Interactive Theorem Proving

master course

13.01.2017

"Program Extraction" by Kenji Miyamoto

(Kenji.Miyamoto@uibk.ac.at)

Take some formula A as a problem.

We are going to think about how to find a program which is a computational solution of A.

Our goal is...

for a given formula A,

finding a program t and a proof that t is a computational solution of A.

Notion of "computational solution"

In the BHK-interpretation,

- Atomic formula case is opened.
- The notion of "construction" is undefined.

We define our atomic formula to be a formula built by a predicate variable or a (co)inductive predicate.

Then we extend Kreisel's modified realizability to formulate "... is a computational solution of ..."

Program Extraction

Program extraction is a method of obtaining from a proof of A a program t which is provably a realizer of a formula A.

We will study the Theory of Computable Functionals* (TCF) which is suitable for program extraction.

Features of TCF

- Gödel's T with extensions
- everything is partial
- First-order minimal logic with (co) inductive definitions
- Computational / non-computational distinction (e.g. $\forall / \forall^{nc}$)
- Concretely described and is ready to implement.

* Schwichtenberg & Wainer, Proofs and Computation, CUP 2012.

Proof Assistant Minlog

- Minlog is an implementation of TCF.
- Advanced proof assistant for program extr.

Informal case studies

1. Even / odd.

Proposition 1

Any natural number is even or odd.

From a proof of the proposition, we extract a program deciding that the input natural is either even or odd. Moreover, there is a proof that the extracted program is a realizer of the proposition.

Informal case studies

2. Approximate split property of reals.

Prop 2 Assume x, y are reals s.t. $x < y$.

For any real z, $z < y$ or $x < z$.

Extracted program A decision procedure telling us

either $z < y$ or $x < z$.

Informal case studies

3. Cauchy real into signed digit stream (SDS)

Prop 3 Any Cauchy real $x \in [-1.1]$ is approximable.

Here, "approximability of x" is coinductively defined to mean that there is approximable $y \in [-1,1]$ s.t. $x = \frac{y+d}{2}$ where $d \in \{-1, 0, 1\}$.

Extracted program input : a Cauchy real in $[-1,1]$.

output : an SDS representing the input.

SDSs as reals in $[-1,1]$

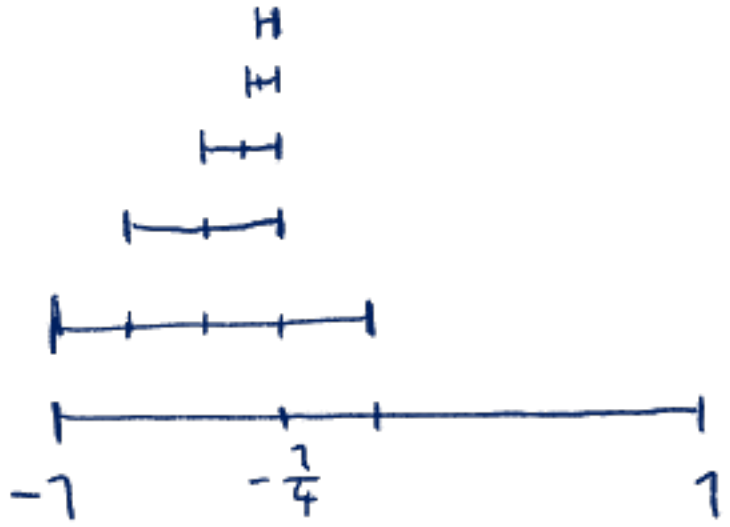
Signed digits (SD) are $\{-1, 0, 1\}$.

- Intuitively they mean left, middle, right, resp.

Signed digit streams are infinite lists of SD.

An SDS represents a real in $[-1,1]$

Example $-1 : 0 : 1 : 1 : 1 : \cdots$



Today's Goal

Understanding those case studies in TCF, and practicing program extraction within the proof assistant Minlog.

... we omit the following items

- General treatment of the theory.
- Semantics of term calculus (Scott-Yashov model)
- extraction from classical logic

Assume we have

- Gödel's T
- first-order minimal logic with $\to, \forall$.
  - we prefer natural deduction
- inductive definitions

The statement of Proposition 7 in TCF can be

$$\forall_{n^{\mathbb{N}}}(\mathrm{Nat}(n) \to \mathrm{Even}(n) \vee \mathrm{Odd}(n))$$

$\mathbb{N}$ is the type of naturals,

Nat, Even, Odd, $\vee$ are inductively defined predicates.

