Intermediate value theorem

Helmut Schwichtenberg
Joint work with Ulrich Berger, Nils Köpp, Kenyi Miyamoto, Franziskus Wiesnet

Mathematisches Institut, LMU, München

Seminar "Constructive Analysis", 27. Januar 2020

- Functionals of higher type are viewed as unions of certain finite approximations (Scott 1970, Ershov 1972).
- For the latter consistency, entailment and application are canonically defined.
- Define $F(f) = (\bigcup_n F_n)(\bigcup_m f_m) := \bigcup_{n,m} F_n(f_m)$.
- Computablity of F means that its set of finite approximations can be enumerated by an elementary function.

- A theory of computable functionals TCF describes this Scott-Ershov model $\mathcal{C}$ of partial continuous functionals.
- $(f \doteq g) := \forall_{x,y}(x \doteq y \to fx \doteq gy)$.
- "x extensionally realizes A"

$$(f \textbf{ er } A \to B) := \begin{cases} \forall_{x,y}(x \doteq x \to x \textbf{ er } A \to fx \textbf{ er } B) & \text{if } A \text{ c.r.} \\ A \to f \textbf{ er } B & \text{if } A \text{ n.c.} \end{cases}$$

- Invariance axioms: $A \leftrightarrow \exists_x (x \textbf{ er } A)$ "A is invariant under the extensional realizability interpretation" (Kolmogorov 1932).
- Invariance axioms are extensionally realized by identities.
- The computational content of a proof M can be defined as a term $\mathrm{et}(M)$ in TCF.
- Soundness theorem: $\mathrm{TCF} \vdash \mathrm{et}(M) \textbf{ er } A$, for M proof of A.

Standard constructive proofs of the IVT (for instance Bishop 1967) are computationally problematic:

- They partition the interval into as many pieces as the modulus of the continuous function requires (for the given error bound), and
- for each of these (many) pieces perform certain operations.
- Generally this problem is unavoidable: continuous increasing functions may be flat.
- Solution here: Assume a lower bound on the slope.

$$\exists_{q \in \mathbb{Z}^+} \forall_{c,d \in \mathbb{Q}, p \in \mathbb{Z}^+} \Big(\frac{1}{2^p} \leq d - c \to \frac{1}{2^{p+q}} \leq f(d) - f(c) \Big).$$

An auxiliary lemma, which from a "correct" interval $c < d$ (that is, $f(c) \leq 0 \leq f(d)$ and $\frac{1}{2^p} \leq d - c$) constructs a new one $c_1 < d_1$ with $d_1 - c_1 = \frac{2}{3}(d - c)$.

Lemma (IVTAux)

Let $f : I \to \mathbb{R}$ be continuous, with a uniform modulus q of increase. Let $a < b$ be rational numbers in I such that $a \leq c < d \leq b$, say $\frac{1}{2^p} < d - c$, and $f(c) \leq 0 \leq f(d)$. Then we can construct c_1, d_1 with $d_1 - c_1 = \frac{2}{3}(d - c)$, such that again

$$a \leq c \leq c_1 < d_1 \leq d \leq b \quad \textit{and} \quad f(c_1) \leq 0 \leq f(d_1).$$

Proof.

- Let $c_0 = \frac{2c+d}{3}$ and $d_0 = \frac{c+2d}{3}$.
- From $\frac{1}{2^p} < d - c$ we obtain $\frac{1}{p+2} \leq d_0 - c_0$, hence $f(c_0) <_{p+2+q} f(d_0)$.
- Compare 0 with this proper interval, using ApproxSplit.
- Case $0 \leq f(d_0)$. Then let $c_1 = c$ and $d_1 = d_0$.
- Case $f(c_0) \leq 0$. Then let $c_1 = c_0$ and $d_1 = d$.

Theorem (IVT)

Let $f : I \to \mathbb{R}$ be continuous, with a uniform modulus of increase. Let $a < b$ be rational numbers in I such that $f(a) \leq 0 \leq f(b)$. Then we can find $x \in [a, b]$ such that $f(x) = 0$.

Proof. Iterating the construction in IVTAux, we construct two sequences $(c_n)_n$ and $(d_n)_n$ of rationals such that for all n

$$a = c_0 \leq c_1 \leq \cdots \leq c_n < d_n \leq \cdots \leq d_1 \leq d_0 = b,$$
$$f(c_n) \leq 0 \leq f(d_n),$$
$$d_n - c_n = \left(\frac{2}{3}\right)^n (b - a).$$

Then:

- Let x, y be given by the Cauchy sequences $(c_n)_n$ and $(d_n)_n$ with the obvious modulus.
- As f is continuous, $f(x) = 0 = f(y)$ for the real number $x = y$.

ApproxSplit
ApproxSplitBoole
IVTAux
DC
IVTcds
IVTFinal
RealApprox
IVTApprox
IVTInst

ApproxSplit

```
all x,y,z,p(Real x -> Real y -> Real z -> RealLt x y p ->
            z<<=y oru x<<=z)
```

ApproxSplit-neterm ("normalized extracted term")

```
[x,x0,x1,p]
 [case x
   (RealConstr as M ->
   [case x0
     (RealConstr as0 M0 ->
     [case x1
       (RealConstr as1 M1 ->
       as1(M1(PosS(PosS p))max M0(PosS(PosS p))
                           max M(PosS(PosS p)))<=
       (as(M0(PosS(PosS p))max M(PosS(PosS p)))+
        as0(M0(PosS(PosS p))max M(PosS(PosS p))))*
       (1#2))])])]
```

ApproxSplitBoole

```
all x,x0,x1,p(
 Real x -> Real x0 -> Real x1 ->
 RealLt x x0 p ->
 exl boole((boole -> x1<<=x0) andnc
           ((boole -> F) -> x<<=x1)))
```

ApproxSplitBoole-neterm is the same as ApproxSplit-neterm.

