Proofs and computations

Helmut Schwichtenberg
(j.w.w. Kenji Miyamoto)

Mathematisches Institut, LMU, München

Leeds University, 7. March 2012

Formalization and extraction

One can extract from a (constructive) proof of a formula with computational content a term that "realizes" (Kleene, Kreisel, Troelstra) the formula. Why should one?

- It can be important to know for sure (and to be able to machine check) that in a proof nothing has been overlooked.
- The same applies to the algorithm implicit in the proof: even if the latter is correct, errors may occur in the implementation of the algorithm.
- Even if the algorithm is correctly implemented, for sensitive applications customers may (and do) require a formal proof that the code implementing the algorithm is correct.

Consequences

- The computational content of a proof should be machine extracted from a formalization of this proof.
- The extract should be a term in the underlying language of the formal system (here: T^+, a common extension of Gödel's T and Plotkin's PCF).
- A soundness theorem should be formally proved: the extract realizes the specification ($:=$ the formula being proved).

Computable functionals

- Types: $\iota \mid \rho \to \sigma$. Ground types ι: free algebras (e.g., **N**).
- Functionals seen as limits of finite approximations: ideals (Kreisel, Scott, Ershov).
- Computable functionals are r.e. sets of finite approximations (example: fixed point functional).
- Functionals are partial. Total functionals are defined (by induction over the types).

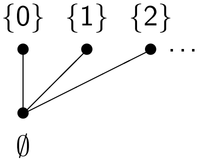
Information systems $\mathbf{C}_\rho$ for partial continuous functionals

- Types ρ, σ, τ: from algebras ι by $\rho \to \sigma$.
- $\mathbf{C}_\rho := (C_\rho, \mathrm{Con}_\rho, \vdash_\rho)$.
- Tokens $a \in C_\rho$ (= atomic pieces of information): constructor trees $\mathrm{C}a_1^*, \dots a_n^*$ with a_i^* a token or $*$. Example: $\mathrm{S}(\mathrm{S}*)$.
- Formal neighborhoods $U \in \mathrm{Con}_\rho$: $\{a_1, \dots, a_n\}$, consistent.
- Entailment $U \vdash_\rho a$.

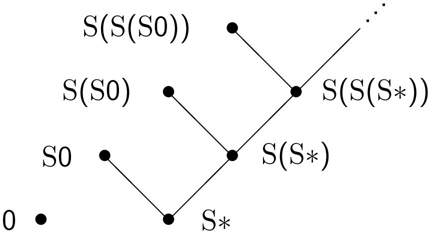
Ideals $x \in |\mathbf{C}_\rho|$ ("points", here: partial continuous functionals): consistent deductively closed sets of tokens.

Flat or non flat algebras?

- Flat:



- Non flat:



Non flat!

- Every constructor C generates an ideal in the function space: $r_{\mathrm{C}} := \{ (U, \mathrm{C}a^*) \mid U \vdash a^* \}$. Associated continuous map:

$$|r_{\mathrm{C}}|(x) = \{ \mathrm{C}a^* \mid \exists_{U \subseteq x}(U \vdash a^*) \}.$$

- Constructors are injective and have disjoint ranges:

$$|r_{\mathrm{C}}|(\vec{x}\,) \subseteq |r_{\mathrm{C}}|(\vec{y}\,) \leftrightarrow \vec{x} \subseteq \vec{y},$$
$$|r_{\mathrm{C}_1}|(\vec{x}\,) \cap |r_{\mathrm{C}_2}|(\vec{y}\,) = \emptyset.$$

- Both properties are false for flat information systems (for them, by monotonicity, constructors need to be strict).

$$|r_{\mathrm{C}}|(\emptyset, y) = \emptyset = |r_{\mathrm{C}}|(x, \emptyset),$$
$$|r_{\mathrm{C}_1}|(\emptyset) = \emptyset = |r_{\mathrm{C}_2}|(\emptyset).$$

A theory of computable functionals, TCF

- A variant of HA^ω.
- Variables range over arbitrary partial continuous functionals.
- Constants for (partial) computable functionals, defined by equations.
- Inductively and coinductively defined predicates. Totality for ground types inductively defined.
- Induction := elimination (or least-fixed-point) axiom for a totality predicate.
- Coinduction := greatest-fixed-point for a coinductively defined predicate.
- Minimal logic: $\to, \forall$ only. $=$ (Leibniz), $\exists$, $\vee$, $\wedge$ (Martin-Löf) inductively defined.
- $\bot := (\mathrm{False} = \mathrm{True})$. Ex-falso-quodlibet: $\bot \to A$ provable.
- Classical logic as a fragment: $\tilde{\exists}_x A$ defined by $\neg\forall_x\neg A$.