Nat comes with the introduction axioms Nat_0^+, Nat_1^+

$$\mathrm{Nat}(0) \quad , \quad \forall_n(\mathrm{Nat}(n) \to \mathrm{Nat}(Sn))$$

and the elimination axiom Nat^- where P is a predicate variable.

$$\forall_n(\mathrm{Nat}(n) \to P(0) \to \forall_m(\mathrm{Nat}(m) \to P(m) \to P(Sm)) \to P(n))$$

Assume A, B, P are nullary predicate variables.

$\vee$ comes with

$$A \to A \vee B \quad (\vee_0^+), \qquad B \to A \vee B \quad (\vee_1^+).$$

$$A \vee B \to (A \to P) \to (B \to P) \to P \quad (\vee^-)$$

We call predicate variables used to be a place-holder in elimination axioms <u>competitor predicates</u>.

Realizability relation

We define the binary relation $\Vdash$ (bold r) on a term and a formula. We read it "realizes" or "... is a realizer of ...".

$$t \Vdash P\vec{s} \quad := P^* t \vec{s} \qquad (P^* \text{ via global assignment})$$

$$t \Vdash A \to B \quad := \forall x (x \Vdash A \to tx \Vdash B)$$

$$t \Vdash \forall_x A \quad := \forall x (tx \Vdash A)$$

For an inductive predicate I,

$$t \Vdash I\vec{s} \quad := I^* t \vec{s},$$

where I^* is another inductive predicate suitable for $\Vdash$.

Generation Trees

We settle the atomic formula case of the BHK.

NB. a proof of $I(n)$ is a tree consisting of $I^+_0, I^+_1, \cdots, I^+_{k-1}$.

Example $Even(0)$ by $Even^+_0$.

$Even(2)$ by $Even^+_0$ and $Even^+_1$.

$Even(4)$ by $Even^+_0$, $Even^+_1$ and $Even^+_1$.

We call such proofs "generation trees", and prepare realizers by means of a new algebra (datatype) "C_I" yielded from $I^+_0, \cdots, I^+_{k-1}$.

Special case for Even.

$\mathrm{Even}(0)$ (Even^{+}_{0}), $\forall_n(\mathrm{Even}(n) \to \mathrm{Even}(S(Sn)))$ (Even^{+}_{1})

We can use L_{Even} defined by 2 constructors,

$$Z : L_{\mathrm{Even}} \qquad C : \mathbb{N} \to L_{\mathrm{Even}} \to L_{\mathrm{Even}}$$

$Z \Vdash \mathrm{Even}(0)$, $C\,0\,Z \Vdash \mathrm{Even}(2)$, $C\,2\,(C\,0\,Z) \Vdash \mathrm{Even}(4)$.

Therefore, we get $\mathrm{Even}^{\Vdash}$ with the following axioms.

$\mathrm{Even}^{\Vdash} Z\,0$ $(\mathrm{Even}^{\Vdash})^{+}_{0}$,

$\forall_{n,x}(\mathrm{Even}^{\Vdash} x\,n \to \mathrm{Even}^{\Vdash}(C\,n\,x\,, S(Sn)))$ $(\mathrm{Even}^{\Vdash})^{+}_{1}$, and $(\mathrm{Even}^{\Vdash})^{-}$.

<u>Question</u> Is there a simpler data type for L_{Even}?

Non-computational $\forall$

We introduce $\forall^{nc}$ in order

1. to have simpler realizers,
2. to extend "$\Vdash$" in a consistent way.

$\forall_x A$ can be replaced by $\forall^{nc}_x A$ if we find a realizer of A without using x.

We extend the realizability by adding

$$t \Vdash \forall^{nc}_x A \;:=\; \forall^{nc}_x (t \Vdash A)$$

We update Even^+_1 to be $\forall^{nc}_n(\mathrm{Even}(n) \to \mathrm{Even}(S(Sn)))$
and C_{Even} to be $\mathbb{N}$.

Decorating $\to$ and predicates

In the same spirit, we introduce $\to^{nc}$.

$$t \mathbin{\#} A \to^{nc} B \quad := \quad \forall_x (x \mathbin{\#} A \to t \mathbin{\#} B)$$

We also allow non-computational predicate variables and non-computational inductively defined predicates.

Especially it is beneficial to define $I^{\#}$ as n.c. ind. pred.

- Nothing new to obtain by having $I^{\#}$ computational,
- We can still safely think about verification proofs.

We don't think about a realizer of atomic formulas of n.c. pred.

L_{Even} and $Even^*$

Even comes with $\begin{cases} Even(0) & (Even_0^\dagger) \\ \forall_n^{nc}(Even(n) \to Even(S(Sn))) & (Even_1^\dagger) \end{cases}$

For each axiom formula we obtain a signature L_{Even} and $L_{Even} \to L_{Even}$ for defining the datatype L_{Even}.

