Program development by proof transformation

Helmut Schwichtenberg

joint work with Luca Chiarabini and Diana Ratiu
Mathematisches Institut, LMU, München

AIST, Osaka, 11. June 2009

Logic

- The only (basic) logical connectives are $\to$, $\forall$.
- Proofs have two aspects:
  (i) They guarantee correctness.
  (ii) They may have computational content.
- Computational content only enters a proof via inductively (or coinductively) defined predicates.
- To fine tune the computational content of a proof, distinguish $\to^{\mathrm{c}}$, $\forall^{\mathrm{c}}$ (computational) and $\to$, $\forall$ (non-computational).

Natural deduction: assumption variables u^A. Rules for $\to^{\mathrm{c}}$:

derivation	proof term
$\dfrac{\begin{array}{c}[u\colon A]\\ \vert\ M\\ B\end{array}}{A \to^{\mathrm{c}} B}\,(\to^{\mathrm{c}})^+\,u$	$(\lambda_{u^A} M^B)^{A\to^{\mathrm{c}} B}$
$\dfrac{\begin{array}{c}\vert\ M\\ A \to^{\mathrm{c}} B\end{array}\qquad\begin{array}{c}\vert\ N\\ A\end{array}}{B}\,(\to^{\mathrm{c}})^-$	$(M^{A\to^{\mathrm{c}} B} N^A)^B$

Natural deduction: rules for $\forall^{\mathrm{c}}$

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

Restrictions to $\to^+$ and $\forall^+$ (non-computational)

$\mathrm{CV}(M) :=$ the set of "computational variables" of a derivation M, relative to a fixed assigment $u^A \mapsto x_u^{\tau(A)}$. Consider

$$\dfrac{\begin{array}{c}[u\colon A]\\ \vert\ M\\ B\end{array}}{A \to B}\to^+ u \qquad \text{or as proof term} \quad (\lambda_{u^A} M^B)^{A\to B}.$$

$(\lambda_{u^A} M^B)^{A\to B}$ is correct if M^B is and $x_u \notin \mathrm{CV}(M^B)$. Consider

$$\dfrac{\begin{array}{c}\vert\ M\\ A\end{array}}{\forall_x A}\forall^+ x \qquad \text{or as proof term} \quad (\lambda_x M^A)^{\forall_x A} \qquad \text{(with var. condition)}.$$

$(\lambda_x M^A)^{\forall_x A}$ is correct if M^A is and $x \notin \mathrm{CV}(M^A)$.

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: $\forall^{\mathrm{c}}_n \exists^{\mathrm{l}}_{m>n} \mathrm{Prime}(m)$ has type $\mathbf{N} \to \mathbf{N}$.
- $A \mapsto \tau(A)$, a type or the "nulltype" symbol ε.
- In case $\tau(A) = \varepsilon$ proofs of A have no computational content; such formulas A are called computationally irrelevant (c.i.) or Harrop formulas; the others computationally relevant (c.r.).

Realizability

Let t be either a term of type $\tau(A)$ if this is a type, or ε if $\tau(A) = \varepsilon$. Extend term application to the "nullterm" symbol ε:

$$\varepsilon t := \varepsilon, \quad t\varepsilon := t, \quad \varepsilon\varepsilon := \varepsilon.$$

We define the formula $t\ \mathbf{r}\ A$, read t realizes A.

$$\begin{aligned}
&\varepsilon\ \mathbf{r}\ I\vec{r} := I\vec{r} \quad \text{for } I \text{ not requiring witnesses (e.g., } \mathrm{Eq}),\\
&t\ \mathbf{r}\ (A \to^{\mathrm{c}} B) := \forall_x(x\ \mathbf{r}\ A \to tx\ \mathbf{r}\ B),\\
&t\ \mathbf{r}\ (A \to B) := \forall_x(x\ \mathbf{r}\ A \to t\ \mathbf{r}\ B),\\
&t\ \mathbf{r}\ \forall^{\mathrm{c}}_x A := \forall_x(tx\ \mathbf{r}\ A), \quad t\ \mathbf{r}\ \forall_x A := \forall_x(t\ \mathbf{r}\ A)
\end{aligned}$$

and similarly for $\exists$, $\wedge$, $\vee$ and other inductively defined I's.