Realizability interpretation

- Define a formula $t \mathbf{r} A$, for A a formula and t a term in T^+.
- Soundness theorem:
  If M proves A, then $\mathrm{et}(M) \mathbf{r} A$ can be proved.
- Decorations ($\to^{\mathrm{c}}, \forall^{\mathrm{c}}$ and $\to^{\mathrm{nc}}, \forall^{\mathrm{nc}}$) for removal of abstract data, and fine-tuning:

$$
\begin{aligned}
t \mathbf{r} (A \to^{\mathrm{c}} B) &:= \forall_x (x \mathbf{r} A \to tx \mathbf{r} B), \\
t \mathbf{r} (A \to^{\mathrm{nc}} B) &:= \forall_x (x \mathbf{r} A \to t \mathbf{r} B), \\
t \mathbf{r} (\forall_x^{\mathrm{c}} A) &:= \forall_x (tx \mathbf{r} A), \\
t \mathbf{r} (\forall_x^{\mathrm{nc}} A) &:= \forall_x (t \mathbf{r} A).
\end{aligned}
$$

Example: decorating the existential quantifier

- $\exists_x A$ is inductively defined by the clause

$$\forall_x (A \to \exists_x A)$$

  with least-fixed-point axiom

$$\exists_x A \to \forall_x (A \to P) \to P.$$

- Decoration leads to variants $\exists^{\mathrm{d}}, \exists^{\mathrm{l}}, \exists^{\mathrm{r}}, \exists^{\mathrm{u}}$ (d for "double", l for "left", r for "right" and u for "uniform").

$$\begin{aligned}
&\forall^{\mathrm{c}}_x (A \to^{\mathrm{c}} \exists^{\mathrm{d}}_x A), && \exists^{\mathrm{d}}_x A \to^{\mathrm{c}} \forall^{\mathrm{c}}_x (A \to^{\mathrm{c}} P) \to^{\mathrm{c}} P, \\
&\forall^{\mathrm{c}}_x (A \to^{\mathrm{nc}} \exists^{\mathrm{l}}_x A), && \exists^{\mathrm{l}}_x A \to^{\mathrm{c}} \forall^{\mathrm{c}}_x (A \to^{\mathrm{nc}} P) \to^{\mathrm{c}} P, \\
&\forall^{\mathrm{nc}}_x (A \to^{\mathrm{c}} \exists^{\mathrm{r}}_x A), && \exists^{\mathrm{r}}_x A \to^{\mathrm{c}} \forall^{\mathrm{nc}}_x (A \to^{\mathrm{c}} P) \to^{\mathrm{c}} P, \\
&\forall^{\mathrm{nc}}_x (A \to^{\mathrm{nc}} \exists^{\mathrm{u}}_x A), && \exists^{\mathrm{u}}_x A \to^{\mathrm{nc}} \forall^{\mathrm{nc}}_x (A \to^{\mathrm{nc}} P) \to^{\mathrm{c}} P.
\end{aligned}$$

Practical aspects

- We need formalized proofs, to allow machine extraction.
- Can't take a proof assistant from the shelf: none fits TCF.

Minlog (`http://www.minlog-system.de`)

- Natural deduction for $\to, \forall$, plus inductively and coinductively defined predicates.
- Partial functionals are first class citizens.
- Allows type and predicate parameters (for abstract developments: groups, fields, reals, ...).

Example: average of two reals

Berger and Seisenberger (2009, 2010).

- Extraction from a proof dealing with abstract reals.
- Proof involving coinduction of the proposition that any two reals in $[-1, 1]$ have their average in the same interval.
- B & S informally extract a Haskell program from this proof, which works with stream representations of reals.

Aim here: discuss formalization of the proof, and machine extraction of its computational content.

Free algebra $\mathbf{J}$ of intervals

- $\mathbf{SD} := \{-1, 0, 1\}$ signed digits (or $\{L, M, R\}$).
- $\mathbf{J}$ free algebra of intervals. Constructors

$$\begin{array}{ll} \mathbb{I} & \text{the interval } [-1,1], \\ C\colon \mathbf{SD} \to \mathbf{J} \to \mathbf{J} & \text{left, middle, right half.} \end{array}$$

- $\mathrm{C}_1\mathbb{I}$ denotes $[0, 1]$.
- $\mathrm{C}_0\mathbb{I}$ denotes $[-\frac{1}{2}, \frac{1}{2}]$.
- $\mathrm{C}_0(\mathrm{C}_{-1}\mathbb{I})$ denotes $[-\frac{1}{2}, 0]$.

