Extracting programs from proofs

Helmut Schwichtenberg

Mathematisches Institut, LMU, München

University of Canterbury, Christchurch, 17 Feb 2016

- Ishihara's trick
- Logic for Gray-code computation (j.w.w. Ulrich Berger, Kenji Miyamoto and Hideki Tsuiki)

Theorem (Ishihara's trick)

Let f be a linear map from a Banach space X into a normed space Y, and let (u_n) be a sequence in X converging to 0. Then for $0 < a < b$ either $a \leq \|fu_n\|$ for some n or $\|fu_n\| \leq b$ for all n.

Proof. Let M be a modulus of convergence of (u_n) to 0; assume $M0 = 0$. Call m a hit on n if $M_n \leq m < M_{n+1}$ and $a \leq \|fu_m\|$. First goal: define a function $h\colon \mathbb{N} \to \mathbb{N}$ such that

- $h_n = 0$ if for all $n' \leq n$ there is no hit;
- $h_n = m + 2$ if at n for the first time we have a hit, with m;
- $h_n = 1$ if there is an $n' < n$ with a hit.

We will need the bounded least number operator $\mu_n g$ defined recursively as follows (g a variable of type $\mathbf{N} \to \mathbf{B}$).

$$\mu_0 g := 0,$$
$$\mu_{Sn} g := \begin{cases} 0 & \text{if } g0 \\ S\mu_n(g \circ S) & \text{otherwise.} \end{cases}$$

From $\mu_n g$ we define

$$\mu_{n_0}^{n} g := \begin{cases} (\mu_{n-n_0} \lambda_m g(m+n_0)) + n_0 & \text{if } n_0 \leq n \\ 0 & \text{otherwise.} \end{cases}$$

To define h we use a function g of type $\mathbf{N} \to \mathbf{B}$ (to be defined from `cApproxSplit`) such that

$$\begin{cases} a \leq \|fu_m\| & \text{if } gm \\ \|fu_m\| \leq b & \text{otherwise.} \end{cases}$$

Then we can define $h_n := H(g, M, n)$ where

$$H(g, M, n) := \begin{cases} 0 & \text{if } M_n \leq \mu_{M_n} g \text{ and } M_{n+1} \leq \mu_{M_n}^{M_{n+1}} g \\ \mu_{M_n}^{M_{n+1}} g + 2 & \text{if } M_n \leq \mu_{M_n} g \text{ and } \mu_{M_n}^{M_{n+1}} g < M_{n+1} \\ 1 & \text{if } \mu_{M_n} g < M_n. \end{cases}$$

Next goal: define from h a sequence (v_n) in X such that

- $v_n = 0$ if $h_n = 0$;
- $v_n = nu_m$ if $h_n = m + 2$;
- $v_n = v_{n-1}$ if $h_n = 1$.

Let ξ be the type of elements of X, and $us\colon \mathbf{N} \to \xi$ a variable.
Define $v_n := V_\xi(g, M, us, n)$ where (writing u_m for $us(m)$)

$$V_\xi(g, M, us, n) := \begin{cases} 0 & \text{if } H(g, M, n) = 0 \\ nu_m & \text{if } H(g, M, n) = m + 2 \\ 0 \quad \text{(arbitrary)} & \text{if } H(g, M, n) = 1 \text{ and } n = 0 \\ V_\xi(g, M, us, n-1) & \text{if } H(g, M, n) = 1 \text{ and } n > 0. \end{cases}$$

One can show that (v_n) has the properties listed above.

Next we show that (v_n) is a Cauchy sequence with modulus $N(k) := 2k + 1$, which satisfies

$$\frac{N(k)+1}{2^{N(k)}} \leq \frac{1}{2^k}.$$

Since our goal is stable, we may employ arbitrary case distinctions (here: there is a hit / there is no hit).

By the assumed completeness of X we have a limit v of (v_n). Pick n_0 such that $\|fv\| \leq n_0 a$. Assume that there is a first hit at some $n > n_0$, with value m. Then $v = v_n = nu_m$ and

$$na \leq n\|fu_m\| = \|n(fu_m)\| = \|f(nu_m)\| = \|fv\| \leq n_0 a < na,$$

a contradiction. Hence beyond this n_0 we cannot have a first hit.