Derivations and extracted terms

For M^A with A c.i. let $[\![M]\!] := \varepsilon$. Assume A is c.r. Then

$$
\begin{aligned}
[\![u^A]\!] &:= x_u^{\tau(A)} \quad (x_u^{\tau(A)} \text{ uniquely associated with } u^A),\\
[\![(\lambda_{u^A} M^B)^{A\to^{\mathrm{c}} B}]\!] &:= \lambda_{x_u^{\tau(A)}} [\![M]\!],\\
[\![(M^{A\to^{\mathrm{c}} B} N^A)^B]\!] &:= [\![M]\!][\![N]\!],\\
[\![(\lambda_{x^\rho} M^A)^{\forall^{\mathrm{c}}_x A}]\!] &:= \lambda_{x^\rho} [\![M]\!],\\
[\![(M^{\forall^{\mathrm{c}}_x A(x)} r)^{A(r)}]\!] &:= [\![M]\!] r,\\
[\![(\lambda_{u^A} M^B)^{A\to B}]\!] &:= [\![(M^{A\to B} N^A)^B]\!] := [\![(\lambda_{x^\rho} M^A)^{\forall_x A}]\!]\\
&:= [\![(M^{\forall_x A(x)} r)^{A(r)}]\!] := [\![M]\!].
\end{aligned}
$$

Define $\mathrm{CV}(M) := \mathrm{FV}([\![M]\!])$.

Soundness

Let M be a derivation of A from assumptions $u_i : C_i$ $(i < n)$. Then we can find a derivation of $[\![M]\!]\ \mathbf{r}\ A$ from assumptions

$$\begin{cases} x_{u_i}\ \mathbf{r}\ C_i & \text{for } \tau(C_i) \neq \varepsilon \text{ and } x_{u_i} \in \mathrm{CV}(M) \\ \exists_x(x\ \mathbf{r}\ C_i) & \text{for } \tau(C_i) \neq \varepsilon \text{ and } x_{u_i} \notin \mathrm{CV}(M) \\ \varepsilon\ \mathbf{r}\ C_i & \text{for } \tau(C_i) = \varepsilon. \end{cases}$$

Bin packing (Goad 1980)

- $X = i_0, i_1, \ldots, i_{p-1}$ list of blocks.
- $B = j_0, j_1, \ldots, j_{q-1}$ list of bins.
- $A = k_0, k_1, \ldots, k_{p-1}$ assigns to an index of a block the index of the bin it should go into.
- $\mathrm{Legal}(A, X, B)$ defined by

$$\mathrm{Legal}(kA, iX, B) := \mathrm{Legal}(A, X, \mathrm{Decr}(B, k, i)).$$

Bin packing

- Specification:

$$\forall^{\mathrm{c}}_{X,B}\exists^{\mathrm{d}}_{p}((p \to \exists^{\mathrm{l}}_{A}\mathrm{Legal}(A,X,B)) \land ((p \to \mathbf{F}) \to \exists^{\mathrm{l}}_{A}\mathrm{Legal}(A,X,B) \to \mathbf{F})$$

- Specializations of the proof contain many case distinctions

$$\dfrac{A \lor B \qquad \begin{array}{c}[u\colon A]\\ \mid M\\ C\end{array} \qquad \begin{array}{c}[v\colon B]\\ \mid N\\ C\end{array}}{C}$$

- Remove predecided case distinctions: replace $v\colon B$ in N by a proof of B from the present context. Result: N'.
- Simplification (Prawitz): replace the whole case dist. by N'.

Bin packing: extracted terms for $X = [i_1, i_2]$, $B = [j, j]$

```
[if (i1<=j)
  [if (i2<=j--i1)
    (True@0::0:)
    [if (i2<=j) (True@0::1:) (False@(Nil nat))]]
  (False@(Nil nat))]
```

With `i2<=j` as premise

```
[if (i1<=j) (True@0::1:) (False@(Nil nat))]
```

- Extensionally different program: it never returns `0::0:`
- Program transformation cannot do this.

Decoration can simplify extracts

- Suppose that a proof M uses a lemma $L^{\mathrm{d}}\colon A \vee^{\mathrm{d}} B$.
- Then the extract $[\![M]\!]$ will contain the extract $[\![L^{\mathrm{d}}]\!]$.
- 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 $L\colon A \vee B$.
- Since $[\![L]\!]$ is a boolean, the extract of the modified proof is "purified": the (possibly large) extract $[\![L^{\mathrm{d}}]\!]$ has disappeared.

Decorating proofs

Goal: Insert as few as possible decorations into a proof.

- $\mathrm{Seq}(M)$ of a proof M consists of its context and end formula.
- The uniform proof pattern $U(M)$ of a proof M is the result of changing in c.r. formulas of M (i.e., not above a c.i. formula) all $\to^{\mathrm{c}}$, $\forall^{\mathrm{c}}$ into $\to$, $\forall$, except "uninstantiated" formulas of axioms, e.g., $\forall^{\mathrm{c}}_n(Q0 \to^{\mathrm{c}} \forall^{\mathrm{c}}_n(Qn \to^{\mathrm{c}} Q(\mathrm{S}n)) \to^{\mathrm{c}} Qn)$.
- A formula D extends C if D is obtained from C by changing some $\to$, $\forall$ 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 of M is extended by the corresponding one in N.

Decoration algorithm

Assumption: We have an algorithm assigning to every axiom A and every decoration variant C of A another axiom whose formula D extends C, and D is the least among those extensions.