Using the constructors of $L_{Even} \cong \mathbb{N}$, we define $Even^*$ so that we have the corresponding axioms

$$0 * Even(0) = Even^* 0\,0 \qquad (Even^*)_0^\dagger$$

$$S * \forall_n^{nc}(Even(n) \to Even(S(Sn))) = \forall_{n,m}(Even^* m\, n \to Even^*(Sm)(S(Sn))) \qquad (Even^*)_1^\dagger$$

Program Extraction

Let M be a proof of $\forall_n^{nc}(Nat(n) \to Even(n) \vee Odd(n))$.

We use the mapping et to find a realizer of the proven formula s.t. there is a proof M' of the formula

$et(M) \Vdash \forall_n^{nc}(Nat(n) \to Even(n) \vee Odd(n))$.

Now we define

1. $\tau(A)$ which gives the type of realizers of the formula A.
2. $et(M)$ as above, extracted term from the proof.

If the final conclusion of A is formed by a n.c. predicate,

$\tau(A) := o$ ... null type. s.t. $o \to \tau := \tau$, $\tau \to o := o \to o := o$

O.W. we have a type.

$\tau(A \to B) := \tau(A) \to \tau(B)$, $\tau(A \to^{nc} B) := \tau(B)$,

$\tau(\forall_x^{\rho} A) := \rho \to \tau(A)$, $\tau(\forall_x^{nc\,\rho} A) := \tau(A)$,

$\tau(P\vec{s}) := \tau_P$ assuming a global assumption.

$\tau(I\vec{s}) := \iota_I$

Our notion of proofs is Gentzen's natural deduction extended by $(\to^{nc})^+$, $(\to^{nc})^-$, $(\forall^{nc})^+$, $(\forall^{nc})^-$, I_i^+, I_i^-.

Via the Curry-Howard isomorphism, we work with proofs in the term representation.

$et(u) := x_u$

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

$et\left(\left(\lambda u^{A} M^{B}\right)^{A\to B}\right) := \lambda x_u\, et(M)$, $\quad et\left(\left(\lambda u^{A} M^{B}\right)^{A\to^{nc} B}\right) := et(M)$.

$et\left(\left(M^{\forall_x A(x)}\, t\right)^{A(t)}\right) := et(M)\, t$, $\quad et\left(\left(M^{\forall_x^{nc} A(x)}\, t\right)^{A(t)}\right) := et(M)$,

$et\left(\left(\lambda_x{}^{\rho} M^{A}\right)^{\forall_x A}\right) := \lambda x\, et(M)$, $\quad et\left(\left(\lambda_x{}^{\rho} M^{A}\right)^{\forall_x^{nc} A}\right) := et(M)$,

$et\left(I_i^{\tau}\right) := C_i$ which is the i-th constructor of L_I,

$et\left(I^{-}\right) := R^{\sigma}_{L_I}$ where σ is the type of realizer of the competitor.

Example ($R_{\mathbb{N}}^{\sigma}$)

$$R_{\mathbb{N}}^{\sigma} : \mathbb{N} \to \sigma \to (\mathbb{N} \to \sigma \to \sigma) \to \sigma$$

$$R_{\mathbb{N}}^{\sigma}\, 0\, M\, N = M$$

$$R_{\mathbb{N}}^{\sigma}\, (S\, n)\, M\, N = N\, n\, (R_{\mathbb{N}}^{\sigma}\, n\, M\, N)$$

Theorem (Soundness)

Let M be a proof of A from assumptions $u_i : B_i$.

Assuming $u_j : B_j$ if B_j is n.c., $u_j \mid x_{u_i} \# B_i$ otherwise,

there is a proof of A if A is n.c., $et(M) \# A$ otherwise.

Formalization of Prop 7

We use Minlog here.

1. Define the predicates,
2. Give a proof of $\forall_n^{nc}(Nat(n) \to Even(n) \vee Odd(n))$,
3. Do program extraction.

Case studies in real numbers

1. Cauchy reals in constructive math.
2. Corecursion and Coinductive predicates
3. Proofs of the propositions.
4. program extraction in Minlog

Cauchy Reals

Classically, a rational sequence $a^{\mathbb{N}\to\mathbb{Q}}$ is a Cauchy real if $\forall_k \exists_l \forall n,m \geq l \left(|a_n - a_m| \leq 2^{-k}\right)$.

In constructive math, a pair of $a^{\mathbb{N}\to\mathbb{Q}}$ and a so-called "Cauchy modulus" $M^{\mathbb{Z}\to\mathbb{N}}$ is a Cauchy real if $\forall_k \forall n,m \geq M_k \left(|a_n - a_m| \leq 2^{-k}\right)$.

We assume arithmetic of constructive reals.

Corecursion

Corecursion is a constant which is a process of forming (possibly) infinite objects.