If $\forall_{n \leq n_0} h_n = 0$ then there is no hit and we have $\|fu_n\| \leq b$ for all n. Otherwise there is a hit before n_0, hence $a \leq \|fu_n\|$ for some n.

The computational content machine extracted from this proof is

```
[f,us,M,a,a0,k]
 [let g
   ([n]negb(cAC([n0]cApproxSplitBooleRat
                    a a0 lnorm(f(us n0))k)n))
   [case (H g M
          (cRealPosRatBound
           lnorm(f((cXCompl xi)
                   ((V xi)g M us)
                   ([k0]abs(IntS(2*k0)max 0))))
           a))
    (Zero -> False)
    (Succ n -> True)]]
```

Here H and V are the functionals defined above.

cAC is the computational content of the axiom of choice

```
(pp "AC")
all m ex boole (Pvar nat boole)^ m boole ->
ex g all m (Pvar nat boole)^ m(g m)
```

and hence the identity. cApproxSplitBooleRat and cRealPosRatBound are the computational content of lemmata

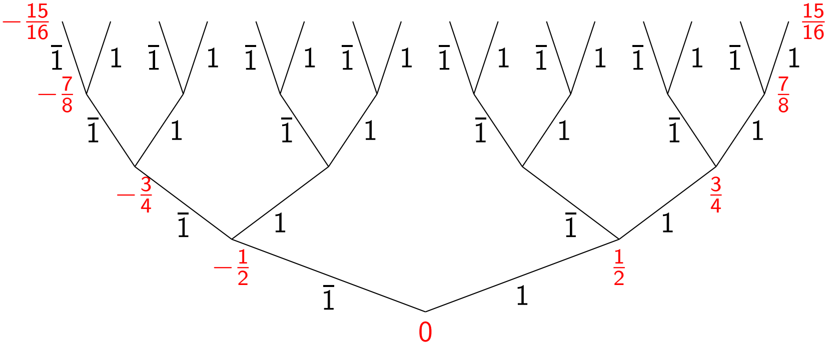
```
all a,b,x,k(Real x -> 1/2**k<=b-a ->
 ex boole((boole -> x<<=b) andu ((boole -> F) -> a<<=x)))
```

```
all x,a(Real x -> 0<a -> ex n x<<=n*a)
```

- Ishihara's trick
- Logic for Gray-code computation (j.w.w. Ulrich Berger, Kenji Miyamoto and Hideki Tsuiki)

Dyadic rationals:

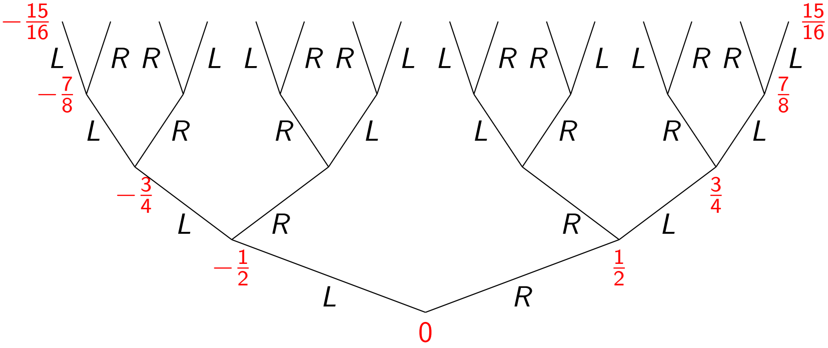
$$\sum_{i<k} \frac{a_i}{2^{i+1}} \qquad \text{with } a_i \in \{-1, 1\} =: \textbf{PSD}.$$



with $\bar{1} := -1$. Adjacent dyadics can differ in many digits:

$$\frac{7}{16} \sim 1\bar{1}11, \qquad \frac{9}{16} \sim 11\bar{1}\bar{1}.$$

Cure: flip after 1. Binary reflected (or Gray-) code.



$$\frac{7}{16} \sim \mathrm{RRRL}, \qquad \frac{9}{16} \sim \mathrm{RLRL}.$$

Problem with productivity:

$$\bar{1}111 + 1\bar{1}\bar{1}\bar{1} = ? \qquad \text{(what is the first digit?)}$$

Cure: delay.