IVTAux

```
all f,q(
 Cont f ->
 f f doml<<=0 ->
 0<<=f f domr ->
 all c,d,p(f doml<=c -> d<=f domr -> c+(1#2**p)<=d ->
           RealLt(f c)(f d)(p+q)) ->
 all p,cd(
  Corr f(lft cd)(rht cd)p ->
  exl cd0(
   Corr f(lft cd0)(rht cd0)(PosS p) andnc
   lft cd<=lft cd0 andnc
   rht cd0<=rht cd andnc
   rht cd0-lft cd0==(2#3)*(rht cd-lft cd))))
```

IVTAux-neterm

```
[f,p,p0,cd]
 [let cd0 ((1#3)*(lft cd+lft cd+rht cd)pair
           (1#3)*(lft cd+rht cd+rht cd))
   [if (cApproxSplitBoole
        (RealConstr
         (f approx(lft cd0 max f doml min f domr))
         ([p1]f uMod(PosS p1)))
        (RealConstr
         (f approx(rht cd0 max f doml min f domr))
         ([p1]f uMod(PosS p1)))
        0
        (2+p0+p))
    (lft cd pair rht cd0)
    (lft cd0 pair rht cd)]]
```

DC ("dependent choice")

```
all xx,g(
 RR^ Zero xx ->
 all n,xx0(RR^ n xx0 -> RR^(Succ n)(g n xx0) andnc
                       SS^ n xx0(g n xx0)) ->
 exl xxs(
  xxs Zero eqd xx andnc
  all n RR^ n(xxs n) andnc
  all n SS^ n(xxs n)(xxs(Succ n))))
```

DC-neterm

```
[xx,g,n](Rec nat=>alpha)n xx g
```

IVTcds

```
all f,q,p(
 Cont f ->
 f f doml<<=0 ->
 0<<=f f domr ->
 f doml+(1#2**p)<=f domr ->
 all c,d,p0(
  f doml<=c -> d<=f domr -> c+(1#2**p0)<=d ->
  RealLt(f c)(f d)(p0+q)) ->
 exl cds(
  cds Zero eqd(f doml pair f domr) andnc
  all n Corr f(lft(cds n))(rht(cds n))(NatToPos(p+n))andnc
  all n(
   lft(cds n)<=lft(cds(Succ n)) andnc
   rht(cds(Succ n))<=rht(cds n) andnc
   rht(cds(Succ n))-lft(cds(Succ n))==
   (2#3)*(rht(cds n)-lft(cds n)))))
```

IVTcds-neterm

```
[f,p,p0](cDC rat yprod rat)(f doml pair f domr)
 ([n]cIVTAux f p
   [if (NatEven(p0+n))
     (SZero
     ((GRecGuard nat pos)([n0]n0)(NatHalf(p0+n))
      ([n0,(nat=>pos)]
        [if (NatEven n0) (SZero((nat=>pos)(NatHalf n0)))
          [if (n0=Succ Zero) 1
            (SOne((nat=>pos)(NatHalf n0)))]])
      (NatHalf(p0+n)<p0+n)))
     [if (p0+n=Succ Zero) 1
      (SOne
      ((GRecGuard nat pos)([n0]n0)(NatHalf(p0+n))
       ([n0,(nat=>pos)]
         [if (NatEven n0) (SZero((nat=>pos)(NatHalf n0)))
           [if (n0=Succ Zero) 1
             (SOne((nat=>pos)(NatHalf n0)))]])
        (NatHalf(p0+n)<p0+n)))]])
```

IVTFinal

```
all f,q,p(
 Cont f ->
 f f doml<<=0 ->
 0<<=f f domr ->
 f doml+(1#2**p)<=f domr ->
 f domr-f doml<=2**p ->
 all c,d,p0(
  f doml<=c -> d<=f domr -> c+(1#2**p0)<=d ->
  RealLt(f c)(f d)(p0+q)) ->
 exl x(Real x andnc f x===0))
```

IVTFinal-neterm

```
[f,p,p0]RealConstr([n]lft(cIVTcds f p p0 n))
                  ([p1]TwoThirdExpBd(p1+p0))
```

RealApprox

```
all x,p(Real x -> exl a abs(a+ ~x)<<=(1#2**p))
```

RealApprox-neterm

```
[x,p][case x (RealConstr as M -> as(M p))]
```

IVTApprox

```
all f,q,p(
 Cont f ->
 f f doml<<=0 ->
 0<<=f f domr ->
 f doml+(1#2**p)<=f domr ->
 f domr-f doml<=2**p ->
 all c,d,p0(
  f doml<=c -> d<=f domr -> c+(1#2**p0)<=d ->
  RealLt(f c)(f d)(p0+q)) ->
 exr x(Real x andr f x===0 andr
       all r exl c abs(c+ ~x)<<=(1#2**r)))
```

IVTApprox-neterm

```
[f,p,p0]cRealApprox(cIVTFinal f p p0)
```

IVTInst

```
exr x(Real x andr SqMTwo x===0 andr
      all r exl c abs(c+ ~x)<<=(1#2**r))
```

IVTInst-neterm

```
cIVTApprox
(ContConstr 1 2
 ([a,n]a*a+IntN 2)
 ([p]Zero)
 ([p]PosS(PosS(PosS p)))IntN 1 2)
1
1
```

```
(time (pp (nt (make-term-in-app-form IVTInst-neterm
                                     (pt "16")))))
;; 23585087634298163#16677181699666569
;; 93 ms

(exact->inexact (/ 23585087634298163 16677181699666569))
;; 1.4142130282582281

(sqrt 2)
;; 1.4142135623730951
```