Proofs and Programs

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

Mathematisches Institut der Universität München

Chalmers University of Technology, Göteborg, Sweden
20. January 2011

- Proofs may have computational content.
- Programs extracted from proofs cannot go wrong.
- Proofs (as opposed to programs) can easily be checked for correctness.

Issues:

- Attention to data necessary.
- Complexity.

Data

- Free algebras (natural numbers, lists, ...).
- Functions (seen as limits of finite approximations).
- Enumerated sets (as opposed to sets given by a property).

More precisely: use the Scott-Ershov partial continuous functionals, as the intended model of a type theory based on free algebras.

- A (higher type) functional is computable if it is the limit of a recursively enumerable set of finite approximations.

Language

- We teach that existence and disjunction are abbreviations:

$$\tilde{\exists}_x A := \neg \forall_x \neg A,$$
$$A \mathrel{\tilde{\vee}} B := \neg(\neg A \wedge \neg B)$$

  and often forget to mention their proper versions $\exists_x A$, $A \vee B$.
- To fine tune the computational content of a proof, distinguish $\to^{\mathrm{c}}$, $\forall^{\mathrm{c}}$ (computational) and $\to^{\mathrm{nc}}$, $\forall^{\mathrm{nc}}$ (non-computational).

Example: Variants of $\vee$, inductively defined by the clauses

$$\begin{cases} A \to^{\mathrm{c}} A \vee^{\mathrm{d}} B \\ B \to^{\mathrm{c}} A \vee^{\mathrm{d}} B \end{cases} \qquad \begin{cases} A \to^{\mathrm{nc}} A \vee^{\mathrm{u}} B \\ B \to^{\mathrm{nc}} A \vee^{\mathrm{u}} B \end{cases} \qquad \begin{cases} A \to^{\mathrm{c}} A \vee^{\mathrm{l}} B \\ B \to^{\mathrm{nc}} A \vee^{\mathrm{l}} B \end{cases}$$

and similar for $\vee^{\mathrm{r}}$.

Formulas as computational problems

- Kolmogorov (1925) 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: $\forall^{\mathrm{c}}_{n} \exists_{m>n} \mathrm{Prime}(m)$ 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.).

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

Goal: Insert as few as possible "decorations" $\forall^{\mathrm{c}}, \to^{\mathrm{c}}$ into a proof.

- $\mathrm{Seq}(M)$ of a proof M consists of its context and end formula.
- The uniform proof pattern $\mathrm{P}(M)$ of a proof M is the result of changing in c.r. formulas of M (i.e., not above a n.c. formula) all $\to^{\mathrm{c}}$, $\forall^{\mathrm{c}}$ into $\to^{\mathrm{nc}}$, $\forall^{\mathrm{nc}}$ (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 $\to^{\mathrm{c}}$, $\forall^{\mathrm{c}}$.
- 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 c.r. formula in M is extended by the corresponding one in N.

Decoration algorithm

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.

Theorem (Ratiu, S.)

Under the assumption above, for every uniform 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

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

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 $\Pi_1,\Delta_1\Rightarrow C_1\to D_1$ extends $\Pi,\Delta\Rightarrow A\to^{\mathrm{nc}}D$ $(\to\in\{\to^{\mathrm{nc}},\to^{\mathrm{c}}\})$. 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 $\to^{\mathrm{c}}$, $\forall^{\mathrm{c}}$ 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 $\Pi_1,\Delta_2\Rightarrow C_2\to D_1\mapsto$ decoration M_3 of U whose sequent $\Pi_3,\Delta_3\Rightarrow C_3\to D_3$ extends $\Pi_1,\Delta_2\Rightarrow C_2\to D_1$.
- $\mathrm{IH}_a(V)$ 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 $\Delta_3,\Sigma_2\Rightarrow C_3$. ...

Example: Euler's φ, or avoiding factorization

Let Pn mean "n is prime". Consider

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

Euler's φ has the properties

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

Using factorization and these properties we obtain a proof of

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

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

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

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

- The decoration algorithm arrives at `Fact` with goal

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

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

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

$$\forall^{\mathrm{c}}_{n} \exists_{i \le k \le n} \forall_{i' \le k' \le n} (\mathrm{seg}(i', k') \preceq \mathrm{seg}(i, k)).$$

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

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

Example: MSS (ctd.)

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

$$\forall^{\mathrm{c}}_n(\exists_{i \le k \le n} \forall_{i' \le k' \le n}(\mathrm{seg}(i', k') \preceq \mathrm{seg}(i, k)) \wedge \\ \exists_{j \le n} \forall_{j' \le n}(\mathrm{seg}(j', n) \preceq \mathrm{seg}(j, n)))$$

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 $n+1$: $\forall^{\mathrm{c}}_n \exists_{j \le n+1} \forall_{j' \le n+1} (\mathrm{seg}(j', n+1) \preceq \mathrm{seg}(j, n+1))$.

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

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

- 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 $\mathtt{L}$:

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

and $\mathtt{LMon}$:

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

- The decoration algorithm arrives at $\mathtt{L}$ with goal

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

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

Result of demo

Extracted term for L

```
[le0,seg1,n2,n3]
 (Rec nat=>nat)n3 0
 ([n4,n5][if (le0(seg1 n5(Succ n2))(seg1(Succ n4)(Succ n2))
             (Succ n4)
             n5])
```

Extracted term for LMon

```
[le0,seg1,n2,n3]
 [if (le0(seg1 n3(Succ n2))(seg1(Succ n2)(Succ n2)))
     (Succ n2)
     n3]
```

Result of demo (ctd.)

Extracted term for `MaxSegMon`

```
[le0,seg1,n2]
 (Rec nat=>nat@@nat@@nat)n2(0@0@0)
 ([n3,ijk4]
   [if (le0(seg1 left ijk4 right right ijk4)
          (seg1((cL alpha)le0 seg1 n3(Succ n3))(Succ n3)))
     ((cL alpha)le0 seg1 n3(Succ n3))
     (left ijk4)]@
   (cL alpha)le0 seg1 n3(Succ n3)@
   [if (le0(seg1 left ijk4 right right ijk4)
          (seg1((cL alpha)le0 seg1 n3(Succ n3))(Succ n3)))
     (Succ n3)
     (right right ijk4)])
```

After decoration `cL` is replaced by `cLMon` $\Rightarrow$ linear algorithm.

References

- U. Berger, Uniform Heyting arithmetic. APAL 133 (2005).
- D. Ratiu and H.S., Decorating proofs. Mints volume (S. Feferman and W. Sieg, eds.). 2010.
- H.S. and S.S. Wainer, Proofs and Computations. Perspectives in Mathematical Logic, Cambridge UP. To appear, 2011.