$\mathrm{C}_{d_0}(\mathrm{C}_{d_1} \dots (\mathrm{C}_{d_{k-1}}\mathbb{I}) \dots)$ denotes the interval in $[-1, 1]$ whose reals have a signed digit representation starting with $d_0 d_1 \dots d_{k-1}$.

- We consider ideals $x \in |\mathbf{C_J}|$.

Total and cototal ideals of base type

Generally:

- Cototal ideals x: every token (i.e., constructor tree) $P(*) \in x$ has a "$\succ_1$-successor" $P(\mathrm{C}\vec{*}) \in x$.
- Total ideals: the cotota ones with $\succ_1$ well-founded.

Examples:

- Total ideals of $\mathbf{J}$:

$$\mathbb{I}_{\frac{i}{2^k},k} := [\frac{i}{2^k} - \frac{1}{2^k}, \frac{i}{2^k} + \frac{1}{2^k}] \qquad \text{for } -2^k < i < 2^k.$$

- Cototal ideals of $\mathbf{J}$: reals in $[-1,1]$, in (non-unique) stream representation using signed digits $-1, 0, 1$.

Inductive and coinductive definitions

- Inductively define a set I of (abstract) reals, by the clauses

$$I0, \qquad \forall^{\mathrm{nc}}_x \forall_d \big(Ix \to I\frac{x+d}{2} \big).$$

  Witnesses are intervals (total ideals in $\mathbf{J}$).

- Coinductively define ${}^{\mathrm{co}}I$, by the (single) clause

$$\forall^{\mathrm{nc}}_x \big({}^{\mathrm{co}}Ix \to x = 0 \vee \exists^{\mathrm{r}}_y \exists_d (x = \frac{y+d}{2} \wedge {}^{\mathrm{co}}Iy) \big).$$

  Witnesses are streams of signed digits (cototal ideals in $\mathbf{J}$).

- From a formalized proof of $\forall^{\mathrm{nc}}_{x,y}({}^{\mathrm{co}}Ix \to {}^{\mathrm{co}}Iy \to {}^{\mathrm{co}}I\frac{x+y}{2})$ extract a stream transformer, of type $\mathbf{J} \to \mathbf{J} \to \mathbf{J}$.

Proof of $\forall^{\mathrm{nc}}_{x,y}({}^{\mathrm{co}}Ix \to {}^{\mathrm{co}}Iy \to {}^{\mathrm{co}}I\frac{x+y}{2})$

$$X := \{\, \frac{x+y}{2} \mid x, y \in {}^{\mathrm{co}}I \,\}, \quad Y := \{\, \frac{x+y+i}{4} \mid x, y \in {}^{\mathrm{co}}I, i \in \mathbf{SD}_2 \,\}.$$

with $\mathbf{SD}_2 := \{-2, -1, 0, 1, 2\}$. Show (i) $X \subseteq Y$ and (ii) that Y satisfies the clause coinductively defining ${}^{\mathrm{co}}I$. Hence $Y \subseteq {}^{\mathrm{co}}I$ (by the greatest-fixed-point for ${}^{\mathrm{co}}I$). Hence $X \subseteq {}^{\mathrm{co}}I$, which is our claim.

XSubY

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

YSatCl

$$\forall_i\forall^{\mathrm{nc}}_{x,y\in{}^{\mathrm{co}}I}\forall^{\mathrm{nc}}_{z}\Bigl(z = \frac{x+y+i}{4} \to z = 0 \lor \exists_{j,d}\exists^{\mathrm{r}}_{x',y'\in{}^{\mathrm{co}}I}\exists^{\mathrm{r}}_{z'}\bigl(z' = \frac{x'+y'+j}{4} \land z = \frac{z'+d}{2}\bigr)\Bigr).$$

Formalization

- Use a type variable ρ to denote an abstract type of reals.
- Need functions P (plus) of type $\rho \to \rho \to \rho$ for addition, and H (half) of type $\rho \to \rho$ for division by 2, with properties

$$
\begin{aligned}
&(x+k)/2 + l = (x + (k +_{\mathbf{Z}} 2l))/2,\\
&(x+k)/4 + l = (x + (k +_{\mathbf{Z}} 4l))/4,\\
&(x+k)/2 + (y+l)/2 = ((x+y) + (k +_{\mathbf{Z}} l))/2,\\
&x + 0 = x, \qquad 0 + y = y,\\
&0/2 = 0, \qquad 2k/2 = k, \qquad k + l = k +_{\mathbf{Z}} l.
\end{aligned}
$$

