \ \texttt{ddfs_1 } v\ x\ l \qquad : x \in_{\texttt{l}} v \to \texttt{ddfs } v\ l \to \texttt{ddfs } v\ (x :: l) \\
\quad | \ \texttt{ddfs_2 } v\ x\ l \qquad : x \notin_{\texttt{l}} v \to \texttt{ddfs } (x :: v)\ (succs\ x +\!\!+ l) \to \texttt{ddfs } v\ (x :: l) \\
\texttt{with Fixpoint dfs } v\ l\ (D : \texttt{ddfs } v\ l) : \texttt{list } X := \texttt{match } D \texttt{ with} \\
\quad | \ \texttt{ddfs_0 } v \qquad\quad \mapsto v \\
\quad | \ \texttt{ddfs_1 } v\ x\ l\ _\ D \mapsto \texttt{dfs } v\ l\ D \\
\quad | \ \texttt{ddfs_2 } v\ x\ l\ _\ D \mapsto \texttt{dfs } (x :: v)\ (succs\ x +\!\!+ l)\ D \\
\texttt{end.}
\end{array}$$

- for $X : \texttt{Type}$ and $succs : X \to \texttt{list } X$
- $\mathbb{I}\ v\ l\ m := v +\!\!+ l \subseteq_{\texttt{l}} m \wedge \forall x, x \in_{\texttt{l}} m \to (x \in_{\texttt{l}} v \vee succs\ x \subseteq_{\texttt{l}} m)$
- $\texttt{dfs\_correct}\ v\ l\ D : \mathbb{I}\ v\ l\ (\texttt{dfs}\ v\ l\ D) \wedge \forall m, \mathbb{I}\ v\ l\ m \to \texttt{dfs}\ D \subseteq_{\texttt{l}} m$
- $\texttt{dfs\_domain}\ v\ l : \texttt{ddfs}\ v\ l \iff \exists m, \mathbb{I}\ v\ l\ m$

μ-recursive algos extraction (improves ITP'17)

Thm ra_compute k $(a : \texttt{recalg}\ k)$ $\vec{v} : (\exists x, [\![a]\!]\ \vec{v}\ x) \to \{x \mid [\![a]\!]\ \vec{v}\ x\}$

```
let min_compute f =
    let rec loop n = match f n with O → n |  S _ → loop (S n) in loop O
(** val ra_compute : recalg → nat vec → nat **)
let rec ra_compute a v⃗ =
    match a with
        | Ra_cst n        → n
        | Ra_zero         → O
        | Ra_succ         → S v⃗_0
        | Ra_proj p       → v⃗_p
        | Ra_comp (f, g⃗)  → ra_compute f (vec_compute (fun h → ra_compute h v⃗) g⃗)
        | Ra_prim (f, g)  → prim_compute (fun w⃗ → ra_compute f w⃗)
                                (fun x y w⃗ → ra_compute g (x#y#w⃗)) (vtail v⃗) v⃗_0
        | Ra_min f        → min_compute (fun x → ra_compute f (x#v⃗))
```