Example (${}^{co}R^{\sigma}_{\mathbb{N}}$)

Assume $\mathbb{U}$ and $+$ are the unit and sum datatype, resp.

$${}^{co}R^{\sigma}_{\mathbb{N}} : \sigma \to (\sigma \to \mathbb{U} + (\mathbb{N} + \sigma)) \to \mathbb{N}$$

$${}^{co}R^{\sigma}_{\mathbb{N}}\, N\, M = \text{Case } (M\, N) \text{ of } InL(u) \to 0$$
$$InR(p^{N+\sigma}) \to S(\text{Case } p \text{ of}$$
$$InL(n) \to n$$
$$InR(x) \to {}^{co}R\, x\, M)$$

Example (Infinite natural)

$$\mathrm{coR}^{\mathbb{U}}_{\mathbb{N}}\ ()\ \ \lambda_u\left(\mathrm{InR}\left(\mathrm{InR}(u)\right)\right) \mapsto S(S(S(\ ---$$

Example ($\mathrm{coR}^{\sigma}_{\mathbb{L}_\alpha}$)

$\mathbb{L}_\alpha$ is the list datatype consisting of Nil. : (cons)

$$\mathrm{coR}^{\sigma}_{\mathbb{L}_\alpha} : \sigma \to \left(\sigma \to \mathbb{U} + (\alpha \times (\mathbb{L}_\alpha + \sigma))\right) \to \mathbb{L}_\alpha$$

$\mathrm{coR}^{\sigma}_{\mathbb{L}_\alpha}\ N\ M =$ Case (MN) of

$\mathrm{InL}(u) \to$ Nil

$\mathrm{InR}\left(p^{\alpha \times (\mathbb{L}_\alpha + \sigma)}\right) \to$ (lft p) ; Case (rht p) of

$\mathrm{InL}(xs) \to x\,s$

$\mathrm{InR}(z) \to \mathrm{coR}\ z\ M$

Example (Ascending list of naturals)

$$\lambda n\,({}^{\mathrm{co}}R\ n\ M) \longmapsto n : Sn : S(Sn) : \dots$$

where

$$M := \lambda n\,(\mathrm{Im}R\langle n, \mathrm{Im}R(n+1)\rangle).$$

Example

SD be the datatype consisting of $-1, 0, 1$

predicate $SD(d)$ by $SD(-1)$, $SD(0)$, $SD(1)$

$\exists_x P(x)$ inductively defined, so that we have

$$\forall_x^{nc}(Px \to \exists_x P(x)) \qquad (\exists^+)$$

$$\exists_x P(x) \to \forall_x^{nc}(P(x) \to Q) \to Q \qquad (\exists^-)$$

Example ($^{\omega}I$, approximability of reals in $[-1,1]$)

Let ρ be the type $(\mathbb{N}\to\mathbb{Q})\times(\mathbb{Z}\to\mathbb{N})$.

We define $^{\omega}I$ as a coinductive predicate which comes with the closure axiom and coinduction axiom.

$$\forall^{nc}_{x^\rho}\left(^{\omega}I(x)\to\exists^{SD}_{d}\,\exists_{y^\rho}\left(x=\frac{y+d}{2}\wedge SD(d)\wedge {}^{\omega}I(y)\right)\right)$$

$$\forall^{nc}_{z^\rho}\Big(P(z)\to\forall^{nc}_{x^\rho}\Big(P(x)\to\exists^{SD}_{d}\,\exists_{y^\rho}\Big(x=\frac{y+d}{2}\wedge SD(d)\wedge\big({}^{\omega}I(y)\vee P(y)\big)\Big)\Big)$$

$$\to {}^{\omega}I(z)\Big)$$

Cauchy Real to SDS

Prop 3

$$\forall x\,\Big(\underbrace{x \text{ is a Cauchy real in } [-7,7]}_{P(x)} \to {}^{\omega}\mathcal{I}(x)\Big)$$

Informal Proof use ${}^{\omega}\mathcal{I}^{+}$. It suffices to prove

$$\forall x\,\Big(P(x) \to \exists d\,\exists y\,\big(x = \tfrac{\delta+d}{2} \wedge SD(d) \wedge ({}^{\omega}\mathcal{I}(y) \vee P(y))\big)\Big)$$

Assume x, $P(x)$. We can decide $x \in [-7,0]$ or $x \in [-\frac{7}{2}, \frac{7}{2}]$ or $x \in [0,7]$. In case $[-7,0]$, let d be -7. and y be $2x-7$. Clearly $SD(-7)$ holds, and $P(y)$ as well due to arithmetic. Same arguments for the rest.

Formalization in Minlog

1. proofs of Prop 2 & Prop 3
2. extracted term making use of R.
3. execution in Minlog
4. Haskell translation