Decorating natural deduction

Helmut Schwichtenberg
(j.w.w. Diana Ratiu)

Mathematisches Institut, LMU, München

Dipartimento di Informatica, Università degli Studi di Verona,
March 14 & 15, 2016

- Proofs may have computational content, which can be extracted (via realizability).
- Proofs can be modified, for adadption of the extracted program to a changed specification.
- Proofs can be transformed and/or decorated, for efficiency of the extracted program.
- Proofs (as opposed to programs) can easily be checked for correctness.

derivation	proof term
$\dfrac{\begin{array}{c}[u\colon A]\\ \mid M\\ B\end{array}}{A \to B} \to^{+} u$	$(\lambda_{u^A} M^B)^{A \to B}$
$\dfrac{\begin{array}{cc}\mid M & \mid N\\ A \to B & A\end{array}}{B} \to^{-}$	$(M^{A \to B} N^A)^B$

derivation	proof term
$\dfrac{\begin{array}{c}\vert\, M \\ A\end{array}}{\forall_x A}\,\forall^+ x \quad \text{(var. cond.)}$	$(\lambda_x M^A)^{\forall_x A}$ (var. cond.)
$\dfrac{\begin{array}{c}\vert\, M \\ \forall_x A(x)\end{array} \qquad r}{A(r)}\,\forall^-$	$(M^{\forall_x A(x)}\, r)^{A(r)}$

$$\dfrac{\dfrac{\dfrac{u\colon \forall_x(A \to Px) \qquad x}{A \to Px}\,\forall^- \qquad v\colon A}{Px}\,\to^-}{\forall_x Px}\,\forall^+ x}{A \to \forall_x Px}\,\to^+ v}{\forall_x(A \to Px) \to A \to \forall_x Px}\,\to^+ u$$

Variable condition: x is not free in A, $\forall_x(A \to Px)$

$$\lambda_{u,v,x}(uxv)$$

An unnecessary detour via implication:

$$\dfrac{\dfrac{\begin{array}{c}[A]\\ \mid M\\ B\end{array}}{A \to B}\to^{+} \qquad \begin{array}{c}\mid N\\ A\end{array}}{B}\to^{-} \qquad \text{reduces to} \qquad \begin{array}{c}\mid N\\ A\\ \mid M\\ B\end{array}$$

Under the Curry-Howard correspondence:

$$(\lambda_u M(u))N \quad \text{reduces to} \quad M(N)$$

or in more detail

$$(\lambda_u M(u^A)^B)^{A\to B} N^A \quad \mapsto_\beta \quad M(N^A)^B.$$

An unnecessary detour via universal quantification:

$$\dfrac{\dfrac{\begin{array}{c}\mid M(x)\\ A(x)\end{array}}{\forall_x A(x)}\,\forall^+ x \qquad r}{A(r)}\,\forall^- \qquad \text{reduces to} \qquad \begin{array}{c}\mid M(r)\\ A(r)\end{array}$$

Under the Curry-Howard correspondence:

$$(\lambda_x M(x))r \quad \text{reduces to} \quad M(r)$$

or in more detail

$$(\lambda_x M(x^\rho)^{A(x)})^{\forall_x A(x)} r^\rho \quad \mapsto_\beta \quad M(r)^{A(r)}.$$

Extend minimal logic by $\exists$, via

$$\exists^+ : \forall_x(A \to \exists_x A),$$
$$\exists^- : \exists_x A \to \forall_x(A \to B) \to B \quad (x \text{ not free in } B).$$

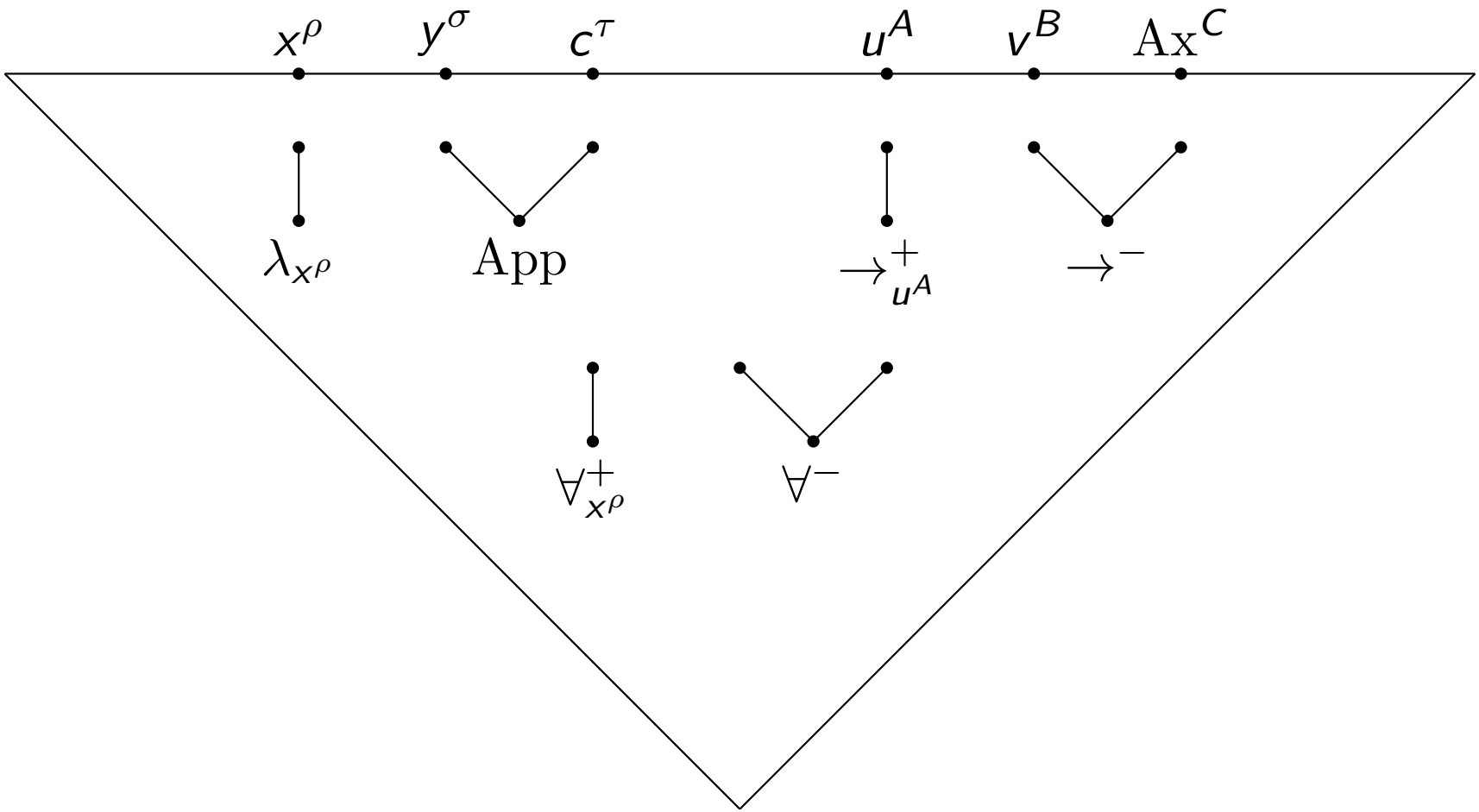
Given $M : \exists_x A(x)$ closed. Normal form of M:

$$\exists^+ r u^{A(r)}.$$

Witness r can be read off.

To make this uniform (in parameters of $\exists_x A(x)$) requires the technique of realizability.

Proof terms in natural deduction



The realizability interpretation transforms such a proof term directly into an object term.

1. Logic
2. The model of partial continuous functionals
3. Formulas as problems
4. Computational content of proofs
5. Decorating proofs

Let A, B, C be propositional variables.

$$(A \to B \to C) \to (A \to B) \to A \to C.$$

Informal proof.

Assume $A \to B \to C$. To show: $(A \to B) \to A \to C$. So assume $A \to B$. To show: $A \to C$. So finally assume A. To show: C. Using the third assumption twice we have $B \to C$ by the first assumption, and B by the second assumption. From $B \to C$ and B we then obtain C. Then $A \to C$, cancelling the assumption on A; $(A \to B) \to A \to C$ cancelling the second assumption; and the result follows by cancelling the first assumption. □

$$\dfrac{\dfrac{\dfrac{u : A \to B \to C \qquad w : A}{B \to C} \qquad \dfrac{v : A \to B \qquad w : A}{B}}{C}\,{\to}^{+}w}{A \to C}\,{\to}^{+}v}{(A \to B) \to A \to C}\,{\to}^{+}u$$

```
(load "~/git/minlog/init.scm")
(add-pvar-name "A" "B" "C" (make-arity))
```

The proof is generated by the following sequence of commands:

```
(set-goal "(A -> B -> C) -> (A -> B) -> A -> C")
(assume "u" "v" "w")
(use "u")
(use "w")
(use "v")
(use "w")

(proof-to-expr-with-formulas (current-proof))

u73: A -> B -> C
v74: A -> B
w75: A

(lambda (u73)
  (lambda (v74) (lambda (w75) ((u73 w75) (v74 w75)))))
```

Let P be a unary predicate variable.

$$\forall_x(A \to Px) \to A \to \forall_x Px.$$

Informal proof.

Assume $\forall_x(A \to Px)$. To show: $A \to \forall_x Px$. So assume A. To show: $\forall_x Px$. Let x be arbitrary; note that we have not made any assumptions on x. To show: Px. We have $A \to Px$ by the first assumption. Hence also Px by the second assumption. Hence $\forall_x Px$. Hence $A \to \forall_x Px$, cancelling the second assumption. Hence the result, cancelling the first assumption. □

$$\dfrac{\dfrac{\dfrac{u\colon \forall_x(A \to Px) \qquad x}{A \to Px} \qquad v\colon A}{Px}}{\dfrac{\dfrac{\forall_x Px}{A \to \forall_x Px}\,{\to}^+ v}{\forall_x(A \to Px) \to A \to \forall_x Px}\,{\to}^+ u}\,\forall^+ x$$

Note that the variable condition is satisfied: x is not free in A (and also not free in $\forall_x(A \to Px)$).

```
(add-var-name "x" (py "alpha"))
(add-pvar-name "P" (make-arity (py "alpha")))
```

The proof is generated by the following sequence of commands:

```
(set-goal "all x(A -> P x) -> A -> all x P x")
(assume "u" "v" "x")
(use "u")
(use "v")

(proof-to-expr-with-formulas (current-proof))

u80: all x(A -> P x)
v81: A

(lambda (u80) (lambda (v81) (lambda (x) ((u80 x) v81))))
```

Add $A \vee B$, $A \wedge B$ and $\exists_x A$. Define

$$\neg A := A \to \bot, \quad \text{with } \bot \text{ an arbitrary propositional symbol.}$$

Axioms:

$$\vee_0^+ : A \to A \vee B$$
$$\vee_1^+ : B \to A \vee B$$
$$\vee^- : A \vee B \to (A \to C) \to (B \to C) \to C$$

$$\wedge^+ : A \to B \to A \wedge B$$
$$\wedge^- : A \wedge B \to (A \to B \to C) \to C$$

$$\exists^+ : A \to \exists_x A$$
$$\exists^- : \exists_x A \to \forall_x (A \to B) \to B \quad (x \notin \mathrm{FV}(B)).$$

Rules for $\vee$, $\wedge$, $\exists$

$$\dfrac{\begin{array}{c}\mid M\\ A\end{array}}{A \vee B}\ \vee_0^+ \qquad \dfrac{\begin{array}{c}\mid M\\ B\end{array}}{A \vee B}\ \vee_1^+$$

$$\dfrac{\begin{array}{ccc} & [u\colon A] & [v\colon B] \\ \mid M & \mid N & \mid K \\ A \vee B & C & C \end{array}}{C}\ \vee^- u, v$$

$$\dfrac{\begin{array}{cc}\mid M & \mid N\\ A & B\end{array}}{A \wedge B}\ \wedge^+$$

$$\dfrac{\begin{array}{ccc} & [u\colon A] & [v\colon B] \\ \mid M & & \mid N \\ A \wedge B & & C \end{array}}{C}\ \wedge^- u, v$$

$$\dfrac{\begin{array}{cc} & \mid M\\ r & A(r)\end{array}}{\exists_x A(x)}\ \exists^+$$

$$\dfrac{\begin{array}{ccc} & [u\colon A] & \\ \mid M & & \mid N \\ \exists_x A & & B \end{array}}{B}\ \exists^- x, u \text{ (var.cond.)}$$

$$\vdash \neg\neg\forall_x A \to \forall_x \neg\neg A$$

Proof:

$$\dfrac{\dfrac{u : \neg\neg\forall_x A \qquad \dfrac{\dfrac{\dfrac{w : \forall_x A \qquad x}{A}}{\dfrac{v : \neg A \qquad \bot}{\neg\forall_x A}\,{\to}^+ w}}{\bot}\,{\to}^+ v}{\neg\neg A}}{\dfrac{\forall_x \neg\neg A}{\neg\neg\forall_x A \to \forall_x \neg\neg A}\,{\to}^+ u}\,\forall^+ x$$

$$\vdash \exists_x(A \to B) \to A \to \exists_x B \quad \text{with } x \notin \mathrm{FV}(A).$$

Proof:

$$\dfrac{\dfrac{u\colon \exists_x(A \to B) \qquad \dfrac{\dfrac{x \qquad \dfrac{w\colon A \to B \qquad v\colon A}{B}\to^- }{\exists_x B}\exists^+}{\exists_x B}\exists^- x, w}{A \to \exists_x B}\to^+ v}{\exists_x(A \to B) \to A \to \exists_x B}\to^+ u$$

$\vee$-conversion:

$$\dfrac{\dfrac{\begin{array}{c}\mid M\\ A\end{array}}{A \vee B}\vee_0^+ \qquad \begin{array}{c}[u\colon A]\\ \mid N\\ C\end{array} \qquad \begin{array}{c}[v\colon B]\\ \mid K\\ C\end{array}}{C}\vee^- u, v \quad \mapsto \quad \begin{array}{c}\mid M\\ A\\ \mid N\\ C\end{array}$$

$\wedge$-conversion:

$$\dfrac{\dfrac{\begin{array}{c}\vert\, M\\ A\end{array}\qquad\begin{array}{c}\vert\, N\\ B\end{array}}{A \wedge B}\,\wedge^{+}\qquad\begin{array}{c}[u\colon A]\quad[v\colon B]\\ \vert\, K\\ C\end{array}}{C}\,\wedge^{-}\, u, v \quad\mapsto\quad \begin{array}{c}\vert\, M\\ A\end{array}\qquad\begin{array}{c}\vert\, N\\ B\end{array}\qquad\begin{array}{c}\vert\, K\\ C\end{array}$$