- In the proof of lemma YSatClause we have to solve $d' + e' + 2i = j + 4d$ for given $d', e' \in \mathbf{SD}$ and $i \in \mathbf{SD}_2$. This is a finite problem and hence can be solved by defining $J \colon \mathbf{SD} \to \mathbf{SD} \to \mathbf{SD}_2 \to \mathbf{SD}_2$ and $D \colon \mathbf{SD} \to \mathbf{SD} \to \mathbf{SD}_2 \to \mathbf{SD}$ explicitly. The validity of $d' + e' + 2i = J(d', e', i) + 4D(d', e', i)$ is proved by cases.

Extraction from lemma XSubY

cXSubY :=

```
[v0,v1]
 [if (des v0)
   [if (des v1)
     (MT@v0@v1)
     ([dv2]JOne M left dv2@v0@right dv2)]
   ([dv2]
    [if (des v1)
      (JOne left dv2 M@right dv2@v1)
      ([dv3]JOne left dv2 left dv3@right dv2@right dv3)])]
```

Here v is a name for variables ranging over **J**, and dv for variables ranging over $\mathbf{SD} \times \mathbf{J}$. The constant des denotes the destructor for **J** of type $\mathbf{J} \to \mathbf{U} + \mathbf{SD} \times \mathbf{J}$, and JOne: $\mathbf{SD} \to \mathbf{SD} \to \mathbf{SD}_2$ adds the two integers.

Extraction from lemma XSubY (continued)

The constant cXSubY of type $\mathbf{J} \to \mathbf{J} \to \mathbf{SD}_2 \times \mathbf{J} \times \mathbf{J}$ is defined to be the term above. It satisfies the equations

$$
\begin{aligned}
\texttt{cXSubY}(\mathbb{I}, \mathbb{I}) &= \langle 0, \mathbb{I}, \mathbb{I} \rangle, \\
\texttt{cXSubY}(\mathbb{I}, \mathrm{C}_e w) &= \langle e, \mathbb{I}, w \rangle, \\
\texttt{cXSubY}(\mathrm{C}_d v, \mathbb{I}) &= \langle d, v, \mathbb{I} \rangle, \\
\texttt{cXSubY}(\mathrm{C}_d v, \mathrm{C}_e w) &= \langle d + e, v, w \rangle.
\end{aligned}
$$

For the given two streams, cXSubY computes the sum of the two head digits (regarding $\mathbb{I}$ as $\mathrm{C}_M \mathbb{I}$), and its tails. This sum of digits of type $\mathbf{SD}_2$ is a "carry" which contains intermediate information to compute the average.

Extraction from lemma YSatClause

cYSatClause :=

```
[i0,v1,v2]
 [if (des v1)
   [if (des v2)
    (J M M i0@D M M i0@v1@v2)
    ([dv3]J M left dv3 i0@D M left dv3 i0@v1@right dv3)]
   ([dv3]
    [if (des v2)
      (J left dv3 M i0@D left dv3 M i0@right dv3@v2)
      ([dv4]J left dv3 left dv4 i0@
            D left dv3 left dv4 i0@
            right dv3@right dv4)])]
```

Extraction from lemma $\mathrm{YSatClause}$ (continued)

The constant `cYSatClause` of type
$\mathbf{SD}_2 \to \mathbf{J} \to \mathbf{J} \to \mathbf{SD}_2 \times \mathbf{SD} \times \mathbf{J} \times \mathbf{J}$ is defined to be the term above. It satisfies the equations

$$
\begin{aligned}
\texttt{cYSatClause}(i, \mathbb{I}, \mathbb{I}) &= \langle J(0,0,i), D(0,0,i), \mathbb{I}, \mathbb{I}\rangle, \\
\texttt{cYSatClause}(i, \mathbb{I}, \mathrm{C}_e w) &= \langle J(0,e,i), D(0,e,i), \mathbb{I}, w\rangle, \\
\texttt{cYSatClause}(i, \mathrm{C}_d v, \mathbb{I}) &= \langle J(d,0,i), D(d,0,i), v, \mathbb{I}\rangle, \\
\texttt{cYSatClause}(i, \mathrm{C}_d v, \mathrm{C}_e w) &= \langle J(d,e,i), D(d,e,i), v, w\rangle.
\end{aligned}
$$

For the given carry and two signed digit streams, `cYSatClause` computes the carry for the next step, the first signed digit of the average of the streams, and the tails of the streams.