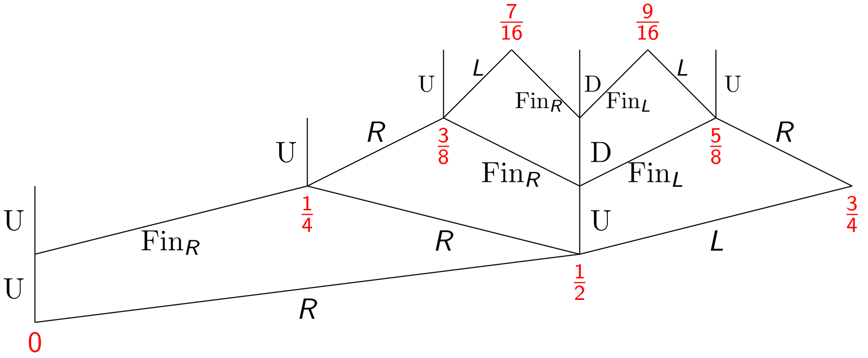
- For binary code: add 0. Signed digit code

$$\sum_{i<k} \frac{d_i}{2^{i+1}} \qquad \text{with } d_i \in \{-1, 0, 1\} =: \mathbf{SD}.$$

  Widely used for real number computation.

- For Gray-code: add U, D, $\mathrm{Fin}_{L/R}$. Pre-Gray code.

Pre-Gray code



After computation in pre-Gray code, one can remove Fin_a up to $\frac{1}{2^k}$:

$$\mathrm{U} \circ \mathrm{Fin}_a \mapsto a \circ \mathrm{R}, \qquad \mathrm{D} \circ \mathrm{Fin}_a \mapsto \mathrm{Fin}_a \circ \mathrm{L},$$

Goal: extract algorithms on infinite objects from proofs (in TCF).
Example:

- Infinite objects: streams, in pre-Gray code.
- Algorithm: average.

Framework:

- Constructive logic
- Types: only function types (Scott/Ershov partial continuous functionals), over base types given by constructors (may contain infinite objects).
- Inductive & coinductive predicates, with their least & greatest fixed point axioms (i.e., induction & coinduction).

We will coinductively define a predicate ${}^{\mathrm{co}}G$ and prove

$$\forall^{\mathrm{nc}}_{x,x'}({}^{\mathrm{co}}G(x) \to {}^{\mathrm{co}}G(x') \to {}^{\mathrm{co}}G(\frac{x+x'}{2})) \tag{1}$$

($\forall^{\mathrm{nc}}_{x,x'}$: the reals x, x' have no computational significance). Associated with ${}^{\mathrm{co}}G$ is its realizability extension $({}^{\mathrm{co}}G)^{\mathbf{r}}(p,x)$ (p is a stream representation of x witnessing ${}^{\mathrm{co}}G(x)$).

Soundness theorem:

$$({}^{\mathrm{co}}G)^{\mathbf{r}}(p,x) \to ({}^{\mathrm{co}}G)^{\mathbf{r}}(p',x') \to ({}^{\mathrm{co}}G)^{\mathbf{r}}(f(p,p'),\frac{x+x'}{2})$$

for some stream transformer f extracted from the proof of (1), which never mentions streams.