$\exists$-conversion:

$$\dfrac{\dfrac{r \qquad \begin{array}{c}\mid M\\ A(r)\end{array}}{\exists_x A(x)}\exists^+ \qquad \begin{array}{c}[u\colon A(x)]\\ \mid N\\ B\end{array}}{B}\exists^- x, u \quad \mapsto \quad \begin{array}{c}\mid M\\ A(r)\\ \mid N'\\ B\end{array}$$

Written with derivation terms:

$$\exists^-(\exists^+ rM)(\lambda_{x,u} N(x,u)) \mapsto_\beta N(r,M).$$

$\vee$-permutative conversion:

$$\dfrac{\dfrac{\dfrac{}{A \vee B}\,|\,M \qquad \dfrac{}{C}\,|\,N \qquad \dfrac{}{C}\,|\,K}{C}\,\vee\text{-rule} \qquad \dfrac{}{C'}\,|\,L}{D}\,\text{E-rule} \quad \mapsto$$

$$\dfrac{\dfrac{}{A \vee B}\,|\,M \qquad \dfrac{\dfrac{}{C}\,|\,N \qquad \dfrac{}{C'}\,|\,L}{D}\,\text{E-rule} \qquad \dfrac{\dfrac{}{C}\,|\,K \qquad \dfrac{}{C'}\,|\,L}{D}\,\text{E-rule}}{D}\,\vee\text{-rule}$$

$\wedge$-permutative conversion:

$$\dfrac{\dfrac{\begin{array}{c}|\,M\\ A\wedge B\end{array}\qquad\begin{array}{c}|\,N\\ C\end{array}}{C}\qquad\begin{array}{c}|\,K\\ C'\end{array}}{D}\,\text{E-rule}\quad\mapsto$$

$$\dfrac{\begin{array}{c}|\,M\\ A\wedge B\end{array}\qquad\dfrac{\begin{array}{c}|\,N\\ C\end{array}\qquad\begin{array}{c}|\,K\\ C'\end{array}}{D}\,\text{E-rule}}{D}$$

$\exists$-permutative conversion:

$$\dfrac{\dfrac{\exists_x A \quad \overset{|\,M}{B}}{B}\qquad \overset{|\,N}{C}}{D}\,\text{E-rule} \qquad \mapsto$$

$$\dfrac{\exists_x A \quad \overset{|\,M}{\dfrac{B \quad \overset{|\,N}{C}}{D}}\,\text{E-rule}}{D}$$

Distinguish two kinds of "exists" and two kinds of "or":

- the weak or "classical" ones, and
- the strong or "non-classical" ones, with constructive content.

Here both kinds occur together.

$$A \mathbin{\tilde{\vee}} B := \neg A \to \neg B \to \bot, \qquad \tilde{\exists}_x A := \neg \forall_x \neg A.$$

The strong ones imply the weak ones:

$$A \vee B \to A \mathbin{\tilde{\vee}} B, \qquad \exists_x A \to \tilde{\exists}_x A.$$

(Put $C := \bot$ in $\vee^-$ and $B := \bot$ in $\exists^-$.)

Since $\tilde{\exists}_x \tilde{\exists}_y A$ unfolds into a rather awkward formula we extend the $\tilde{\exists}$-terminology to lists of variables:

$$\tilde{\exists}_{x_1,\ldots,x_n} A := \forall_{x_1,\ldots,x_n}(A \to \bot) \to \bot.$$

Moreover let

$$\tilde{\exists}_{x_1,\ldots,x_n}(A_1 \mathbin{\tilde{\wedge}} \ldots \mathbin{\tilde{\wedge}} A_m) := \forall_{x_1,\ldots,x_n}(A_1 \to \cdots \to A_m \to \bot) \to \bot.$$

This allows to stay in the $\to, \forall$ part of the language. Notice that $\tilde{\wedge}$ only makes sense in this context, i.e., in connection with $\tilde{\exists}$.

In the definition of derivability in falsity $\bot$ plays no role. We can change this and require ex-falso-quodlibet axioms:

$$\mathrm{Efq} := \{\, \forall_{\vec{x}}(\bot \to R\vec{x}\,) \mid R \neq \bot \,\}.$$

A formula A is intuitionistically derivable, written $\vdash_i A$, if $\mathrm{Efq} \vdash A$. We write $\Gamma \vdash_i B$ for $\Gamma \cup \mathrm{Efq} \vdash B$.

We may even go further and require stability axioms:

$$\mathrm{Efq} := \{\, \forall_{\vec{x}}(\neg\neg R\vec{x} \to R\vec{x}\,) \mid R \neq \bot \,\}.$$

A formula A is classically derivable, written $\vdash_c A$, if $\mathrm{Stab} \vdash A$. We write $\Gamma \vdash_c B$ for $\Gamma \cup \mathrm{Stab} \vdash B$.

Using the introduction rules one easily proves

$$\vdash_i \bot \to A$$

for arbitrary A.

Theorem (Stability, or principle of indirect proof)

(a) $\vdash (\neg\neg A \to A) \to (\neg\neg B \to B) \to \neg\neg(A \wedge B) \to A \wedge B$.
(b) $\vdash (\neg\neg B \to B) \to \neg\neg(A \to B) \to A \to B$.
(c) $\vdash (\neg\neg A \to A) \to \neg\neg\forall_x A \to A$.
(d) $\vdash_c \neg\neg A \to A$ *for every formula* A *without* $\vee, \exists$.

(b) $\vdash (\neg\neg B \to B) \to \neg\neg(A \to B) \to A \to B$.

$$\dfrac{u : \neg\neg B \to B \qquad \dfrac{\dfrac{v : \neg\neg(A \to B) \qquad \dfrac{\dfrac{\dfrac{u_1 : \neg B \qquad \dfrac{u_2 : A \to B \quad w : A}{B}}{\bot}}{\neg(A \to B)} \to^+ u_2}{\bot}}{\neg\neg B} \to^+ u_1}{B}$$

(c) $\vdash (\neg\neg A \to A) \to \neg\neg\forall_x A \to A$.

$$\dfrac{u : \neg\neg A \to A \qquad \dfrac{\dfrac{v : \neg\neg\forall_x A \qquad \dfrac{\dfrac{u_1 : \neg A \qquad \dfrac{u_2 : \forall_x A \qquad x}{A}}{\bot}\,{\to}^+ u_2}{\neg\forall_x A}}{\bot}\,{\to}^+ u_1}{\neg\neg A}}{A}$$

(d) $\vdash_c \neg\neg A \to A$ for every formula A without $\vee, \exists$.

Proof.

Induction on A. The case $R\vec{t}$ with $R \neq \bot$ is given by Stab. In the case $\bot$ the desired derivation is

$$\dfrac{v\colon (\bot \to \bot) \to \bot \qquad \dfrac{u\colon \bot}{\bot \to \bot}\to^+ u}{\bot}$$

In the cases $A \wedge B$, $A \to B$ and $\forall_x A$ use (a), (b) and (c). $\square$

Lemma

The following are derivable.

$$
\begin{array}{lll}
 & (\tilde{\exists}_x A \to B) \to \forall_x (A \to B) & \text{if } x \notin \mathrm{FV}(B), \\
(\neg\neg B \to B) \to & \forall_x (A \to B) \to \tilde{\exists}_x A \to B & \text{if } x \notin \mathrm{FV}(B), \\
(\bot \to B[x{:=}c]) \to & (A \to \tilde{\exists}_x B) \to \tilde{\exists}_x (A \to B) & \text{if } x \notin \mathrm{FV}(A), \\
 & \tilde{\exists}_x (A \to B) \to A \to \tilde{\exists}_x B & \text{if } x \notin \mathrm{FV}(A).
\end{array}
$$

Last two simplify a weakly existentially quantified implication whose premise does not contain the quantified variable. In case the conclusion does not contain the quantified variable we have

$$
\begin{array}{lll}
(\neg\neg B \to B) \to & \tilde{\exists}_x (A \to B) \to \forall_x A \to B & \text{if } x \notin \mathrm{FV}(B), \\
\forall_x (\neg\neg A \to A) \to & (\forall_x A \to B) \to \tilde{\exists}_x (A \to B) & \text{if } x \notin \mathrm{FV}(B).
\end{array}
$$

$(\tilde{\exists}_x A \to B) \to \forall_x (A \to B)$ if $x \notin \mathrm{FV}(B)$.

Proof.

$$\dfrac{\tilde{\exists}_x A \to B \qquad \dfrac{\dfrac{u_1 : \forall_x \neg A \qquad x}{\neg A} \qquad A}{\dfrac{\bot}{\neg \forall_x \neg A}\, \to^+ u_1}}{B}$$

□

$(\neg\neg B \to B) \to \forall_x(A \to B) \to \tilde{\exists}_x A \to B$ if $x \notin \mathrm{FV}(B)$.

Proof.

$$\dfrac{\neg\neg B \to B \qquad \dfrac{\neg\forall_x \neg A \qquad \dfrac{\dfrac{\dfrac{\forall_x(A \to B) \qquad x}{A \to B} \qquad u_1 : A}{B}}{\dfrac{\bot}{\neg A} \to^+ u_1}}{\forall_x \neg A}}{\dfrac{\bot}{\neg\neg B} \to^+ u_2}}{B}$$

□

$(\bot \to B[x{:=}c]) \to (A \to \tilde{\exists}_x B) \to \tilde{\exists}_x (A \to B)$ if $x \notin \mathrm{FV}(A)$.

Proof.

Writing B_0 for $B[x{:=}c]$ we have

$$\dfrac{\dfrac{\forall_x \neg(A \to B)\ \ c}{\neg(A \to B_0)} \quad \dfrac{\dfrac{\dfrac{\dfrac{\forall_x \neg(A \to B)\ \ x}{\neg(A \to B)} \quad \dfrac{u_1 : B}{A \to B}}{\bot}}{\neg B} \to^+ u_1}{\forall_x \neg B} \qquad \dfrac{\dfrac{A \to \tilde{\exists}_x B \quad u_2 : A}{\tilde{\exists}_x B} \qquad \dfrac{\bot \to B_0}{B_0} \to^+ u_2}{A \to B_0}}{\bot}$$

□

$\tilde{\exists}_x(A \to B) \to A \to \tilde{\exists}_x B$ if $x \notin \mathrm{FV}(A)$.

Proof.

$$\dfrac{\tilde{\exists}_x(A \to B) \qquad \dfrac{\forall_x \neg(A \to B)}{\dfrac{\neg(A \to B)}{\dfrac{\dfrac{\dfrac{\forall_x \neg B \qquad x}{\neg B} \qquad \dfrac{u_1 : A \to B \qquad A}{B}}{\bot}}{\bot} \to^+ u_1}}}{\bot}$$

□

$(\neg\neg B \to B) \to \tilde{\exists}_x(A \to B) \to \forall_x A \to B$ if $x \notin \mathrm{FV}(B)$.

Proof.

$$\dfrac{\neg\neg B \to B \qquad \dfrac{\tilde{\exists}_x(A \to B) \qquad \dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\forall_x A \qquad x}{A} \qquad u_1 : A \to B}{B} \qquad u_2 : \neg B}{\bot}}{\neg(A \to B)} \to^+ u_1}{\forall_x \neg(A \to B)}}{\bot}}{\neg\neg B} \to^+ u_2}{B}$$

□

$\forall_x(\neg\neg A \to A) \to (\forall_x A \to B) \to \tilde{\exists}_x(A \to B)$ if $x \notin \mathrm{FV}(B)$.

We derive $\forall_x(\bot \to A) \to (\forall_x A \to B) \to \forall_x \neg(A \to B) \to \neg\neg A$.

$$\dfrac{\dfrac{\dfrac{\forall_x \neg(Ax \to B) \quad x}{\neg(Ax \to B)} \qquad \dfrac{\dfrac{\forall_x Ax \to B \qquad \dfrac{\dfrac{\forall_y(\bot \to Ay) \quad y}{\bot \to Ay} \qquad \dfrac{u_1 : \neg Ax \qquad u_2 : Ax}{\bot}}{Ay}}{\forall_y Ay}}{B}}{Ax \to B} \to^+ u_2}{\bot} \to^+ u_1}{\neg\neg Ax}$$

Using this derivation M we obtain

$$\dfrac{\dfrac{\dfrac{\forall_x \neg(Ax \to B) \quad x}{\neg(Ax \to B)} \qquad \dfrac{\dfrac{\forall_x Ax \to B \qquad \dfrac{\dfrac{\forall_x(\neg\neg Ax \to Ax) \quad x}{\neg\neg Ax \to Ax} \qquad \neg\neg Ax \mid M}{Ax}}{\forall_x Ax}}{B}}{Ax \to B}}{\bot}$$

Since clearly $\vdash (\neg\neg A \to A) \to \bot \to A$ the claim follows.

A consequence of

$$\forall_x(\neg\neg A \to A) \to (\forall_x A \to B) \to \tilde{\exists}_x(A \to B) \quad \text{with } x \notin \mathrm{FV}(B)$$

is the classical derivability of the drinker formula $\tilde{\exists}_x(Px \to \forall_x Px)$, to be read

> *in every non-empty bar there is a person such that, if this person drinks, then everybody drinks.*

To see this let $A := Px$ and $B := \forall_x Px$.

There is a similar lemma on weak disjunction:

Lemma

The following are derivable.

$$
\begin{aligned}
&(A \mathbin{\tilde{\vee}} B \to C) \to (A \to C) \wedge (B \to C),\\
(\neg\neg C \to C) \to\ &(A \to C) \to (B \to C) \to A \mathbin{\tilde{\vee}} B \to C,\\
(\bot \to B) \to\ &(A \to B \mathbin{\tilde{\vee}} C) \to (A \to B) \mathbin{\tilde{\vee}} (A \to C),\\
&(A \to B) \mathbin{\tilde{\vee}} (A \to C) \to A \to B \mathbin{\tilde{\vee}} C,\\
(\neg\neg C \to C) \to\ &(A \to C) \mathbin{\tilde{\vee}} (B \to C) \to A \to B \to C,\\
(\bot \to C) \to\ &(A \to B \to C) \to (A \to C) \mathbin{\tilde{\vee}} (B \to C).
\end{aligned}
$$

$(\bot \to C) \to (A \to B \to C) \to (A \to C) \tilde{\vee} (B \to C)$.