Extraction from theorem Average

The term `eterm` extracted from the proof is

```
[v0,v1]
 (CoRec sdtwo@@iv@@iv=>iv)(cXSubY v0 v1)
 ([ivw2]
   Inr
   [let jdvw3
     (cYSatClause left ivw2
                  left right ivw2
                  right right ivw2)
     (left right jdvw3@
     (InR sdtwo@@iv@@iv iv)
     (left jdvw3@right right jdvw3))])
```

of type $\mathbf{J} \to \mathbf{J} \to \mathbf{J}$. It calls `cXSubY` to compute the first carry and the tails of the inputs. Then `CoRec` repeatedly calls `cYSatClause`, to compute the average step by step.

Corecursion

- The conversion rules for $\mathcal{R}$ with total ideals as recursion arguments work from the leaves towards the root, and terminate because total ideals are well-founded.
- For cototal ideals (streams) a similar operator is available to define functions with cototal ideals as values: corecursion.
- ${}^{\mathrm{co}}\mathcal{R}^{\tau}_{\mathbf{J}} \colon \tau \to (\tau \to \mathbf{U} + \mathbf{SD} \times (\mathbf{J} + \tau)) \to \mathbf{J}$ ($\mathbf{U}$ unit type).
- Conversion rule

$$
\begin{aligned}
{}^{\mathrm{co}}\mathcal{R}^{\tau}_{\mathbf{J}} NM \mapsto [&\mathbf{case}\ (MN)^{\mathbf{U}+\mathbf{SD}\times(\mathbf{J}+\tau)}\ \mathbf{of} \\
&\mathrm{inl}\ \_ \mapsto \mathbb{I} \mid \\
&\mathrm{inr}\langle d, z\rangle \mapsto \mathrm{C}_d[\mathbf{case}\ z^{\mathbf{J}+\tau}\ \mathbf{of} \\
&\qquad\qquad\qquad \mathrm{inl}\ \_ \mapsto \mathbb{I} \mid \\
&\qquad\qquad\qquad \mathrm{inr}\ u^{\tau} \mapsto {}^{\mathrm{co}}\mathcal{R}^{\tau}_{\mathbf{J}} uM]].
\end{aligned}
$$

An experiment

- Apply eterm to $1/2 + 1/8 = 5/8$ and $1/2 + 1/4 = 3/4$.
- Type the commands

```
(define test (nt (mk-term-in-app-form
  eterm (pt "C R(C M(C R II))") (pt "C R(C R II)"))))
(define neterm10 (nt (undelay-delayed-corec test 10)))
(pp neterm10)
```

- The result is

```
C R (C R (C M (C L (C M (C M (C M (C M (C M (C M
          ((CoRec sdtwo@@iv@@iv=>iv) ...))))))))))
```

- The result is correct, as
  $(5/8 + 3/4)/2 = 11/16 = 1/2 + 1/4 - 1/16$.

Conclusion

- Both $\forall^{\mathrm{c}}$ and $\forall^{\mathrm{nc}}$. Similarly: both $\to^{\mathrm{c}}$ and $\to^{\mathrm{nc}}$.
- Inductively defined predicates, in particular $=$, $\exists$, $\vee$.
- Computational variants $\exists^{\mathrm{d}}, \exists^{\mathrm{l}}, \exists^{\mathrm{r}}, \exists^{\mathrm{u}}, \dots$.
- Coinductively defined predicates.
- Recursion and corecursion operators $\mathcal{R}_\iota^\tau$, ${}^{\mathrm{co}}\mathcal{R}_\iota^\tau$.

By the soundness theorem one can

Extract stream transformers from proofs on abstract reals.

References

- U. Berger, From coinductive proofs to exact real arithmetic. CSL 2009.
- U. Berger, K. Miyamoto, H.S. and M. Seisenberger, The interactive proof system Minlog. Calco-Tools 2011.
- U. Berger and M. Seisenberger, Proofs, programs, processes. CiE 2010.
- H.S., Realizability interpretation of proofs in constructive analysis. Theory of Computing Systems, 2008.
- H.S. and S.S. Wainer, Proofs and Computations. Perspectives in Logic, ASL & Cambridge UP, 2012.