What is ${}^{\mathrm{co}}G$? Need simultaneously ${}^{\mathrm{co}}H$.

$$\Gamma(X,Y) := \{\, y \mid \exists^{\mathrm{r}}_{x\in X}\exists^{\mathrm{l}}_{a}(y = -a\frac{x-1}{2}) \vee^{\mathrm{d}} \exists^{\mathrm{r}}_{x\in Y}(y = \frac{x}{2})\,\},$$
$$\Delta(X,Y) := \{\, y \mid \exists^{\mathrm{r}}_{x\in X}\exists^{\mathrm{l}}_{a}(y = a\frac{x+1}{2}) \vee^{\mathrm{d}} \exists^{\mathrm{r}}_{x\in Y}(y = \frac{x}{2})\,\}$$

($\exists^{\mathrm{r}}_{x}$: the real x has no computational significance)
Define $({}^{\mathrm{co}}G, {}^{\mathrm{co}}H) := \nu_{(X,Y)}(\Gamma(X,Y), \Delta(X,Y))$.
Coinduction:

$$(X,Y) \subseteq (\Gamma({}^{\mathrm{co}}G \cup X, {}^{\mathrm{co}}H \cup Y), \Delta({}^{\mathrm{co}}G \cup X, {}^{\mathrm{co}}H \cup Y)) \to (X,Y) \subseteq ({}^{\mathrm{co}}G, {}^{\mathrm{co}}H).$$

Associated to Γ, Δ are algebras $\mathbf{G}, \mathbf{H}$ with constructors

$$\mathrm{LR}\colon \mathbf{PSD} \to \mathbf{G} \to \mathbf{G},$$
$$\mathrm{U}\colon \mathbf{H} \to \mathbf{G} \qquad \text{(for “undefined”)},$$

$$\mathrm{Fin}\colon \mathbf{PSD} \to \mathbf{G} \to \mathbf{H},$$
$$\mathrm{D}\colon \mathbf{H} \to \mathbf{H} \qquad \text{(for “delay”)}.$$

Realizability extensions $({}^{\mathrm{co}}G)^{\mathbf{r}}$ and $({}^{\mathrm{co}}H)^{\mathbf{r}}$:

$$\Gamma^{\mathbf{r}}(Z, W) := \{\, (p, x) \mid \exists_{(p', x') \in Z} \exists_a (x = -a\frac{x' - 1}{2} \land p = \mathrm{LR}_a(p')) \lor \exists_{(q', x') \in W} (x = \frac{x'}{2} \land p = \mathrm{U}(q')) \,\},$$

$$\Delta^{\mathbf{r}}(Z, W) := \{\, (q, x) \mid \exists_{(p', x') \in Z} \exists_a (x = a\frac{x' + 1}{2} \land q = \mathrm{Fin}_a(p')) \lor \exists_{(q', x') \in W} (x = \frac{x'}{2} \land q = \mathrm{D}(q')) \,\}$$

Define

$$(({}^{\mathrm{co}}G)^{\mathbf{r}}, ({}^{\mathrm{co}}H)^{\mathbf{r}}) := \nu_{(Z,W)}(\Gamma^{\mathbf{r}}(Z, W), \Delta^{\mathbf{r}}(Z, W))$$

CoGAverage:

$$\forall^{\mathrm{nc}}_{x,y}({}^{\mathrm{co}}G(x) \to {}^{\mathrm{co}}G(y) \to {}^{\mathrm{co}}G(\frac{x+y}{2})).$$

Consider two sets of averages, the second one with a "carry" $i \in \mathbf{SD}_2 := \{-2, -1, 0, 1, 2\}$:

$$\mathrm{Av} := \{ \frac{x+y}{2} \mid x, y \in {}^{\mathrm{co}}G \},$$
$$\mathrm{Avc} := \{ \frac{x+y+i}{4} \mid x, y \in {}^{\mathrm{co}}G,\ i \in \mathbf{SD}_2 \}.$$

Suffices: Avc satisfies the clause coinductively defining ${}^{\mathrm{co}}G$, for then by the greatest-fixed-point axiom for ${}^{\mathrm{co}}G$ we have $\mathrm{Avc} \subseteq {}^{\mathrm{co}}G$. Since we also have $\mathrm{Av} \subseteq \mathrm{Avc}$ we obtain $\mathrm{Av} \subseteq {}^{\mathrm{co}}G$, i.e., our claim.