Proof.

$$\dfrac{\dfrac{\neg(B \to C) \qquad \dfrac{\dfrac{\bot \to C \qquad \dfrac{\neg(A \to C) \qquad \dfrac{\dfrac{A \to B \to C \qquad u_1 : A}{B \to C} \qquad u_2 : B}{C}}{\bot}}{C}}{A \to C} \to^+ u_1}{B \to C} \to^+ u_2}{\bot}$$

□

As a corollary we have

$$\vdash_c (A \mathbin{\tilde{\vee}} B \to C) \leftrightarrow (A \to C) \wedge (B \to C) \quad \text{for } C \text{ without } \vee, \exists,$$
$$\vdash_i (A \to B \mathbin{\tilde{\vee}} C) \leftrightarrow (A \to B) \mathbin{\tilde{\vee}} (A \to C),$$
$$\vdash_c (A \to C) \mathbin{\tilde{\vee}} (B \to C) \leftrightarrow (A \to B \to C) \quad \text{for } C \text{ without } \vee, \exists.$$

$\tilde{\vee}$ and $\tilde{\exists}$ satisfy the same axioms as $\vee$ and $\exists$, if one restricts the conclusion of the elimination axioms to formulas without $\vee, \exists$:

$$\vdash A \to A \mathbin{\tilde{\vee}} B, \quad \vdash B \to A \mathbin{\tilde{\vee}} B,$$
$$\vdash_c A \mathbin{\tilde{\vee}} B \to (A \to C) \to (B \to C) \to C \quad (C \text{ without } \vee, \exists),$$
$$\vdash A \to \tilde{\exists}_x A,$$
$$\vdash_c \tilde{\exists}_x A \to \forall_x (A \to B) \to B \quad (x \notin \mathrm{FV}(B),\ B \text{ without } \vee, \exists).$$

$\vdash_c A \tilde{\vee} B \to (A \to C) \to (B \to C) \to C$ for C without $\vee, \exists$.

Proof.

$$\dfrac{\dfrac{\neg\neg C \to C \qquad \dfrac{\dfrac{\neg A \to \neg B \to \bot \qquad \dfrac{\dfrac{u_1 : \neg C \qquad \dfrac{A \to C \quad u_2 : A}{C}}{\bot}}{\neg A} \to^+ u_2}{\neg B \to \bot} \qquad \dfrac{\dfrac{u_1 : \neg C \qquad \dfrac{B \to C \quad u_3 : B}{C}}{\bot}}{\neg B} \to^+ u_3}{\bot}}{\neg\neg C} \to^+ u_1}{C}$$

□

$\vdash_c \tilde{\exists}_x A \to \forall_x(A \to B) \to B$ for $x \notin \mathrm{FV}(B)$, B without $\vee, \exists$.

Proof.

$$\dfrac{\dfrac{\neg\neg B \to B \qquad \dfrac{\dfrac{\neg\forall_x \neg A \qquad \dfrac{\dfrac{\dfrac{\dfrac{\forall_x(A \to B) \quad x}{A \to B} \qquad u_2 : A}{B}}{\bot}}{\neg A}\to^+ u_2}{\forall_x \neg A}}{\bot}}{\neg\neg B}\to^+ u_1}{B}$$

□

Gödel-Gentzen translation A^g

The embedding of classical logic into minimal logic can be expressed in a different form: as a syntactic translation $A \mapsto A^g$:

$$\begin{aligned}
(R\vec{t}\,)^g &:= \neg\neg R\vec{t} \quad \text{for } R \text{ distinct from } \bot,\\
\bot^g &:= \bot,\\
(A \vee B)^g &:= A^g \,\tilde{\vee}\, B^g,\\
(\exists_x A)^g &:= \tilde{\exists}_x A^g,\\
(A \circ B)^g &:= A^g \circ B^g \quad \text{for } \circ = \to, \wedge,\\
(\forall_x A)^g &:= \forall_x A^g.
\end{aligned}$$

Lemma

$\vdash \neg\neg A^g \to A^g$.

Proof of $\vdash \neg\neg A^g \to A^g$.

Induction on A. Case $R\vec{t}$ with R distinct from $\bot$. To show $\neg\neg\neg\neg R\vec{t} \to \neg\neg R\vec{t}$, which is a special case of $\vdash \neg\neg\neg B \to \neg B$.

Case $\bot$. Use $\vdash \neg\neg\bot \to \bot$.

Case $A \vee B$. We must show $\vdash \neg\neg(A^g \mathbin{\tilde{\vee}} B^g) \to A^g \mathbin{\tilde{\vee}} B^g$, which is a special case of $\vdash \neg\neg(\neg C \to \neg D \to \bot) \to \neg C \to \neg D \to \bot$:

$$\dfrac{\neg\neg(\neg C \to \neg D \to \bot) \qquad \dfrac{\dfrac{u_1 : \neg C \to \neg D \to \bot \qquad \neg C}{\neg D \to \bot} \qquad \neg D}{\bot}}{\bot} \to^+ u_1$$

Case $\exists_x A$. To show $\vdash \neg\neg\tilde{\exists}_x A^g \to \tilde{\exists}_x A^g$, which is special case of $\vdash \neg\neg\neg B \to \neg B$, because $\tilde{\exists}_x A^g$ is the negation $\neg\forall_x \neg A^g$.

Case $A \wedge B$. To show $\vdash \neg\neg(A^g \wedge B^g) \to A^g \wedge B^g$. By IH $\vdash \neg\neg A^g \to A^g$ and $\vdash \neg\neg B^g \to B^g$. Use (a) of the stability thm.

The cases $A \to B$ and $\forall_x A$ are similar, using (b) and (c) of the stability theorem. □

Theorem

(a) $\Gamma \vdash_c A$ *implies* $\Gamma^g \vdash A^g$.

(b) $\Gamma^g \vdash A^g$ *implies* $\Gamma \vdash_c A$ *for* Γ, A *without* $\vee, \exists$.

Proof. (a) Use induction on $\Gamma \vdash_c A$. For a stability axiom $\forall_{\vec{x}}(\neg\neg R\vec{x} \to R\vec{x})$ we must derive $\forall_{\vec{x}}(\neg\neg\neg\neg R\vec{x} \to \neg\neg R\vec{x})$; easy.
For $\to^+$, $\to^-$, $\forall^+$, $\forall^-$, $\wedge^+$ and $\wedge^-$ the claim follows from the IH, using the same rule ($A \mapsto A^g$ acts as a homomorphism).
For $\vee_i^+$, $\vee^-$, $\exists^+$ and $\exists^-$ the claim follows from the IH and the remark above. For example, in case $\exists^-$ the IH gives

$$\begin{array}{c} \\ \mid M \\ \tilde{\exists}_x A^g \end{array} \quad \text{and} \quad \begin{array}{c} u\colon A^g \\ \mid N \\ B^g \end{array}$$

with $x \notin \mathrm{FV}(B^g)$. Now use
$\vdash (\neg\neg B^g \to B^g) \to \tilde{\exists}_x A^g \to \forall_x(A^g \to B^g) \to B^g$. Its premise $\neg\neg B^g \to B^g$ is derivable by the lemma above.

Proof of (b): $\Gamma^g \vdash A^g$ implies $\Gamma \vdash_c A$ for Γ, A without $\vee, \exists$.

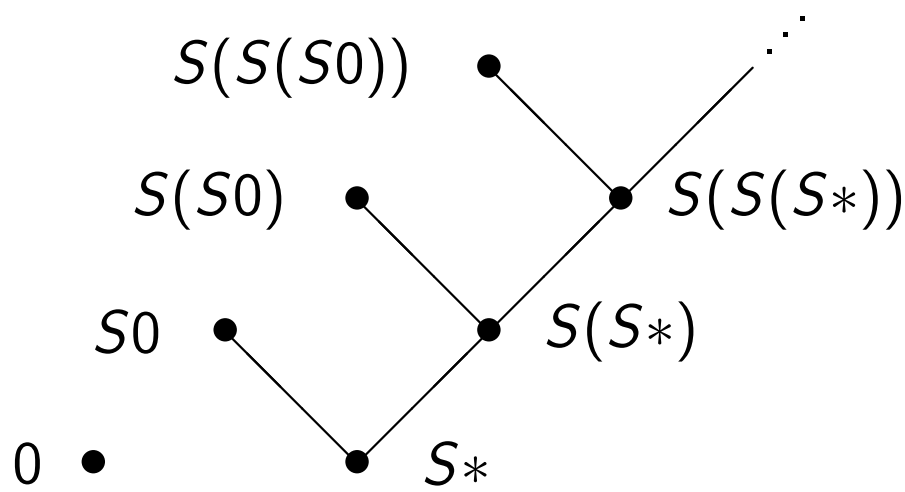
First note that $\vdash_c (B \leftrightarrow B^g)$ if B is without $\vee, \exists$. Now assume that Γ, A are without $\vee, \exists$. From $\Gamma^g \vdash A^g$ we obtain $\Gamma \vdash_c A$ as follows. We argue informally. Assume Γ. Then Γ^g by the note, hence A^g because of $\Gamma^g \vdash A^g$, hence A again by the note.

1. Logic
2. The model of partial continuous functionals
3. Formulas as problems
4. Computational content of proofs
5. Decorating proofs

Basic intuition: describe $x \mapsto f(x)$ in the infinite (or "ideal") world by means of finite approximations.

Given an atomic piece b (a "token") of information on the value $f(x)$, we should have a finite set U (a "formal neighborhood") of tokens approximating the argument x such that $b \in f_0(U)$, where f_0 is a finite approximation of f.

Want the constructors to be continuous and with disjoint ranges. This requires



Structural recursion operators:

$$\mathcal{R}^{\tau}_{\mathbf{N}} \colon \mathbf{N} \to \tau \to (\mathbf{N} \to \tau \to \tau) \to \tau$$

given by the defining equations

$$\begin{aligned} \mathcal{R}^{\tau}_{\mathbf{N}}(0, a, f) &= a, \\ \mathcal{R}^{\tau}_{\mathbf{N}}(S(n), a, f) &= f(n, \mathcal{R}^{\tau}_{\mathbf{N}}(n, a, f)). \end{aligned}$$

Similarly for lists of objects of type ρ we have

$$\mathcal{R}^{\tau}_{\mathbf{L}(\rho)} \colon \mathbf{L}(\rho) \to \tau \to (\rho \to \mathbf{L}(\rho) \to \tau \to \tau) \to \tau$$

with defining equations

$$\begin{aligned} \mathcal{R}^{\tau}_{\mathbf{L}(\rho)}([], a, f) &= a, \\ \mathcal{R}^{\tau}_{\mathbf{L}(\rho)}(x :: \ell, a, f) &= f(x, \ell, \mathcal{R}^{\tau}_{\mathbf{L}(\rho)}(\ell, a, f)). \end{aligned}$$

The defining equation

$$Y(f) = f(Y(f))$$

is admitted as well, and it defines a partial functional.

f of type $\rho \to \sigma$ is called total if it maps total objects of type ρ to total objects of type σ.

Natural numbers

```
(load "~/git/minlog/init.scm")

(set! COMMENT-FLAG #f)
(libload "nat.scm")
(set! COMMENT-FLAG #t)

(display-alg "nat")
(display-pconst "NatPlus")
```

Normalizing, apply term rewriting rules.

```
(pp (nt (pt "3+4")))
(pp (nt (pt "Succ n+Succ m+0")))
```

Defining program constants.

```
(add-program-constant "Double" (py "nat=>nat"))
(add-computation-rules
 "Double 0" "0"
 "Double(Succ n)" "Succ(Succ(Double n))")

(pp (nt (pt "Double 3")))
(pp (nt (pt "Double (n+2)")))
```

Proof by induction, apply term-rewriting-rules.

```
(set-goal "all n Double n=n+n")
(ind)
;; base
(ng)
(use "Truth")
;; step
(assume "n" "IH")
(ng)
(use "IH")
```

Boolean-valued functions

```
(add-program-constant "Odd" (py "nat=>boole"))
(add-program-constant "Even" (py "nat=>boole"))

(add-computation-rules
 "Odd 0" "False"
 "Even 0" "True"
 "Odd(Succ n)" "Even n"
 "Even(Succ n)" "Odd n")

(set-goal "all n Even(Double n)")
(ind)
(prop)
(search)
```

```
(display-pconst "NatLt")

NatLt
  comprules
        nat<0                    False
        0<Succ nat               True
        Succ nat1<Succ nat2      nat1<nat2
  rewrules
        nat<Succ nat             True
        nat<nat                  False
        Succ nat<nat             False
        nat1+nat2<nat1           False
```

Quotient and remainder

$\forall_{m,n}\exists_{q,r}(n = (m+1)q + r \wedge r < m+1)$.

Proof.

Induction on n. Base. Pick $q = r = 0$. Step. By IH have q, r for n. Argue by cases.

- If $r < m$ let $q' = q$ and $r' = r + 1$.
- If $r = m$ let $q' = q + 1$ and $r' = 0$. □

Will be an easy example for program extraction from proofs.

Lists

```
(load "~/git/minlog/init.scm")

(set! COMMENT-FLAG #f)
(libload "nat.scm")
(libload "list.scm")
(set! COMMENT-FLAG #t)

(add-var-name "x" "a" "b" "c" "d" (py "alpha"))
(add-var-name "xs" "ys" "v" "w" "u" (py "list alpha"))

(add-program-constant
 "ListRv" (py "list alpha=>list alpha") t-deg-one)
(add-prefix-display-string "ListRv" "Rv")
(add-computation-rules
 "Rv(Nil alpha)" "(Nil alpha)"
 "Rv(x::xs)" "Rv xs++x:")
```

```
(display-pconst "ListAppd")
```

We prove that Rv commutes with ++

```
(set-goal "all v,w Rv(v++w)eqd Rv w++Rv v")
(ind)
;; Base
(ng)
(assume "w")
(use "InitEqD")
;; Step
(assume "a" "v" "IHw" "w")
(ng)
(simp "IHw")
(simp "ListAppdAssoc")
(use "InitEqD")
```

List reversal

We give an informal existence proof for list reversal.

$$R([],[]),$$
$$\forall_{v,w,x}(Rvw \to R(vx, xw)).$$

View R as an inductive predicate without computational content.

$$\texttt{ListInitLastNat}\colon \forall_{u,y}\exists_{v,x}(yu = vx).$$
$$\texttt{ExR}\colon \forall_{n,v}(n = |v| \to \exists_w Rvw).$$

Proof of ExR.
By induction on the length of v. In the step case, our list is non-empty, and hence can be written in the form vx. Since v has smaller length, the IH yields its reversal w. Take xw. □

Will be another example for program extraction from proofs.

Binary trees

Nodes in a binary tree can be viewed as lists of booleans, where tt means left and ff means right. Brouwer-Kleene ordering:

$$\begin{aligned}
[] &\ll b && := \mathrm{ff}\\
p :: a &\ll [] && := \mathrm{tt}\\
\mathrm{tt} :: a &\ll \mathrm{tt} :: b && := a \ll b\\
\mathrm{tt} :: a &\ll \mathrm{ff} :: b && := \mathrm{tt}\\
\mathrm{ff} :: a &\ll \mathrm{tt} :: b && := \mathrm{ff}\\
\mathrm{ff} :: a &\ll \mathrm{ff} :: b && := a \ll b
\end{aligned}$$

Let $\mathrm{Incr}(a_0 :: a_1 :: \cdots :: a_{n-1})$ mean $a_0 \ll a_1 \ll \ldots \ll a_{n-1}$.

$$\mathrm{ExBK}\colon \forall_r \exists_\ell (|\ell| = \|r\| \wedge \forall_{n<|\ell|}((\ell)_n \in r) \wedge \mathrm{Incr}(\ell)).$$

Will be another example for program extraction from proofs.