Theorem (Ratiu, H.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_∞.*

Passing continuations

- The idea of continuation passing style programming can be expressed by a formula

$$\forall^{\mathrm{c}}_n(Qn \to^{\mathrm{c}} Q(\mathrm{S}n)) \to^{\mathrm{c}}$$
$$\forall^{\mathrm{c}}_{n,m}((Qn \to^{\mathrm{c}} Q(n+m)) \to^{\mathrm{c}} Q0 \to^{\mathrm{c}} Q(n+m)).$$

- We prove induction $\forall^{\mathrm{c}}_n(Qn \to^{\mathrm{c}} Q(\mathrm{S}n)) \to^{\mathrm{c}} \forall^{\mathrm{c}}_n(Q0 \to^{\mathrm{c}} Qn)$ in continuation passing style, i.e., not directly, but using the formula above as an intermediate assertion.

Result of demo: extracted term E

```
[f0,n1]
 (Rec nat=>nat=>(alpha=>alpha)=>alpha=>alpha)n1([n3,k4]k4)
 ([n3,p4,n5,k6]p4(Succ n5)([x8]k6(f0 n3 x8)))
```

applied to `0` and `([x3]x3)`.
E has value type $\mathbf{N} \to (\alpha \to \alpha) \to \alpha \to \alpha$.

$$E(f, 0, m, k) = k,$$
$$E(f, n+1, m, k) = E(f, n, m+1, k \circ f(n)).$$

This is almost continuation passing style: m is unwanted.

Extracted term D after decoration

```
[f0,n1]
 (Rec nat=>(alpha=>alpha)=>alpha=>alpha)n1([k3]k3)
 ([n3,p4,k5]p4([x7]k5(f0 n3 x7)))
```

applied to `([x3]x3)`. D has value type $(\alpha \to \alpha) \to \alpha \to \alpha$.

$$D(f, 0, k) = k,$$
$$D(f, n+1, k) = D(f, n, k \circ f(n)).$$

This is continuation passing style: f, n are mapped to $k \mapsto k \circ f(n-1) \circ \ldots \circ f(0)$.

Example: Maximal Scoring Segment (MSS)

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

$$\forall^{\mathrm{c}}_n \exists^{\mathrm{l}}_{i \leq k \leq n} \forall_{i' \leq k' \leq n} (\mathrm{seg}(i', k') \leq \mathrm{seg}(i, k)).$$

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

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

- Special case: maximal end segment

$$\forall^{\mathrm{c}}_n \exists^{\mathrm{l}}_{j \leq n} \forall_{j' \leq n} (\mathrm{seg}(j', n) \leq \mathrm{seg}(j, n)).$$

Example: MSS (ctd.)

Two proofs of the existence of a maximal end segment for $n+1$

$$\forall_n^{\mathrm{c}} \exists_{j \le n+1}^{\mathrm{l}} \forall_{j' \le n+1} (\mathrm{seg}(j', n+1) \le \mathrm{seg}(j, n+1)).$$

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

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

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

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

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

Example: MSS (ctd.)

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

$$\forall_n^{\mathrm{c}}(\exists^{\mathrm{l}}_{i\le k\le n}\forall_{i'\le k'\le n}(\mathrm{seg}(i',k')\le \mathrm{seg}(i,k))\wedge^{\mathrm{d}}$$
$$\exists^{\mathrm{l}}_{j\le n}\forall_{j'\le n}(\mathrm{seg}(j',n)\le \mathrm{seg}(j,n)))$$

In the step:

- Compare the maximal segment i,k for n with the maximal end segment $j,n+1$ proved separately.
- If $\le$, 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.)

Could the better proof be found automatically? Have L1 and L2:

$$\forall_n^{\mathrm{c}}\forall_{m\le n+1}^{\mathrm{c}}\exists_{j\le n+1}^{\mathrm{l}}\forall_{j'\le m}(\mathrm{seg}(j',n{+}1)\le\mathrm{seg}(j,n{+}1)),$$
$$\mathtt{Mon}\to\forall_n^{\mathrm{c}}(\mathtt{ES}_n\to^{\mathrm{c}}\forall_{m\le n+1}\exists_{j\le n+1}^{\mathrm{l}}\forall_{j'\le m}(\mathrm{seg}(j',n{+}1)\le\mathrm{seg}(j,n{+}1))).$$

- The decoration algorithm arrives at L1 with

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

- L2 fits as well, its assumptions Mon and $\mathtt{ES}_n$ are in the context, and it is the less extended ($\forall_{m\le n+1}$ rather than $\forall_{m\le n+1}^{\mathrm{c}}$), hence is preferred.

Further work, problems, outlook

- Luca Chiarabini applied pruning successfully in bioinformatics: string alignment, bounded perfect matching.
- Efficiency problem: proofs must be well structured.
- More experience needed.
- Code carrying proofs for high security requirements.

References

- U. Berger, Uniform Heyting arithmetic. APAL 133 (2005).
- D. Ratiu and H.S., Decorating proofs. To appear, Mints volume (S. Feferman and W. Sieg, eds.), 2009.