CoGAvToAvc:

$$\forall^{\mathrm{nc}}_{x,y \in {}^{\mathrm{co}}G} \exists^{\mathrm{r}}_{x',y' \in {}^{\mathrm{co}}G} \exists^{\mathrm{l}}_{i} \left(\frac{x+y}{2} = \frac{x'+y'+i}{4}\right).$$

Implicit algorithm. $f^* :=$ cCoGPsdTimes, and $s :=$ cCoHToCoG.
(cL denotes the function extracted from the proof of a lemma L.)
CoGPsdTimes: $\forall^{\mathrm{nc}}_{x} \forall_{a} ({}^{\mathrm{co}}G(x) \to {}^{\mathrm{co}}G(a * x))$.

$$\begin{aligned}
f(\mathrm{LR}_a(p), \mathrm{LR}_{a'}(p')) &= (a + a', f^*(-a, p), f^*(-a', p')), \\
f(\mathrm{LR}_a(p), \mathrm{U}(q)) &= (a, f^*(-a, p), s(q)), \\
f(\mathrm{U}(q), \mathrm{LR}_a(p)) &= (a, s(q), f^*(-a, p)), \\
f(\mathrm{U}(q), \mathrm{U}(q')) &= (0, s(q), s(q')).
\end{aligned}$$

Need $J\colon \mathbf{SD} \to \mathbf{SD} \to \mathbf{SD}_2 \to \mathbf{SD}_2$, $K\colon \mathbf{SD} \to \mathbf{SD} \to \mathbf{SD}_2 \to \mathbf{SD}$ with $d + e + 2i = J(d, e, i) + 4K(d, e, i)$ (cases on d, e, i). Then

$$\frac{\frac{x+d}{2} + \frac{y+e}{2} + i}{4} = \frac{\frac{x+y+J(d,e,i)}{4} + K(d, e, i)}{2}.$$

CoGAvcSatCoICl:

$$\forall_i \forall^{\mathrm{nc}}_{x,y \in {}^{\mathrm{co}}G} \exists^{\mathrm{r}}_{x',y' \in {}^{\mathrm{co}}G} \exists^{\mathrm{l}}_{j,d} \Big(\frac{x+y+i}{4} = \frac{\frac{x'+y'+j}{4} + d}{2}\Big).$$

Implicit algorithm.

$$f(i, \mathrm{LR}_a(p), \mathrm{LR}_{a'}(p')) = (J(a, a', i), K(a, a', i), f^*(-a, p), f^*(-a', p')),$$
$$f(i, \mathrm{LR}_a(p), \mathrm{U}(q)) = (J(a, 0, i), K(a, 0, i), f^*(-a, p), s(q)),$$
$$f(i, \mathrm{U}(q), \mathrm{LR}_a(p)) = (J(0, a, i), K(0, a, i), s(q), f^*(-a, p)),$$
$$f(i, \mathrm{U}(q), \mathrm{U}(q')) = (J(0, 0, i), K(0, 0, i), s(q), s(q')).$$

CoGAvcToCoG:

$$\forall^{\mathrm{nc}}_z(\exists^{\mathrm{r}}_{x,y\in{}^{\mathrm{co}}G}\exists^{\mathrm{l}}_i(z=\frac{x+y+i}{4})\to{}^{\mathrm{co}}G(z)),$$
$$\forall^{\mathrm{nc}}_z(\exists^{\mathrm{r}}_{x,y\in{}^{\mathrm{co}}G}\exists^{\mathrm{l}}_i(z=\frac{x+y+i}{4})\to{}^{\mathrm{co}}H(z)).$$

Implicit algorithm. Proof uses SdDisj: $\forall_d(d=0\vee^{\mathrm{r}}\exists^{\mathrm{l}}_a(d=a))$.