1. Logic
2. The model of partial continuous functionals
3. Formulas as problems
4. Computational content of proofs
5. Decorating proofs

Formulas as computational problems

- Kolmogorov (1932) proposed to view a formula A as a computational problem, of type $\tau(A)$, the type of a potential solution or "realizer" of A.
- Example: $\forall_n \exists_{m>n} \mathrm{Prime}(m)$ has type $\mathbf{N} \to \mathbf{N}$.
- $A \mapsto \tau(A)$, a type or the "nulltype" symbol $\circ$.
- In case $\tau(A) = \circ$ proofs of A have no computational content; such formulas A are called non-computational (n.c.) or Harrop formulas; the others computationally relevant (c.r.).

Examples.

$$\begin{aligned}
\tau(\forall_{m,n}\exists_{q,r}(n = (m+1)q + r \wedge r < m+1)) &= \mathbf{N} \to \mathbf{N} \to \mathbf{N} \times \mathbf{N} \\
\tau(\forall_{n,v}(n = |v| \to \exists_w Rvw)) &= \mathbf{N} \to \mathbf{L}(\mathbf{N}) \to \mathbf{L}(\mathbf{N}) \\
\tau(\forall_r \exists_\ell(|\ell| = \|r\| \wedge \forall_{n<|\ell|}((\ell)_n \in r) \wedge \mathrm{Incr}(\ell))) &= \mathbf{D} \to \mathbf{L}(\mathbf{L}(\mathbf{B}))
\end{aligned}$$

Decoration

Which of the variables $\vec{x}$ and assumptions $\vec{A}$ are actually used in the "solution" provided by a proof of

$$\forall_{\vec{x}}(\vec{A} \to I\vec{r})?$$

To express this we split each of $\to, \forall$ into two variants:

- a "computational" one $\to^{\mathrm{c}}, \forall^{\mathrm{c}}$ and
- a "non-computational" one $\to^{\mathrm{nc}}, \forall^{\mathrm{nc}}$ (with restricted rules)

and consider

$$\forall^{\mathrm{nc}}_{\vec{x}}\forall^{\mathrm{c}}_{\vec{y}}(\vec{A} \to^{\mathrm{nc}} \vec{B} \to^{\mathrm{c}} X\vec{r}).$$

This will lead to a different (simplified) algebra ι_I associated with the inductive predicate I.

Decorated predicates and formulas

Distinguish two sorts of predicate variables, computationally relevant ones $X, Y, Z \ldots$ and non-computational ones $\hat{X}, \hat{Y}, \hat{Z} \ldots$.

$$P, Q ::= X \mid \hat{X} \mid \{\, \vec{x} \mid A \,\} \mid \mu_X^{\mathrm{c/nc}}(\forall_{\vec{x}_i}^{\mathrm{c/nc}}((A_{i\nu})_{\nu<n_i} \to^{\mathrm{c/nc}} X\vec{r}_i))_{i<k}$$

$$A, B ::= P\vec{r} \mid A \to^{\mathrm{c}} B \mid A \to^{\mathrm{nc}} B \mid \forall_x^{\mathrm{c}} A \mid \forall_x^{\mathrm{nc}} A$$

with $k \geq 1$ and $\vec{x}_i$ all free variables in $(A_{i\nu})_{\nu<n_i} \to^{\mathrm{c/nc}} X\vec{r}_i$. In the $\mu^{\mathrm{c/nc}}$ case we require that X occurs only "strictly positive" in the formulas $A_{i\nu}$, i.e., never on the left hand side of an implication.

- We usually write $\to$, $\forall$, μ for $\to^{\mathrm{c}}$, $\forall^{\mathrm{c}}$, μ^{c}.
- In the clauses of an n.c. inductive predicate $\mu_X^{\mathrm{nc}}\vec{K}$ decorations play no role; hence we write $\to, \forall$ for $\to^{\mathrm{c/nc}}, \forall^{\mathrm{c/nc}}$.

The type $\tau(C)$ of a formula or predicate C

$\tau(C)$ type or the "nulltype symbol" $\circ$. Extend use of $\rho \to \sigma$ to $\circ$:

$$(\rho \to \circ) := \circ, \quad (\circ \to \sigma) := \sigma, \quad (\circ \to \circ) := \circ.$$

Assume a global injective assignment of a type variable ξ to every c.r. predicate variable X. Let $\tau(C) := \circ$ if C is non-computational. In case C is c.r. let

$$\tau(P\vec{r}) := \tau(P),$$
$$\tau(A \to B) := (\tau(A) \to \tau(B)), \quad \tau(A \to^{\mathrm{nc}} B) := \tau(B),$$
$$\tau(\forall_{x^\rho} A) := (\rho \to \tau(A)), \quad \tau(\forall^{\mathrm{nc}}_{x^\rho} A) := \tau(A),$$
$$\tau(X) := \xi,$$
$$\tau(\{\, \vec{x} \mid A \,\}) := \tau(A),$$
$$\tau(\underbrace{\mu_X(\forall^{\mathrm{nc}}_{\vec{x}_i}\forall_{\vec{y}_i}(\vec{A}_i \to^{\mathrm{nc}} \vec{B}_i \to X\vec{r}_i))_{i<k}}_{I}) := \underbrace{\mu_\xi(\tau(\vec{y}_i) \to \tau(\vec{B}_i) \to \xi)_{i<k}}_{\iota_I}.$$

ι_I is the algebra associated with I.

We define when a predicate or formula is non-computational (n.c.) (or Harrop):

- $\hat{X}$ is n.c. but X is not,
- $\{\,\vec{x} \mid A\,\}$ is n.c. if A is,
- $\mu_X^{\mathrm{nc}}\vec{K}$ is n.c. but $\mu_X\vec{K}$ is not,
- $P\vec{r}$ is n.c. if P is,
- $A \to^{\mathrm{c/nc}} B$ is n.c. if B is, and
- $\forall_x^{\mathrm{c/nc}} A$ is n.c. if A is.

The other predicates and formulas are computationally relevant (c.r.).

To avoid unnecessarily complex types we extend the use of $\rho \times \sigma$ to the nulltype sumbol $\circ$ by

$$(\rho \times \circ) := \rho, \quad (\circ \times \sigma) := \sigma, \quad (\circ \times \circ) := \circ.$$

Moreover we identify the unit type **U** with $\circ$.

For the even numbers we now have two variants:

$$\mathrm{EvenI} := \mu_X(X0, \forall_n^{\mathrm{nc}}(Xn \to X(S(Sn)))),$$
$$\mathrm{EvenI}^{\mathrm{nc}} := \mu_X^{\mathrm{nc}}(X0, \forall_n(Xn \to X(S(Sn)))).$$

In Minlog this is written as

```
(add-ids
  (list (list "EvenI" (make-arity (py "nat")) "algEvenI"))
  '("EvenI 0" "InitEvenI")
  '("allnc n(EvenI n -> EvenI(n+2))" "GenEvenI"))

(add-ids
  (list (list "EvenNc" (make-arity (py "nat"))))
  '("EvenNc 0" "InitEvenNc")
  '("all n(EvenNc n -> EvenNc(n+2))" "GenEvenNc"))
```

Generally for every c.r. inductive predicate I (i.e., defined as $\mu_X \vec{K}$) we have an n.c. variant I^{nc} defined as $\mu_X^{\mathrm{nc}} \vec{K}$.

Since decorations can be inserted arbitrarily and parameter predicates can be either n.c. or c.r. we obtain many variants of inductive predicates. For the existential quantifier we have

$$\begin{aligned}
\mathrm{ExD}\,Y &:= \mu X(\forall_x(Yx \to X)),\\
\mathrm{ExL}\,Y &:= \mu X(\forall_x(Yx \to^{\mathrm{nc}} X)).\\
\mathrm{ExR}\,Y &:= \mu X(\forall^{\mathrm{nc}}_x(Yx \to X)),\\
\mathrm{ExU}\,Y &:= \mu^{\mathrm{nc}}_X(\forall^{\mathrm{nc}}_x(Yx \to^{\mathrm{nc}} X)).
\end{aligned}$$

Here D is for "double", L for "left", R for "right" and U for "uniform". We will use the abbreviations

$$\begin{aligned}
\exists^{\mathrm{d}}_x A &:= \mathrm{ExD}_{\{x|A\}},\\
\exists^{\mathrm{l}}_x A &:= \mathrm{ExL}_{\{x|A\}},\\
\exists^{\mathrm{r}}_x A &:= \mathrm{ExR}_{\{x|A\}},\\
\exists^{\mathrm{u}}_x A &:= \mathrm{ExU}_{\{x|A\}}.
\end{aligned}$$

For intersection we only consider the nullary case (i.e., conjunction). Then

$$\begin{aligned}
\mathrm{CapD}_{Y,Z} &:= \mu_X(Y \to Z \to X),\\
\mathrm{CapL}_{Y,Z} &:= \mu_X(Y \to Z \to^{\mathrm{nc}} X),\\
\mathrm{CapR}_{Y,Z} &:= \mu_X(Y \to^{\mathrm{nc}} Z \to X),\\
\mathrm{CapU}_{Y,Z} &:= \mu^{\mathrm{nc}}_X(Y \to^{\mathrm{nc}} Z \to^{\mathrm{nc}} X).
\end{aligned}$$

We use the abbreviations

$$\begin{aligned}
A \wedge^{\mathrm{d}} B &:= \mathrm{CapD}_{\{|A\},\{|B\}},\\
A \wedge^{\mathrm{l}} B &:= \mathrm{CapL}_{\{|A\},\{|B\}},\\
A \wedge^{\mathrm{r}} B &:= \mathrm{CapR}_{\{|A\},\{|B\}},\\
A \wedge^{\mathrm{u}} B &:= \mathrm{CapU}_{\{|A\},\{|B\}}.
\end{aligned}$$

For union: nullary case only (i.e., disjunction). Then

$$\begin{aligned}
\mathrm{CupD}_{Y,Z} &:= \mu_X(Y \to X,\ Z \to X),\\
\mathrm{CupL}_{Y,Z} &:= \mu_X(Y \to X,\ Z \to^{\mathrm{nc}} X),\\
\mathrm{CupR}_{Y,Z} &:= \mu_X(Y \to^{\mathrm{nc}} X,\ Z \to X),\\
\mathrm{CupU}_{Y,Z} &:= \mu_X(Y \to^{\mathrm{nc}} X,\ Z \to^{\mathrm{nc}} X),\\
\mathrm{CupNC}_{Y,Z} &:= \mu^{\mathrm{nc}}_X(Y \to X,\ Z \to X).
\end{aligned}$$

The final nc-variant is used to suppress even the information which clause has been used. We use the abbreviations

$$\begin{aligned}
A \vee^{\mathrm{d}} B &:= \mathrm{CupD}_{\{|A\},\{|B\}},\\
A \vee^{\mathrm{l}} B &:= \mathrm{CupL}_{\{|A\},\{|B\}},\\
A \vee^{\mathrm{r}} B &:= \mathrm{CupR}_{\{|A\},\{|B\}},\\
A \vee^{\mathrm{u}} B &:= \mathrm{CupU}_{\{|A\},\{|B\}},\\
A \vee^{\mathrm{nc}} B &:= \mathrm{CupNC}_{\{|A\},\{|B\}}.
\end{aligned}$$

For Leibniz equality we take the definition

$$\mathrm{EqD} := \mu^{\mathrm{nc}}_X(\forall_x Xxx).$$

Logical rules for the decorated connectives

We need to adapt our logical rules to $\to, \to^{\mathrm{nc}}$ and $\forall, \forall^{\mathrm{nc}}$.

- The introduction and elimination rules for $\to$, $\forall$ remain, and
- the elimination rules for $\to^{\mathrm{nc}}$, $\forall^{\mathrm{nc}}$ remain.

The introduction rules for $\to^{\mathrm{nc}}$, $\forall^{\mathrm{nc}}$ are restricted: the abstracted (assumption or object) variable must be "non-computational":

Simultaneously with a derivation M we define the sets $\mathrm{CV}(M)$ and $\mathrm{CA}(M)$ of computational object and assumption variables of M, as follows.

Let M^A be a derivation. If A is non-computational (n.c.) then $\mathrm{CV}(M^A) := \mathrm{CA}(M^A) := \emptyset$. Otherwise:

$$
\begin{aligned}
&\mathrm{CV}(c^A) := \emptyset \quad (c^A \text{ an axiom}),\\
&\mathrm{CV}(u^A) := \emptyset,\\
&\mathrm{CV}((\lambda_{u^A} M^B)^{A\to B}) := \mathrm{CV}((\lambda_{u^A} M^B)^{A\to^{\mathrm{nc}} B}) := \mathrm{CV}(M),\\
&\mathrm{CV}((M^{A\to B} N^A)^B) := \mathrm{CV}(M) \cup \mathrm{CV}(N),\\
&\mathrm{CV}((M^{A\to^{\mathrm{nc}} B} N^A)^B) := \mathrm{CV}(M),\\
&\mathrm{CV}((\lambda_x M^A)^{\forall_x A}) := \mathrm{CV}((\lambda_x M^A)^{\forall_x^{\mathrm{nc}} A}) := \mathrm{CV}(M) \setminus \{x\},\\
&\mathrm{CV}((M^{\forall_x A(x)} r)^{A(r)}) := \mathrm{CV}(M) \cup \mathrm{FV}(r),\\
&\mathrm{CV}((M^{\forall_x^{\mathrm{nc}} A(x)} r)^{A(r)}) := \mathrm{CV}(M),
\end{aligned}
$$

and similarly

$$\mathrm{CA}(c^A) := \emptyset \quad (c^A \text{ an axiom}),$$
$$\mathrm{CA}(u^A) := \{u\},$$
$$\mathrm{CA}((\lambda_{u^A} M^B)^{A\to B}) := \mathrm{CA}((\lambda_{u^A} M^B)^{A\to^{\mathrm{nc}} B}) := \mathrm{CA}(M) \setminus \{u\},$$
$$\mathrm{CA}((M^{A\to B} N^A)^B) := \mathrm{CA}(M) \cup \mathrm{CA}(N),$$
$$\mathrm{CA}((M^{A\to^{\mathrm{nc}} B} N^A)^B) := \mathrm{CA}(M),$$
$$\mathrm{CA}((\lambda_x M^A)^{\forall_x A}) := \mathrm{CA}((\lambda_x M^A)^{\forall^{\mathrm{nc}}_x A}) := \mathrm{CA}(M),$$
$$\mathrm{CA}((M^{\forall_x A(x)} r)^{A(r)}) := \mathrm{CA}((M^{\forall^{\mathrm{nc}}_x A(x)} r)^{A(r)}) := \mathrm{CA}(M).$$

The introduction rules for $\to^{\mathrm{nc}}$ and $\forall^{\mathrm{nc}}$ then are

- If M^B is a derivation and $u^A \notin \mathrm{CA}(M)$ then $(\lambda_{u^A} M^B)^{A\to^{\mathrm{nc}} B}$ is a derivation.
- If M^A is a derivation, x is not free in any formula of a free assumption variable of M and $x \notin \mathrm{CV}(M)$, then $(\lambda_x M^A)^{\forall^{\mathrm{nc}}_x A}$ is a derivation.

Decorated axioms

Consider a c.r. inductive predicate

$$I := \mu X(\forall_{\vec{x}_i}^{\mathrm{c/nc}}((A_{i\nu}(X))_{\nu<n_i} \to^{\mathrm{c/nc}} X\vec{r}_i))_{i<k}.$$

Then for every $i < k$ we have a clause (or introduction axiom

$$I_i^+ : \forall_{\vec{x}_i}^{\mathrm{c/nc}}((A_{i\nu}(I))_{\nu<n_i} \to^{\mathrm{c/nc}} I\vec{r}_i).$$

Moreover, we have an elimination axiom

$$I^- : \forall_{\vec{x}}^{\mathrm{nc}}(I\vec{x} \to (\forall_{\vec{x}_i}^{\mathrm{c/nc}}((A_{i\nu}(I \cap^{\mathrm{d}} X))_{\nu<n_i} \to^{\mathrm{c/nc}} X\vec{r}_i))_{i<k} \to X\vec{x}).$$

For example

$$
\begin{aligned}
(\mathrm{ExD}_{\{x|A\}})_0^+ &: \forall_x(A \to \exists_x^{\mathrm{d}} A),\\
(\mathrm{ExL}_{\{x|A\}})_0^+ &: \forall_x(A \to^{\mathrm{nc}} \exists_x^{\mathrm{l}} A),\\
(\mathrm{ExR}_{\{x|A\}})_0^+ &: \forall_x^{\mathrm{nc}}(A \to \exists_x^{\mathrm{r}} A),\\
(\mathrm{ExU}_{\{x|A\}})_0^+ &: \forall_x^{\mathrm{nc}}(A \to^{\mathrm{nc}} \exists_x^{\mathrm{u}} A).
\end{aligned}
$$

When $\{\, x \mid A \,\}$ is clear from the context we abbreviate

$$
\begin{aligned}
(\exists^{\mathrm{d}})^+ &:= (\mathrm{ExD}_{\{x|A\}})_0^+,\\
(\exists^{\mathrm{l}})^+ &:= (\mathrm{ExL}_{\{x|A\}})_0^+,\\
(\exists^{\mathrm{r}})^+ &:= (\mathrm{ExR}_{\{x|A\}})_0^+,\\
(\exists^{\mathrm{u}})^+ &:= (\mathrm{ExU}_{\{x|A\}})_0^+.
\end{aligned}
$$

For an n.c. inductive predicate $\hat{I}$ the introduction axioms $(\hat{I})^+_i$ are formed similarly. However, the elimination axiom $(\hat{I})^-$ needs to be restricted to non-computational competitor predicates $\hat{X}$, except when $\hat{I}$ is given by a one-clause-nc definition (i.e., with only one clause involving $\to^{\mathrm{nc}}, \forall^{\mathrm{nc}}$ only). Examples:

- Leibniz equality `EqD`, and
- uniform variants `ExU` and `AndU` of the existential quantifier and conjunction.

Recall that totality for the natural numbers was defined by the clauses

$$\texttt{TotalNatZero}\colon \texttt{TotalNat}\ 0$$

$$\texttt{TotalNatSucc}\colon \forall^{\mathrm{nc}}_{\hat{n}}(\texttt{TotalNat}\ \hat{n} \to \texttt{TotalNat}(\texttt{Succ}\ \hat{n}))$$

Using $\forall_{n\in T} Pn$ to abbreviate $\forall^{\mathrm{nc}}_{\hat{n}}(T_{\mathbf{N}}\hat{n} \to P\hat{n})$, the elimination axiom for `TotalNat` can be written as

$$\mathrm{Ind}_{n,A(n)}\colon \forall_{n\in T}(A(0) \to \forall_{n\in T}(A(n) \to A(Sn)) \to A(n^{\mathbf{N}})).$$

This is the usual induction axiom for natural numbers. We further abbreviate $\forall_{n\in T} Pn$ by $\forall_n Pn$, where using n rather than $\hat{n}$ indicates the n is meant to be restricted to the totality predicate T.

1. Logic
2. The model of partial continuous functionals
3. Formulas as problems
4. Computational content of proofs
5. Decorating proofs

Brouwer-Heyting-Kolmogorov

- p proves $A \to B$ if and only if p is a construction transforming any proof q of A into a proof $p(q)$ of B.
- p proves $\forall_{x^\rho} A(x)$ if and only if p is a construction such that for all a^ρ, $p(a)$ proves $A(a)$.

Leaves open:

- What is a "construction"?
- What is a proof of a prime formula?

Proposal:

- Construction: computable functional.
- Proof of a prime formula $I\vec{r}$: generation tree.

Example: generation tree for $\mathrm{Even}(6)$ should consist of a single branch with nodes $\mathrm{Even}(0)$, $\mathrm{Even}(2)$, $\mathrm{Even}(4)$ and $\mathrm{Even}(6)$.

Every constructive proof of an existential theorem contains – by the very meaning of "constructive proof" – a construction of a solution in terms of the parameters of the problem. To get hold of such a solution we have two methods.

Write-and-verify. Guided by our understanding of how the constructive proof works we directly write down a program to compute the solution, and then formally prove ("verify") that this indeed is the case.

Prove-and-extract. Formalize the constructive proof, and then extract the computational content of this proof in the form of a realizing term t. The soundness theorem guarantees (and even provides a formal proof) that t is a solution to the problem.

Realizability

For every predicate or formula C we define an n.c. predicate $C^{\mathbf{r}}$.
For n.c. C let

$$C^{\mathbf{r}} := C.$$

In case C is c.r. the arity of $C^{\mathbf{r}}$ is $(\tau(C), \vec{\sigma})$ with $\vec{\sigma}$ the arity of C.
For c.r. formulas define

$$(P\vec{r})^{\mathbf{r}} := \{ u \mid P^{\mathbf{r}} u\vec{r} \}$$

$$(A \to B)^{\mathbf{r}} := \begin{cases} \{ u \mid \forall_v (A^{\mathbf{r}} v \to B^{\mathbf{r}}(uv)) \} & \text{if } A \text{ is c.r.} \\ \{ u \mid A \to B^{\mathbf{r}} u \} & \text{if } A \text{ is n.c.} \end{cases}$$

$$(A \to^{\mathrm{nc}} B)^{\mathbf{r}} := \{ u \mid A \to B^{\mathbf{r}} u \}$$

$$(\forall_x A)^{\mathbf{r}} := \{ u \mid \forall_x A^{\mathbf{r}}(ux) \}$$

$$(\forall_x^{\mathrm{nc}} A)^{\mathbf{r}} := \{ u \mid \forall_x A^{\mathbf{r}} u \}.$$

For c.r. predicates: given n.c. $X^{\mathbf{r}}$ for all predicate variables X.

$$\{ \vec{x} \mid A \}^{\mathbf{r}} := \{ u, \vec{x} \mid A^{\mathbf{r}} u \}.$$

Consider a c.r. inductive predicate

$$I := \mu X(\forall^{\mathrm{c/nc}}_{\vec{x}_i}((A_{i\nu})_{\nu<n_i} \to^{\mathrm{c/nc}} X\vec{r}_i))_{i<k}.$$

$\vec{Y}$: all predicate variables strictly positive in some $A_{i\nu}$ except X.
Define the witnessing predicate with free predicate variables $\vec{Y}^{\mathbf{r}}$ by

$$I^{\mathbf{r}} := \mu^{\mathrm{nc}}_{X^{\mathbf{r}}}(\forall_{\vec{x}_i,\vec{u}_i}((A^{\mathbf{r}}_{i\nu}u_{i\nu})_{\nu<n_i} \to X^{\mathbf{r}}(\mathrm{C}_i\vec{x}_i\vec{u}_i)\vec{r}_i))_{i<k}$$

with the understanding that

(i) $u_{i\nu}$ occurs only when $A_{i\nu}$ is c.r., and it occurs as an argument in $\mathrm{C}_i\vec{x}_i\vec{u}_i$ only if $A_{i\nu}$ is c.r. and followed by $\to$, and

(ii) only those x_{ij} with $\forall^{\mathrm{c}}_{x_{ij}}$ occur as arguments in $\mathrm{C}_i\vec{x}_i\vec{u}_i$.

We write $u\ \mathbf{r}\ A$ for $A^{\mathbf{r}}u$ (u realizes A).

For the even numbers we obtain

$$\mathrm{Even} := \mu_X(X0, \forall_n^{\mathrm{nc}}(Xn \to X(S(Sn))))$$
$$\mathrm{Even}^{\mathbf{r}} := \mu_{X^{\mathbf{r}}}^{\mathrm{nc}}(X^{\mathbf{r}}00, \forall_{n,m}(X^{\mathbf{r}}mn \to X^{\mathbf{r}}(Sm)(S(Sn)))).$$

Axiom (Invariance under realizability)

$$\mathrm{Inv}_A\colon A \leftrightarrow \exists_u^{\mathrm{l}}(u\ \mathbf{r}\ A) \qquad \text{for c.r. formulas } A.$$

Lemma

For c.r. formulas A we have

$$(\lambda_u u)\ \mathbf{r}\ (A \to \exists_u^{\mathrm{l}}(u\ \mathbf{r}\ A)),$$
$$(\lambda_u u)\ \mathbf{r}\ (\exists_u^{\mathrm{l}}(u\ \mathbf{r}\ A) \to A).$$

From the invariance axioms we can derive

Theorem (Choice)

$$\forall_x \exists^{\mathrm{l}}_y A(y) \to \exists^{\mathrm{l}}_f \forall_x A(fx) \quad \text{for } A \text{ n.c.}$$
$$\forall_x \exists^{\mathrm{d}}_y A(y) \to \exists^{\mathrm{d}}_f \forall_x A(fx) \quad \text{for } A \text{ c.r.}$$

Theorem (Independence of premise). Assume $x \notin \mathrm{FV}(A)$.

$$(A \to \exists^{\mathrm{l}}_x B) \to \exists^{\mathrm{l}}_x (A \to B) \quad \text{for } A, B \text{ n.c.}$$
$$(A \to^{\mathrm{nc}} \exists^{\mathrm{l}}_x B) \to \exists^{\mathrm{l}}_x (A \to B) \quad \text{for } B \text{ n.c.}$$
$$(A \to \exists^{\mathrm{d}}_x B) \to \exists^{\mathrm{d}}_x (A \to B) \quad \text{for } A \text{ n.c., } B \text{ c.r.}$$
$$(A \to^{\mathrm{nc}} \exists^{\mathrm{d}}_x B) \to \exists^{\mathrm{d}}_x (A \to B) \quad \text{for } B \text{ c.r.}$$

Extracted terms

For derivations M^A with A n.c. let $\mathrm{et}(M^A) := \varepsilon$. Otherwise

$$\mathrm{et}(u^A) := v_u^{\tau(A)} \quad (v_u^{\tau(A)} \text{ uniquely associated to } u^A),$$

$$\mathrm{et}((\lambda_{u^A} M^B)^{A\to B}) := \begin{cases} \lambda_{v_u}^{\tau(A)} \mathrm{et}(M) & \text{if } A \text{ is c.r.} \\ \mathrm{et}(M) & \text{if } A \text{ is n.c.} \end{cases}$$

$$\mathrm{et}((M^{A\to B} N^A)^B) := \begin{cases} \mathrm{et}(M)\mathrm{et}(N) & \text{if } A \text{ is c.r.} \\ \mathrm{et}(M) & \text{if } A \text{ is n.c.} \end{cases}$$

$$\mathrm{et}((\lambda_{x^\rho} M^A)^{\forall_x A}) := \lambda_x^\rho \mathrm{et}(M),$$

$$\mathrm{et}((M^{\forall_x A(x)} r)^{A(r)}) := \mathrm{et}(M) r,$$

$$\mathrm{et}((\lambda_{u^A} M^B)^{A\to^{\mathrm{nc}} B}) := \mathrm{et}(M),$$

$$\mathrm{et}((M^{A\to^{\mathrm{nc}} B} N^A)^B) := \mathrm{et}(M),$$

$$\mathrm{et}((\lambda_{x^\rho} M^A)^{\forall_x^{\mathrm{nc}} A}) := \mathrm{et}(M),$$

$$\mathrm{et}((M^{\forall_x^{\mathrm{nc}} A(x)} r)^{A(r)}) := \mathrm{et}(M).$$

Extracted terms for the axioms.

- Let I be c.r.

$$\mathrm{et}(I_i^+) := \mathrm{C}_i, \qquad \mathrm{et}(I^-) := \mathcal{R},$$

  where both C_i and $\mathcal{R}$ refer to the algebra ι_I associated with I.

- For the invariance axioms we take identities.

Theorem (Soundness)

Let M be a derivation of a c.r. formula A from assumptions $u_i : C_i$ $(i < n)$. Then we can derive $\mathrm{et}(M)$ $\mathbf{r}$ A from assumptions v_{u_i} $\mathbf{r}$ C_i in case C_i is c.r. and C_i otherwise.

Proof.