$$\begin{aligned}
g(i,p,p') = {} & \mathrm{let}\ (i_1,d,p_1,p_1')=\mathrm{cCoGAvcSatCoICl}(i,p,p')\ \mathrm{in}\\
& \mathrm{case}\ \mathrm{cSdDisj}(d)\ \mathrm{of}\\
& \quad 0\to \mathrm{U}(h(i_1,p_1,p_1'))\\
& \quad a\to \mathrm{LR}_a(g(-ai_1,f^*(-a,p_1),f^*(-a,p_1'))),
\end{aligned}$$

$$\begin{aligned}
h(i,p,p') = {} & \mathrm{let}\ (i_1,d,p_1,p_1')=\mathrm{cCoGAvcSatCoICl}(i,p,p')\ \mathrm{in}\\
& \mathrm{case}\ \mathrm{cSdDisj}(d)\ \mathrm{of}\\
& \quad 0\to \mathrm{D}(h(i_1,p_1,p_1'))\\
& \quad a\to \mathrm{Fin}_a(g(-ai_1,f^*(-a,p_1),f^*(-a,p_1'))).
\end{aligned}$$

Composing CoGAvToAvc and CoGAvcToCoG gives CoGAverage.

Extracted term for CoGAvcToCoG:

```
[ipp](CoRec sdtwo@@ag@@ag=>ag sdtwo@@ag@@ag=>ah)ipp
 ([ipp0][let idpp (cCoGAvcSatCoICl
           left ipp0 left right ipp0 right right ipp0)
      [case (cSdDisj left right idpp)
       (DummyL -> InR(InR(left idpp@right right idpp)))
       (Inr a -> InL(a@InR
        (a times inv left idpp@
         cCoGPsdTimes inv a left right right idpp@
         cCoGPsdTimes inv a right right right idpp)))]])
 ([ipp0][let idpp ...] ...)
```

`ipp`	variable of type $\mathbf{SD}_2 \times \mathbf{G} \times \mathbf{G}$
`idpp`	variable of type $\mathbf{SD}_2 \times \mathbf{SD} \times \mathbf{G} \times \mathbf{G}$
`[ipp]r`	lambda abstraction $\lambda_{ipp} r$
`sdtwo@@ag@@ag=>ah`	function type $\mathbf{SD}_2 \times \mathbf{G} \times \mathbf{G} \to \mathbf{H}$
`r@s`, `left r`, `right r`	product term, components
`cL`	realizer for lemma L

Corecursion $\sim$ coinduction.

$$
\begin{aligned}
{}^{\mathrm{co}}\mathcal{R}_{\mathbf{G}}^{(\mathbf{G},\mathbf{H}),(\sigma,\tau)} &\colon \sigma \to \delta_{\mathbf{G}} \to \delta_{\mathbf{H}} \to \mathbf{G} \\
{}^{\mathrm{co}}\mathcal{R}_{\mathbf{H}}^{(\mathbf{G},\mathbf{H}),(\sigma,\tau)} &\colon \tau \to \delta_{\mathbf{G}} \to \delta_{\mathbf{H}} \to \mathbf{H}
\end{aligned}
$$

with step types

$$
\begin{aligned}
\delta_{\mathbf{G}} &:= \sigma \to \mathbf{PSD} \times (\mathbf{G} + \sigma) + (\mathbf{H} + \tau), \\
\delta_{\mathbf{H}} &:= \tau \to \mathbf{PSD} \times (\mathbf{G} + \sigma) + (\mathbf{H} + \tau).
\end{aligned}
$$

$\mathbf{PSD} \times (\mathbf{G} + \sigma) + (\mathbf{H} + \tau)$ appears since $\mathbf{G}$ has constructors

$$\mathrm{LR}\colon \mathbf{PSD} \to \mathbf{G} \to \mathbf{G} \text{ and } \mathrm{U}\colon \mathbf{H} \to \mathbf{G},$$

and $\mathbf{H}$ has constructors

$$\mathrm{Fin}\colon \mathbf{PSD} \to \mathbf{G} \to \mathbf{H} \text{ and } \mathrm{D}\colon \mathbf{H} \to \mathbf{H}.$$

- Analyzing the step terms gives the "implicit algorithm".
- Extracted terms are in an extension T^+ of Gödel's T, the term language of TCF. They denote partial continuous functionals (Scott/Ershov).
- Verification is automatic (soundness theorem).
- Minlog provides a translation to Haskell for (lazy) evaluation.
- "Code carrying proof" can be a reasonable alternative to "Proof carrying code" (Necula).