By induction on M. □

Quotient and remainder

Recall `QR`: $\forall_{m,n}\exists_{q,r}(n = (m+1)q + r \wedge r < m+1)$.

```
(define eterm
 (proof-to-extracted-term (theorem-name-to-proof "QR")))
```

To display this term it is helpful to first add a variable name `p` for pairs of natural numbers and then normalize.

```
(add-var-name "p" (py "nat@@nat"))
(define neterm (rename-variables (nt eterm)))
(pp neterm)
```

This "normalized extracted term" `neterm` is the program we are looking for. To display it we write:

```
(pp neterm)
```

The output will be:

```
[n,n0](Rec nat=>nat@@nat)n0(0@0)
      ([n1,p][if (right p<n)
                 (left p@Succ right p)
                 (Succ left p@0)])
```

Here `[n,n0]` denotes abstraction on the variables `n,n0`, usually written by use of the λ notation. In more familiar terms:

$$f(m,0) = 0@0$$

$$f(m,n{+}1) = \begin{cases} \mathrm{left}(f(m,n))@\mathrm{right}(f(m,n)){+}1 & \text{if } \mathrm{right}(f(m,n)) < m \\ \mathrm{left}(f(m,n)) + 1@0 & \text{else} \end{cases}$$

List reversal

Recall

$$\texttt{ListInitLastNat}\colon \forall_{u,y}\exists_{v,x}(yu = vx).$$
$$\texttt{ExR}\colon \forall_{n,v}(n = |v| \to \exists_w Rvw).$$

```
(define eterm (proof-to-extracted-term proof))
(add-var-name "f" (py "list nat=>list nat"))
(add-var-name "p" (py "list nat@@nat"))
(define neterm (rename-variables (nt eterm)))
```

This "normalized extracted term" `neterm` is the program we are looking for. To display it we write `(pp neterm)`:

```
[x](Rec nat=>list nat=>list nat)x([v](Nil nat))
 ([x0,f,v]
   [if v
     (Nil nat)
     ([x1,v0][let p (cListInitLastNat v0 x1)
                (right p::f left p)])])
```

- animate / deanimate. Suppose a proof M of uses a lemma `L`. Then `cL` may appear in $\mathrm{et}(M)$. We may or may not add computation rules for `cL`.
- To obtain the `let` expression in the term above, we have used implicitly the "identity lemma" `Id`: $P \to P$; its realizer has the form $\lambda_{f,x}(fx)$. If `Id` is not animated, the extracted term has the form $\mathtt{cId}(\lambda_x M)N$, which is printed as [`let` x N M].

The term contains the constant `cListInitLastNat` denoting the content of the auxiliary proposition, and in the step the function defined recursively calls itself via `f`. The underlying algorithm defines an auxiliary function g by

$$\begin{aligned} g(0, v) &:= [], \\ g(n+1, []) &:= [], \\ g(n+1, xv) &:= \text{let } wy = xv \text{ in } y :: g(n, w) \end{aligned}$$

and gives the result by applying g to $|v|$ and v. It clearly takes quadratic time.

Binary trees

Recall

$$\mathrm{ExBK}\colon \forall_r \exists_\ell (|\ell| = \|r\| \land \forall_{n<|\ell|}((\ell)_n \in r) \land \mathrm{Incr}(\ell)).$$

```
(define eterm (proof-to-extracted-term
                (theorem-name-to-proof "ExBK")))
(define neterm (rename-variables (nt eterm)))
(pp neterm)
```

The result is

```
[r](Rec bin=>list list boole)r(Nil boole):
 ([r0,as,r1,as0]((Cons boole)True map as)++
                 ((Cons boole)False map as0)++(Nil boole):)
```

Computational content of classical proofs

Well known: from $\vdash \tilde{\exists}_y G$ with G quantifier-free one can read off an instance.

- Idea for a proof: replace $\bot$ anywhere in the derivation by $\exists_y G$.
- Then the end formula $\forall_y(G \to \bot) \to \bot$ is turned into

$$\forall_y(G \to \exists_y G) \to \exists_y G,$$

  and since the premise is trivially provable, we have the claim.

Unfortunately, this simple argument is not quite correct.

- G may contain $\bot$, hence changes under $\bot \mapsto \exists_y G$.
- we may have used axioms or lemmata involving $\bot$ (e.g., $\bot \to P$), which need not be derivable after the substitution.

But in spite of this, the simple idea can be turned into something useful.

Use the arithmetical falsity $\mathbf{F}$ rather than the logical one, $\bot$. Let $A^{\mathbf{F}}$ denote the result of substituting $\bot$ by $\mathbf{F}$ in A. Assume

$$\begin{aligned} &D^{\mathbf{F}} \to D, \\ &(G^{\mathbf{F}} \to \bot) \to G \to \bot. \end{aligned} \tag{1}$$

Using (1) we can now correct the argument: from the given derivation of $D \to \forall_y(G \to \bot) \to \bot$ we obtain

$$D^{\mathbf{F}} \to \forall_y(G^{\mathbf{F}} \to \bot) \to \bot,$$

since $D^{\mathbf{F}} \to D$ and $(G^{\mathbf{F}} \to \bot) \to G \to \bot$. Substituting $\bot$ by $\exists_y G^{\mathbf{F}}$ gives

$$D^{\mathbf{F}} \to \forall_y(G^{\mathbf{F}} \to \exists_y G^{\mathbf{F}}) \to \exists_y G^{\mathbf{F}}.$$

Since $\forall_y(G^{\mathbf{F}} \to \exists_y G^{\mathbf{F}})$ is derivable we obtain $D^{\mathbf{F}} \to \exists_y G^{\mathbf{F}}$.

Therefore we need to pick our assumptions D and goal formulas G from appropriately chosen sets $\mathcal{D}$ and $\mathcal{G}$ which guarantee (1).

An easy way to achieve this is to replace in D and G every atomic formula P different from $\bot$ by its double negation $(P \to \bot) \to \bot$. This corresponds to the original A-translation of Friedman (1978). However, then the computational content of the resulting constructive proof is unnecessarily complex, since each occurrence of $\bot$ gets replaced by the c.r. formula $\exists_y G^{\mathbf{F}}$.

Goal: eliminate unnecessary double negations. To this end we define sets $\mathcal{D}$ and $\mathcal{G}$ of formulas which ensure that their elements $D \in \mathcal{D}$ and $G \in \mathcal{G}$ satisfy the DG-property (1).

$\mathcal{D}$, $\mathcal{G}$, $\mathcal{R}$ and $\mathcal{I}$ are generated by the clauses

- $R,\ P,\ I \to D,\ \forall_x D \in \mathcal{D}$.
- $I,\ \bot,\ R \to G,\ D_0 \to G \in \mathcal{G}$.
- $\bot,\ G \to R,\ \forall_x R \in \mathcal{R}$.
- $P,\ D \to I,\ \forall_x I \in \mathcal{I}$.

Let $A^{\mathbf{F}} := A[\bot := \mathbf{F}]$, and $\neg A$, $\neg_\bot A$ abbreviate $A \to \mathbf{F}$, $A \to \bot$.

Lemma (Ishihara (2000))

We have derivations from $\mathbf{F} \to \bot$ *and* $\mathbf{F} \to P$ *of*

$$D^{\mathbf{F}} \to D,$$
$$G \to \neg_\bot \neg_\bot G^{\mathbf{F}},$$
$$\neg_\bot \neg R^{\mathbf{F}} \to R,$$
$$I \to I^{\mathbf{F}}.$$

We give some examples of definite and goal formulas. Keep in mind that $\mathcal{R} \subseteq \mathcal{D}$ and $\mathcal{I} \subseteq \mathcal{G}$.

- $P \in \mathcal{D} \cap \mathcal{I}$.
- $\bot \in \mathcal{R} \cap \mathcal{G}$.
- $P \to \bot \in \mathcal{R} \cap \mathcal{G}$.
- $(P \to \bot) \to \bot \in \mathcal{R} \cap \mathcal{G}$.

Lemma

$C \in \mathcal{D} \cap \mathcal{G}$ for C quantifier-free such that no implication in C has $\bot$ as its final conclusion, and $C \in \mathcal{R}$ $(\in \mathcal{I})$ if and only if $\bot$ is (is not) the final conclusion of C.

List reversal, weak form

From the clauses

$$\mathrm{InitR}\colon R([],[]),$$
$$\mathrm{GenR}\colon \forall_{v,w,x}(Rvw \to R(vx,xw)).$$

we prove

$$\forall_v \tilde{\exists}_w Rvw \qquad (:= \forall_v(\forall_w(Rvw \to \bot) \to \bot)).$$

Fix R, v and assume `InitR`, `GenR` and the "false" assumption $u\colon \forall_w \neg Rvw$; goal: $\bot$. To this end we prove that all initial segments of v are non-revertible, which contradicts `InitR`. More precisely, from u and `GenR` we prove

$$\forall_{v_2} A(v_2) \quad \text{with } A(v_2) := \forall_{v_1}(v_1 v_2 = v \to \forall_w \neg Rv_1 w)$$

by $\mathrm{Ind}(v_2)$. For $v_2 = []$ this follows from $u_0\colon v_1\,[] = v$ and u. For the step, assume $u_1\colon v_1(xv_2) = v$, fix w and assume $u_2\colon Rv_1 w$. Goal: $\bot$. We use the IH with $v_1 x$ and xw to obtain $\bot$. This requires (i) $(v_1 x)v_2 = v$ and (ii) $R(v_1 x, xw)$. But (i) follows from u_1 using properties of append, and (ii) follows from u_2 using `GenR`.

We formalize this proof, to prepare it for the refined A-translation. The following lemmata will be used:

$$\begin{aligned}\mathrm{Compat}'\colon&\quad \forall^{\mathrm{nc}}_{v,w}(v =^{\mathrm{d}} w \to Xw \to Xv),\\ \mathrm{EqToEqD}\colon&\ \forall_{v,w}(v = w \to v =^{\mathrm{d}} w).\end{aligned}$$

The proof term is

$$\begin{aligned}M :=&\lambda_{R,v}\lambda_{u_{\mathrm{InitR}}}\lambda_{u_{\mathrm{GenR}}}\lambda_u^{\forall_w\neg Rvw}(\\ &\mathrm{Ind}_{v_2,A(v_2)}vRvM_{\mathrm{Base}}M_{\mathrm{Step}}\ []\ \mathrm{Truth}^{[]\,v=v}\ []\ u_{\mathrm{InitR}})\end{aligned}$$

with

$$\begin{aligned}M_{\mathrm{Base}} :=&\ \lambda_{v_1}\lambda_{u_0}^{v_1[]=v}(\\ &\mathrm{Compat}'\,\{\,v \mid \forall_w\neg Rvw\,\}\,R\,v\,v_1\,v\,(\mathrm{EqToEqD}\,v_1\,v u_0)u),\\ M_{\mathrm{Step}} :=&\ \lambda_{x,v_2}\lambda_{u_0}^{A(v_2)}\lambda_{v_1}\lambda_{u_1}^{v_1(xv_2)=v}\lambda_w\lambda_{u_2}^{Rv_1w}(\\ &u_0(v_1x)u_1(xw)(u_{\mathrm{GenR}}v_1wxu_2)).\end{aligned}$$

Have $M\colon \forall_v \tilde{\exists}_w Rvw$ from $\mathrm{InitR}\colon D_1$ and $\mathrm{GenR}\colon D_2$, with $D_1 := R([],[])$ and $D_2 := \forall_{v,w,x}(Rvw \to R(vx, xw))$.

- We can replace $\bot$ throughout by $\exists_w Rvw$.
- $\tilde{\exists}_w Rvw := \neg\forall_w\neg Rvw := \forall_w(Rvw \to \bot) \to \bot$ is turned into $\forall_w(Rvw \to \exists_w Rvw) \to \exists_w Rvw$.
- Premise is an instance of $\exists^+$; hence we obtain $M^{\exists}\colon \exists_w Rvw$.
- Neither the D_i nor an axiom has $\bot$ in its uninstantiated formulas, hence correctness is not affected by the substitution.

The term `neterm` extracted in Minlog is

```
[R,v]
 (Rec list nat=>list nat=>list nat=>list nat)v([v0,v1]v1)
 ([x,v0,g,v1,v2]g(v1++x:)(x::v2)) (Nil nat) (Nil nat)
```

with g a variable for binary functions on lists. In fact, the underlying algorithm defines an auxiliary function h by

$$h([], v_1, v_2) := v_2, \qquad h(xv, v_1, v_2) := h(v, v_1 x, x v_2)$$

and gives the result by applying h to the original list and twice [].

- The second argument of h is not needed.
- Its presence makes the algorithm quadratic rather than linear, because in each recursion step $v_1 x$ is computed, and the list append function is defined by recursion on its first argument.
- We will be able to get rid of this superfluous second argument by decorating the proof.
- It will turn out that in the proof (by induction on v_2) of the formula $A(v_2) := \forall_{v_1}(v_1 v_2 = v \to \forall_w \neg R v_1 w))$, the variable v_1 is not used computationally.
- Hence, in the decorated version of the proof, we can use $\forall^{\mathrm{nc}}_{v_1}$.

1. Logic
2. The model of partial continuous functionals
3. Formulas as problems
4. Computational content of proofs
5. Decorating proofs

Decoration can simplify extracts

- Suppose that a proof M uses a lemma $L^{\mathrm{d}} \colon A \vee^{\mathrm{d}} B$.
- Then the extract $\mathrm{et}(M)$ will contain the extract $\mathrm{et}(L^{\mathrm{d}})$.
- Suppose that the only computationally relevant use of L^{d} in M was which one of the two alternatives holds true, A or B.
- Express this by using a weakened lemma $L \colon A \vee^{\mathrm{u}} B$.
- Since $\mathrm{et}(L)$ is a boolean, the extract of the modified proof is "purified": the (possibly large) extract $\mathrm{et}(L^{\mathrm{d}})$ has disappeared.

Decoration algorithm

- $\mathrm{Seq}(M)$ of a proof M consists of its context and end formula.
- The proof pattern $\mathrm{P}(M)$ of a proof M is the result of marking in c.r. parts of M (i.e., not above a n.c. formula) all occurrences of implications and universal quantifiers as n.c. (some restrictions apply on axioms and theorems).
- A formula D extends C if D is obtained from C by changing some $\to^{\mathrm{nc}}$, $\forall^{\mathrm{nc}}$ into $\to$, $\forall$.
- A proof N extends M if (i) N and M are the same up to variants of $\to$, $\forall$ in their formulas, and (ii) every formula in c.r. parts of M is extended by the corresponding one in N.

Decoration algorithm (ctd.)

- Assumption: For every axiom or theorem A and every decoration variant C of A we have another axiom or theorem whose formula D extends C, and D is the least among those extensions.
- Example: Induction

$$A'(0) \to^{\mathrm{c/nc}} \forall_n^{\mathrm{c/nc}}(A''(n) \to^{\mathrm{c/nc}} A'''(n{+}1))) \to^{\mathrm{c/nc}} \forall_n^{\mathrm{c/nc}} A''''(n).$$

Let A be the lub (w.r.t. deco) of $A', \ldots, A''''$. Extended axiom:

$$A(0) \to \forall_n(A(n) \to A(n+1))) \to \forall_n A(n).$$

Decoration algorithm (ctd.)

Theorem (Ratiu & S., 2010)

Under the assumption above, for every proof pattern U and every extension of its sequent $\mathrm{Seq}(U)$ we can find a decoration M_∞ of U such that

(a) *$\mathrm{Seq}(M_\infty)$ extends the given extension of $\mathrm{Seq}(U)$, and*

(b) *M_∞ is optimal in the sense that any other decoration M of U whose sequent $\mathrm{Seq}(M)$ extends the given extension of $\mathrm{Seq}(U)$ has the property that M also extends M_∞.*

Case $(\to^{\mathrm{nc}})^-$. Consider a proof pattern

$$\dfrac{\begin{array}{cc}\Phi,\Gamma & \Gamma,\Psi\\ \mid U & \mid V\\ A\to^{\mathrm{nc}} B & A\end{array}}{B}\,(\to^{\mathrm{nc}})^-$$

Given: extension $\Pi,\Delta,\Sigma\Rightarrow D$ of $\Phi,\Gamma,\Psi\Rightarrow B$. Alternating steps:

- $\mathrm{IH}_a(U)$ for extension $\Pi,\Delta\Rightarrow A\to^{\mathrm{nc}} D\mapsto$ decoration M_1 of U whose sequent $\Pi_1,\Delta_1\Rightarrow C_1\to^{\mathrm{c/nc}} D_1$ extends $\Pi,\Delta\Rightarrow A\to^{\mathrm{nc}} D$ ($\to^{\mathrm{c/nc}}\in\{\to^{\mathrm{nc}},\to\}$). Suffices if A is n.c.: extension $\Delta_1,\Sigma\Rightarrow C_1$ of V is a proof (in n.c. parts of a proof $\to^{\mathrm{nc}}$, $\forall^{\mathrm{nc}}$ and $\to$, $\forall$ are identified). For A c.r:
- $\mathrm{IH}_a(V)$ for the extension $\Delta_1,\Sigma\Rightarrow C_1\mapsto$ decoration N_2 of V whose sequent $\Delta_2,\Sigma_2\Rightarrow C_2$ extends $\Delta_1,\Sigma\Rightarrow C_1$.
- $\mathrm{IH}_a(U)$ for $\Pi_1,\Delta_2\Rightarrow C_2\to^{\mathrm{c/nc}} D_1\mapsto$ decoration M_3 of U whose sequent $\Pi_3,\Delta_3\Rightarrow C_3\to^{\mathrm{c/nc}} D_3$ extends $\Pi_1,\Delta_2\Rightarrow C_2\to^{\mathrm{c/nc}} D_1$.
- $\mathrm{IH}_a(V)$ for the extension $\Delta_3,\Sigma_2\Rightarrow C_3\mapsto$ decoration N_4 of V whose sequent $\Delta_4,\Sigma_4\Rightarrow C_4$ extends $\Delta_3,\Sigma_2\Rightarrow C_3$. ...

List reversal: decoration of the weak existence proof

We present our proof in more detail, particularly by writing proof trees with formulas. The decoration algorithm then is applied to its proof pattern with the sequent consisting of the context $\mathrm{InitR}\colon R([],[])$ and $\mathrm{GenR}\colon \forall_{v,w,x}(Rvw \to R(vx,xw))$ and the end formula $\forall_v \exists_w Rvw$. Relevant axioms: list induction, CompatRev and $\exists^+$.

$$\begin{array}{ll} \mathrm{CompatRev}\colon & \forall^{\mathrm{nc}}_{R,v,v_1,v_2}(v_1 =^{\mathrm{d}} v_2 \to \forall_w \neg^{\exists} Rv_2w \to \forall_w \neg^{\exists} Rv_1w), \\ \exists^+\colon & \forall^{\mathrm{nc}}_{R,v}\forall_w(Rvw \to \exists_w Rvw) \end{array}$$

with $A(v_2) := \forall^{\mathrm{nc}}_{v_1}(v_1v_2{=}v \to \forall_w \neg^{\exists} Rv_1w)$ and $\neg^{\exists} Rv_1w := Rv_1w \to \exists_w Rvw$.

$$\dfrac{\dfrac{\dfrac{\mathrm{Ind}\quad v\quad R\quad v}{A([]) \to \forall_{x,v_2}(A(v_2) \to A(xv_2)) \to A(v)} \quad A([])}{\forall_{x,v_2}(A(v_2) \to A(xv_2)) \to A(v)} \quad \forall_{x,v_2}(A(v_2) \to A(xv_2))}{\forall_{v_1}(v_1 v = v \to \forall_w \neg^{\exists} R v_1 w) \quad (= A(v))}$$

$\mid M_B$

$\mid M_S$

where

$$\begin{aligned}
&\mathrm{Ind}: \qquad \forall^{\mathrm{nc}}_{v,R} \forall_w (A([]) \to \forall_{x,v_2}(A(v_2) \to A(xv_2)) \to A(w)) \\
&A(v_2) := \forall_{v_1}(v_1 v_2 = v \to \forall_w \neg^{\exists} R v_1 w) \\
&\neg^{\exists} B := \quad B \to \exists_w R v w
\end{aligned}$$

Applied to [], Truth, [] and InitR this gives $\exists_w R v w$.

M_B

$$
\dfrac{\dfrac{\dfrac{\text{CompatRev} \quad R \quad v \quad v_1 \quad v}{v_1 =^{\mathrm{d}} v \to \forall_w \neg^{\exists} Rvw \to \forall_w \neg^{\exists} Rv_1 w} \qquad \begin{matrix}[u_1 \colon v_1\,[] = v] \\ \mid N_1 \\ v_1\,[] =^{\mathrm{d}} v\end{matrix}}{\forall_w \neg^{\exists} Rvw \to \forall_w \neg^{\exists} Rv_1 w} \qquad \dfrac{\exists^+ \quad R \quad v}{\forall_w \neg^{\exists} Rvw}}{\dfrac{\dfrac{\forall_w \neg^{\exists} Rv_1 w}{v_1\,[] = v \to \forall_w \neg^{\exists} Rv_1 w} \to^+ u_1}{\forall_{v_1}(v_1\,[] = v \to \forall_w \neg^{\exists} Rv_1 w) \quad (= A([]))}}
$$

with N_1 involving $\mathrm{EqToEqD} \colon \forall_{v,w}(v = w \to v =^{\mathrm{d}} w)$, and

$$
\begin{array}{ll}
\mathrm{CompatRev} \colon & \forall^{\mathrm{nc}}_{R,v,v_1,v_2}(v_1 =^{\mathrm{d}} v_2 \to \forall_w \neg^{\exists} Rv_2 w \to \forall_w \neg^{\exists} Rv_1 w) \\
\exists^+ \colon & \forall^{\mathrm{nc}}_{R,v} \forall_w (Rvw \to \exists_w Rvw) \\
\neg^{\exists} B := & B \to \exists_w Rvw
\end{array}
$$

M_S

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{[u_0\colon A(v_2)]\quad v_1x}{(v_1x)v_2{=}v \to \forall_w \neg^{\exists} R(v_1x,w)}\quad [u_1\colon v_1(xv_2){=}v]}{\forall_w \neg^{\exists} R(v_1x,w)}\quad xw}{\neg^{\exists} R(v_1x,xw)}\qquad \begin{array}{c}[u_2\colon Rv_1w]\\ \mid N_2\\ R(v_1x,xw)\end{array}}{\dfrac{\exists_w Rvw}{\neg^{\exists} Rv_1w}\to^{+} u_2}}{\forall_w \neg^{\exists} Rv_1w}}{v_1(xv_2) = v \to \forall_w \neg^{\exists} Rv_1w}\to^{+} u_1}{\forall^{\mathrm{nc}}_{v_1}(v_1(xv_2){=}v \to \forall_w \neg^{\exists} Rv_1w)\ (=A(xv_2))}}{A(v_2) \to A(xv_2)}\to^{+}}{\forall_{x,v_2}(A(v_2) \to A(xv_2))}$$

with N_2 involving $\mathrm{GenR}\colon \forall_{v,w,x}(Rvw \to R(vx,xw))$.

$$\dfrac{\dfrac{\dfrac{\dfrac{\mathrm{Ind}\quad v\quad R\quad v}{\hat{\mathrm{A}}([])\to^{\mathrm{nc}}\forall_{x,v_2}(\hat{\mathrm{A}}(v_2)\to^{\mathrm{nc}}\hat{\mathrm{A}}(xv_2))\to^{\mathrm{nc}}\hat{\mathrm{A}}(v)}\ |M_B \quad \hat{\mathrm{A}}([])}}{\forall_{x,v_2}(\hat{\mathrm{A}}(v_2)\to^{\mathrm{nc}}\hat{\mathrm{A}}(xv_2))\to^{\mathrm{nc}}\hat{\mathrm{A}}(v)}\quad |\,M_S \quad \forall_{x,v_2}(\hat{\mathrm{A}}(v_2)\to^{\mathrm{nc}}\hat{\mathrm{A}}(xv_2))}{\forall^{\mathrm{nc}}_{v_1}(v_1v=v\to\forall^{\mathrm{nc}}_{w}\neg^{\exists}Rv_1w)\quad (=\hat{\mathrm{A}}(v))}$$

where

$$\begin{aligned}
&\mathrm{Ind}: && \forall^{\mathrm{nc}}_{v,R}\forall_w(\hat{\mathrm{A}}([])\to\forall_{x,v_2}(\hat{\mathrm{A}}(v_2)\to\hat{\mathrm{A}}(xv_2))\to\hat{\mathrm{A}}(w))\\
&\hat{\mathrm{A}}(v_2) := && \forall^{\mathrm{nc}}_{v_1}(v_1v_2=v\to\forall^{\mathrm{nc}}_{w}\neg^{\exists}Rv_1w)\\
&\neg^{\exists}B := && B\to\exists_wRvw
\end{aligned}$$

Applied to [], Truth, [] and InitR this gives $\exists_wRvw$.

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{}{\mathrm{CompatRev}\quad R\quad v\quad v_1\quad v}}{v_1 =^{\mathrm{d}} v \to \forall^{\mathrm{nc}}_w \neg^{\exists} Rvw \to \forall^{\mathrm{nc}}_w \neg^{\exists} Rv_1 w} \qquad \dfrac{[u_1 \colon v_1\,[] = v]}{\mid N_1}}{v_1\,[] =^{\mathrm{d}} v}}{\forall^{\mathrm{nc}}_w \neg^{\exists} Rvw \to \forall^{\mathrm{nc}}_w \neg^{\exists} Rv_1 w} \qquad \dfrac{\exists^{+}\quad R\quad v}{\forall^{\mathrm{nc}}_w \neg^{\exists} Rvw}}{\dfrac{\forall^{\mathrm{nc}}_w \neg^{\exists} Rv_1 w}{v_1\,[] = v \to \forall^{\mathrm{nc}}_w \neg^{\exists} Rv_1 w} \to^{+} u_1}}{\forall^{\mathrm{nc}}_{v_1}(v_1\,[] = v \to \forall^{\mathrm{nc}}_w \neg^{\exists} Rv_1 w) \quad (= \hat{\mathrm{A}}([]))}$$

with

$$\begin{aligned}
&\mathrm{CompatRev}\colon && \forall^{\mathrm{nc}}_{R,v,v_1,v_2}(v_1 =^{\mathrm{d}} v_2 \to \forall^{\mathrm{nc}}_w \neg^{\exists} Rv_2 w \to \forall^{\mathrm{nc}}_w \neg^{\exists} Rv_1 w)\\
&\exists^{+}\colon && \forall^{\mathrm{nc}}_{R,v} \forall_w (Rvw \to \exists_w Rvw)\\
&\neg^{\exists} B := && B \to \exists_w Rvw
\end{aligned}$$

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{}{v_1 =^{\mathrm{d}} v \to \forall_w \neg^{\exists} R v w \to \forall_w \neg^{\exists} R v_1 w}\ \mathrm{CompatRev}\quad R\quad v\quad v_1\quad v \qquad [u_1 \colon v_1\,[] = v] \mid N_1}{v_1\,[] =^{\mathrm{d}} v}}{\forall_w \neg^{\exists} R v w \to \forall_w \neg^{\exists} R v_1 w} \qquad \dfrac{\exists^{+}\quad R\quad v}{\forall_w \neg^{\exists} R v w}}{\forall_w \neg^{\exists} R v_1 w}}{v_1\,[] = v \to \forall_w \neg^{\exists} R v_1 w} \to^{+} u_1}{\forall^{\mathrm{nc}}_{v_1}(v_1\,[] = v \to \forall_w \neg^{\exists} R v_1 w) \quad (= A'([]))}$$

with

$$\begin{aligned}
&\mathrm{CompatRev}\colon && \forall^{\mathrm{nc}}_{R,v,v_1,v_2}(v_1 =^{\mathrm{d}} v_2 \to \forall_w \neg^{\exists} R v_2 w \to \forall_w \neg^{\exists} R v_1 w)\\
&\exists^{+}\colon && \forall^{\mathrm{nc}}_{R,v} \forall_w (R v w \to \exists_w R v w)\\
&\neg^{\exists} B := && B \to \exists_w R v w
\end{aligned}$$

$$\dfrac{\dfrac{\dfrac{\mathrm{Ind}\quad v\quad R\quad v}{\hat{\mathrm{A}}([]) \to \forall_{x,v_2}(\hat{\mathrm{A}}(v_2) \to \hat{\mathrm{A}}(xv_2)) \to \hat{\mathrm{A}}(v)} \mid M_B \quad A'([])}{\forall_{x,v_2}(\hat{\mathrm{A}}(v_2) \to^{\mathrm{nc}} \hat{\mathrm{A}}(xv_2)) \to^{\mathrm{nc}} \hat{\mathrm{A}}(v)} \quad \forall_{x,v_2}(\hat{\mathrm{A}}(v_2) \to^{\mathrm{nc}} \hat{\mathrm{A}}(xv_2)) \mid M_S}{\forall^{\mathrm{nc}}_{v_1}(v_1 v = v \to \forall^{\mathrm{nc}}_{w} \neg^{\exists} R v_1 w) \quad (= \hat{\mathrm{A}}(v))}$$

where

$$\begin{aligned}
\hat{\mathrm{A}}(v_2) &:= \forall^{\mathrm{nc}}_{v_1}(v_1 v_2 = v \to \forall^{\mathrm{nc}}_{w} \neg^{\exists} R v_1 w) \\
A'(v_2) &:= \forall^{\mathrm{nc}}_{v_1}(v_1 v_2 = v \to \forall_{w} \neg^{\exists} R v_1 w) \\
\neg^{\exists} B &:= \quad B \to \exists_w R v w
\end{aligned}$$

Applied to [], Truth, [] and InitR this gives $\exists_w R v w$.

$$\dfrac{\dfrac{\dfrac{\mathrm{Ind}\quad v\quad R\quad v}{A'([]) \to \forall_{x,v_2}(A'(v_2) \to A'(xv_2)) \to A'(v)} \quad A'([])}{\forall_{x,v_2}(A'(v_2) \to A'(xv_2)) \to A'(v)} \quad \forall^{\mathrm{nc}}_{x,v_2}(\hat{\mathrm{A}}(v_2) \to^{\mathrm{nc}} \hat{\mathrm{A}}(xv_2))}{\forall^{\mathrm{nc}}_{v_1}(v_1 v = v \to \forall^{\mathrm{nc}}_{w} \neg^{\exists} R v_1 w) \quad (= \hat{\mathrm{A}}(v))} \quad \begin{array}{l} |M_B \\ |\ M_S \end{array}$$

where

$$\begin{aligned} \hat{\mathrm{A}}(v_2) &:= \forall^{\mathrm{nc}}_{v_1}(v_1 v_2 = v \to \forall^{\mathrm{nc}}_{w} \neg^{\exists} R v_1 w) \\ A'(v_2) &:= \forall^{\mathrm{nc}}_{v_1}(v_1 v_2 = v \to \forall_{w} \neg^{\exists} R v_1 w) \\ \neg^{\exists} B &:= \quad B \to \exists_w R v w \end{aligned}$$

Applied to [], Truth, [] and InitR this gives $\exists_w R v w$.

$$\dfrac{\dfrac{\dfrac{\dfrac{[u_0\colon \hat{\mathrm{A}}(v_2)] \quad v_1 x}{(v_1x)v_2{=}v \to \forall^{\mathrm{nc}}_w \neg^{\exists} R(v_1x,w)} \quad [u_1\colon v_1(xv_2){=}v]}{\forall^{\mathrm{nc}}_w \neg^{\exists} R(v_1x,w)} \quad xw}{\neg^{\exists} R(v_1x,xw)} \qquad \begin{array}{c}[u_2\colon Rv_1w] \\ \mid N_2 \\ R(v_1x,xw)\end{array}}{\dfrac{\dfrac{\dfrac{\exists_w Rvw}{\neg^{\exists} Rv_1w} \to^{+} u_2}{\forall^{\mathrm{nc}}_w \neg^{\exists} Rv_1w}}{v_1(xv_2) = v \to \forall^{\mathrm{nc}}_w \neg^{\exists} Rv_1w} \to^{+} u_1}$$

$$\dfrac{\forall^{\mathrm{nc}}_{v_1}(v_1(xv_2){=}v \to \forall^{\mathrm{nc}}_w \neg^{\exists} Rv_1w) \ (=\hat{\mathrm{A}}(xv_2))}{\hat{\mathrm{A}}(v_2) \to^{\mathrm{nc}} \hat{\mathrm{A}}(xv_2)} \to^{+}$$

$$\forall^{\mathrm{nc}}_{x,v_2}(\hat{\mathrm{A}}(v_2) \to^{\mathrm{nc}} \hat{\mathrm{A}}(xv_2))$$

This $P(M_S)$ with extension $\forall_{x,v_2}(A'(v_2) \to A'(xv_2))$ yields

$$\dfrac{\dfrac{\dfrac{[u_0\colon A'(v_2)] \quad v_1x}{(v_1x)v_2{=}v \to \forall_w \neg^{\exists} R(v_1x,w)} \quad [u_1\colon v_1(xv_2){=}v]}{\forall_w \neg^{\exists} R(v_1x,w)} \quad xw \qquad [u_2\colon Rv_1w]}{\neg^{\exists} R(v_1x,xw)} \qquad \begin{array}{c} \mid N_2 \\ R(v_1x,xw) \end{array}}{\dfrac{\dfrac{\dfrac{\exists_w Rvw}{\neg^{\exists} Rv_1w} \to^+ u_2}{\forall_w \neg^{\exists} Rv_1w}}{v_1(xv_2) = v \to \forall_w \neg^{\exists} Rv_1w} \to^+ u_1}$$

$$\dfrac{\forall^{\mathrm{nc}}_{v_1}(v_1(xv_2){=}v \to \forall_w \neg^{\exists} Rv_1w) \ \ (=A'(xv_2))}{A'(v_2) \to A'(xv_2)} \to^+$$

$$\forall_{x,v_2}(A'(v_2) \to A'(xv_2))$$

Finally

$$\dfrac{\dfrac{\dfrac{\mathrm{Ind}\quad v\quad R\quad v}{A'([]) \to \forall_{x,v_2}(A'(v_2) \to A'(xv_2)) \to A'(v)}\ |M_B \quad A'([])}{\forall_{x,v_2}(A'(v_2) \to A'(xv_2)) \to A'(v)}\ |\, M_S \qquad \forall_{x,v_2}(A'(v_2) \to A'(xv_2))}{\forall^{\mathrm{nc}}_{v_1}(v_1 v = v \to \forall_w \neg^{\exists} R v_1 w) \quad (= A'(v))}$$

where

$$A'(v_2) := \forall^{\mathrm{nc}}_{v_1}(v_1 v_2 = v \to \forall_w \neg^{\exists} R v_1 w)$$
$$\neg^{\exists} B := \quad B \to \exists_w R v w$$

Applied to [], Truth, [] and InitR this gives $\exists_w R v w$.

The extracted term `neterm` then is

```
[R,v](Rec list nat=>list nat=>list nat)v([v0]v0)
  ([x,v0,f,v1]f(x::v1))(Nil nat)
```

with `f` a variable for unary functions on lists. To run this algorithm one normalizes the term obtained by applying `neterm` to a list:

```
(pp (nt (mk-term-in-app-form neterm (pt "1::2::3::4:"))))
```

The returned value is the reverted list `4::3::2::1:`. This time, the underlying algorithm defines an auxiliary function g by

$$g([], w) := w, \qquad g(x :: v, w) := g(v, x :: w)$$

and gives the result by applying g to the original list and []. In conclusion, we have obtained (by machine extraction from an automated decoration of a weak existence proof) the standard linear algorithm for list reversal, with its use of an accumulator.

Fibonacci numbers

An application of decoration occurs when one derives double induction

$$\forall_n(Qn \to Q(Sn) \to Q(S(Sn))) \to \forall_n(Q0 \to Q1 \to Qn)$$

in *continuation passing style*, i.e., not directly, but using as an intermediate assertion (proved by induction)

$$\forall_{n,m}((Qn \to Q(Sn) \to Q(n+m)) \to Q0 \to Q1 \to Q(n+m)).$$

After decoration, the formula becomes

$$\forall_n \forall_m^{\mathrm{nc}}((Qn \to Q(Sn) \to Q(n+m)) \to Q0 \to Q1 \to Q(n+m)).$$

This can be applied to obtain a continuation based tail recursive definition of the Fibonacci function, from a proof of its totality. Let G be the (n.c.) graph of the Fibonacci function, defined by the clauses

$$G(0,0), \quad G(1,1),$$
$$\forall_{n,v,w}(G(n,v) \to G(Sn,w) \to G(S(Sn),v+w)).$$

From these assumptions one can easily derive

$$\forall_n \exists_v G(n,v),$$

using double induction (proved in continuation passing style). The term extracted from this proof is

```
[n](Rec nat=>nat=>(nat=>nat=>nat)=>nat=>nat=>nat)n([n0,k]k)
 ([n0,p,n1,k]p(Succ n1)([n2,n3]k n3(n2+n3)))
```

applied to `0`, `([n0,n1]n0)`, `0` and `1`.

Unclean aspect: that the recursion operator has value type

```
nat=>(nat=>nat=>nat)=>nat=>nat=>nat
```

rather than `(nat=>nat=>nat)=>nat=>nat=>nat`, which would correspond to an iteration. We can repair this by decoration. After decoration, the extracted term becomes

```
[n](Rec nat=>(nat=>nat=>nat)=>nat=>nat=>nat)n([k]k)
 ([n0,p,k]p([n1,n2]k n2(n1+n2)))
```

applied to `([n0,n1]n0)`, `0` and `1` (`k`, `p` are variables of type `nat=>nat=>nat` and `(nat=>nat=>nat)=>nat=>nat=>nat`, respectively.) This is iteration in continuation passing style: the functional F recursively defined by

$$F(0,k) := k$$
$$F(n+1,k) := F(n, \lambda_{n,n'}(k(n', n+n')))$$

is applied to n, the left projection $\lambda_{n_0,n_1} n_0$ and $0, 1$.

Example: Euler's φ, or avoiding factorization

Let $P(n)$ mean "n is prime". Consider

$$\begin{array}{ll} \texttt{Fact}\colon \forall_n(P(n) \vee^{\mathrm{r}} \exists_{m,k>1}(n = mk)) & \text{factorization,} \\ \texttt{PTest}\colon \forall_n(P(n) \vee^{\mathrm{u}} \exists_{m,k>1}(n = mk)) & \text{prime number test.} \end{array}$$

Euler's φ has the properties

$$\begin{cases} \varphi(n) = n - 1 & \text{if } P(n), \\ \varphi(n) < n - 1 & \text{if } n \text{ is composed.} \end{cases}$$

Using factorization and these properties we obtain a proof of

$$\forall_n(\varphi(n) = n - 1 \vee^{\mathrm{u}} \varphi(n) < n - 1).$$

Goal: get rid of the expensive factorization algorithm in the computational content, via decoration.

Example: Euler's φ, or avoiding factorization (ctd.)

How could the better proof be found? Recall that we assumed

$$\texttt{Fact}\colon \forall_n(P(n) \vee^{\mathrm{r}} \exists_{m,k>1}(n = mk)),$$
$$\texttt{PTest}\colon \forall_n(P(n) \vee^{\mathrm{u}} \exists_{m,k>1}(n = mk))$$

and have a proof of $\forall_n(\varphi(n) = n-1 \vee^{\mathrm{u}} \varphi(n) < n-1)$ from `Fact`.

- The decoration algorithm arrives at `Fact` with goal

$$P(n) \vee^{\mathrm{u}} \exists_{m,k>1}(n = mk).$$

- `PTest` fits as well, and it has $\vee^{\mathrm{u}}$ rather than $\vee^{\mathrm{r}}$, hence is preferred.

```
(define decnproof (fully-decorate nproof "Fact" "PTest"))
(proof-to-expr-with-formulas decnproof) =>
Elim: allnc n((C n -> F) oru C n ->
 ((C n -> F) -> phi n=n--1 oru phi n<n--1) ->
 (C n --> phi n=n--1 oru phi n<n--1) ->
 phi n=n--1 oru phi n<n--1)
PTest: all n((C n -> F) oru C n)
Intro: allnc n(phi n=n--1 -> phi n=n--1 oru phi n<n--1)
EulerPrime: allnc n((C n -> F) -> phi n=n--1)
Intro: allnc n(phi n<n--1 -> phi n=n--1 oru phi n<n--1)
EulerComp: allnc n(C n -> phi n<n--1)

(lambda (n)
  ((((Elim n) (PTest n))
     (lambda (u1542) ((Intro n) ((EulerPrime n) u1542))))
    (lambda (u1544) ((Intro n) ((EulerComp n) u1544)))))

(pp (nt (proof-to-extracted-term decnproof))) => cPTest
```

Example: Maximal Scoring Segment (MSS)

- Let X be linearly ordered by $\preceq$. Given $\mathrm{seg}\colon \mathbf{N} \to \mathbf{N} \to X$.
  Want: maximal segment

$$\forall_n \exists_{i \le k \le n} \forall_{i' \le k' \le n} (\mathrm{seg}(i', k') \preceq \mathrm{seg}(i, k)).$$

- Example: Regions with high G, C content in DNA.

$$X := \{G, C, A, T\},$$
$$g\colon \mathbf{N} \to X \quad \text{(gene)},$$
$$f\colon \mathbf{N} \to \mathbf{Z}, \quad f(i) := \begin{cases} 1 & \text{if } g(i) \in \{G, C\}, \\ -1 & \text{if } g(i) \in \{A, T\}, \end{cases}$$
$$\mathrm{seg}(i, k) = f(i) + \cdots + f(k).$$

Example: MSS (ctd.)

Prove the existence of a maximal segment by induction on n, simultaneously with the existence of a maximal end segment.

$$\forall_n(\exists_{i\leq k\leq n}\forall_{i'\leq k'\leq n}(\mathrm{seg}(i',k') \preceq \mathrm{seg}(i,k)) \wedge \exists_{j\leq n}\forall_{j'\leq n}(\mathrm{seg}(j',n) \preceq \mathrm{seg}(j,n)))$$

In the step:

- Compare the maximal segment i, k for n with the maximal end segment $j, n+1$ proved separately.
- If $\preceq$, take the new i, k to be $j, n+1$. Else take the old i, k.

Depending on how the existence of a maximal end segment was proved, we obtain a quadratic or a linear algorithm.

Example: MSS (ctd.)

Two proofs of the existence of a maximal end segment for $n+1$:
$\forall_n \exists_{j \le n+1} \forall_{j' \le n+1} (\mathrm{seg}(j', n+1) \preceq \mathrm{seg}(j, n+1))$.

- Introduce an auxiliary parameter m; prove by induction on m

$$\forall_n \forall_{m \le n+1} \exists_{j \le n+1} \forall_{j' \le m} (\mathrm{seg}(j', n+1) \preceq \mathrm{seg}(j, n+1)).$$

- Use $\mathtt{ES}_n$: $\exists_{j \le n} \forall_{j' \le n} (\mathrm{seg}(j', n) \preceq \mathrm{seg}(j, n))$ and the additional assumption of monotonicity

$$\forall_{i,j,n} (\mathrm{seg}(i, n) \preceq \mathrm{seg}(j, n) \to \mathrm{seg}(i, n+1) \preceq \mathrm{seg}(j, n+1)).$$

Proceed by cases on $\mathrm{seg}(j, n+1) \preceq \mathrm{seg}(n+1, n+1)$.
If $\preceq$, take $n+1$, else the previous j.

Example: MSS (ctd.)

Could decoration help to find the better proof? Have lemmas L:

$$\forall_n \forall_{m \leq n+1} \exists_{j \leq n+1} \forall_{j' \leq m} (\mathrm{seg}(j', n{+}1) \preceq \mathrm{seg}(j, n{+}1))$$

and LMon:

$$\mathtt{Mon} \to \forall_n (\mathtt{ES}_n \to \forall^{\mathrm{nc}}_{m \leq n+1} \exists_{j \leq n+1} \forall_{j' \leq m} (\mathrm{seg}(j', n{+}1) \preceq \mathrm{seg}(j, n{+}1))).$$

- The decoration algorithm arrives at L with goal

$$\forall^{\mathrm{nc}}_{m \leq n+1} \exists_{j \leq n+1} \forall_{j' \leq m} (\mathrm{seg}(j', n{+}1) \preceq \mathrm{seg}(j, n{+}1)).$$

- LMon fits as well, its assumptions Mon and $\mathtt{ES}_n$ are in the context, and it is less extended ($\forall^{\mathrm{nc}}_{m \leq n+1}$ rather than $\forall_{m \leq n+1}$), hence is preferred.