Constructive logic for concurrent real number computation

Ulrich Berger
Swansea University

MAP 2016

January 11-15, CIRM, Luminy, France

Overview

The aim of this talk is to show that concurrent programs can be specified and extracted at an abstract logical level using realizability.

The crucial novelty is a concurrent treatment of disjunction expressed by a semi-constructive axiom scheme and a concurrent realizability interpretation.

As a concrete example we extract a concurrent program translating Tsuiki's infinite Gray-code for real numbers to signed digit representation.

Formal framework

- Intuitionistic many-sorted logic in finite types.
- Sorts represent abstract mathematical structures given by $\vee$-free axioms.
- Inductive and coinductive definitions of predicates as least and greatest fixed points of monotone predicate transformers.
- Realizers are untyped recursive programs.
- The definition of realizability is usual except that quantifiers are interpreted uniformly:
  - $a\,\mathbf{r}\,\exists x\,A(x)$ means $\exists x\,(a\,\mathbf{r}\,A(x))$.
  - $a\,\mathbf{r}\,\forall x\,A(x)$ means $\forall x\,(a\,\mathbf{r}\,A(x))$.

Real, natural and rational numbers

The structure of the *real numbers* $\mathbb{R} = (0, 1, +, *, -, /, <, |\cdot|)$ is treated as a sort specified by $\vee$-free axioms.

The *natural numbers* are defined as the least subset of $\mathbb{R}$ that contains 0 and is closed under successor:

$$\mathbb{N} \stackrel{\mu}{=} \{\, 0 \,\} \cup \{\, x + 1 \mid x \in \mathbb{N} \,\}$$

Realizability automatically associates with this definition the unary representation of natural numbers and with proofs of closure properties of $\mathbb{N}$ programs operating on that representation.

$$\mathbb{Q} := \{\, p * \frac{m}{n} \mid p \in \{-1, 1\}, m, n \in \mathbb{N}, n \neq 0 \,\} \subseteq \mathbb{R}$$

Cauchy- and Signed-Digit-representation

A **Cauchy representation** of a real number $x \in \mathbb{I} = [-1, 1] \subseteq \mathbb{R}$ is a sequence $f : \mathbb{N} \to \mathbb{Q}$ such that for all $n \in \mathbb{N}$

$$|x - f(n)| \leq 2^{-n}$$

A **signed digit representation** of a real number $x \in \mathbb{I}$ is a stream $d_0 : d_1 : \ldots \in \mathrm{SD}^\omega$, where $\mathrm{SD} = \{-1, 0, 1\}$, such that

$$x = \sum_{i \in \mathbb{N}} d_i * 2^{-(i+1)}$$

Gray code

Gray code (named after Frank Gray in 1946 who called it "reflected binary code") is an alternative to the binary representation of natural numbers where neighbouring numbers differ in only one digit.

Tsuiki extended this to a representation of real numbers.

Hideki Tsuiki: Real Number Computation through Gray Code Embedding. TCS 284, 2002.

(Dagstuhl seminar *Mathematical Structures for Computable Topology and Geometry*, May 2002)

Tsuiki's partial Gray code for real numbers

The **Gray code** or **Gray representation** of $x \in [-1,1]$ is the itinerary of the tent map $\mathrm{t}(x) = 1 - 2|x|$. This means that the n-th digit is 0 resp. 1 if $t^n(x) < 0$ resp. > 0.

If $t^n(x) = 0$, then the n-th digit is undefined.

Remarkably, **every real in** $[-1,1]$ **has a unique Gray code**.

One easily sees that at most one digit of the Gray code can be undefined. Therefore, computation with the Gray code can be modeled by a *Two-Head-Turing-Machine*.

Such a machine cannot be extracted from a proof in the current system since it exhibits a kind of parallelism, or rather concurrency, that is absent in extracted programs.

Problem: Devise a logic that can extract concurrent programs.

Related work: Realizing Gray Code deterministically

B., Kenji Miyamoto, Helmut Schwichtenberg, Hideki Tsuiki: Logic for Gray-code computation (submitted)

gives a realizability interpretation and Minlog implementation of an intensional version of Gray Code, called pre-Gray code, using a conventional constructive system and conventional program extraction. This skirts the issue of concurrency at the price of giving up the uniqueness of Gray code.

This approach is currently being extended to pre-Gray code for compact sets by Dieter Spreen and Hideki Tsuiki.

In this talk we dive headlong into concurrency. The results that follow are not published yet, but exist as a draft paper.

Representations in logical form

We call a predicate $A(x)$ a

$$\Phi\text{-}\textit{representation in logical form}$$

where Φ is a notion of representation, for example 'Cauchy', 'Signed digit' or 'Gray', if for all $x \in \mathbb{I}$ and potential realizers a

$$a \,\mathbf{r}\, A(x) \text{ iff } a \text{ is is an } \Phi\text{-representation of } x$$

I.o.w. the realizers of $A(x)$ are exactly the Φ-representations of x.

Three representations of real numbers in logical form

Cauchy representation

$\mathrm{A}(x) := \forall n \in \mathbb{N}\, \exists q \in \mathbb{Q} \cap \mathbb{I}\,.\, |x - q| \leq 2^{-n}$

Signed Digit representation

$\mathrm{C}(x) \stackrel{\nu}{=} \exists d \in \mathrm{SD}\,.\, x \in \mathbb{I}_d \wedge \mathrm{C}(2x - d)$

where $\mathbb{I}_d := [d/2 - 1/2, d/2 + 1/2]$ and $\stackrel{\nu}{=}$ means 'largest'.

Gray code

$\mathrm{G}(x) \stackrel{\nu}{=} \mathrm{D}(x) \wedge \mathrm{G}(\mathrm{t}(x))$

where $\mathrm{D}(x) := x \neq 0 \to x \leq 0 \vee x \geq 0$.

We want to show constructively $\mathrm{A} = \mathrm{C} = \mathrm{G}$ which will give us programs translating between the three representations.

We will show $\mathrm{A} \subseteq \mathrm{G} \subseteq \mathrm{C} \subseteq \mathrm{A}$.
Since the last inclusion is straightforward, we omit it.

Proving $\mathrm{A} \subseteq \mathrm{G}$

To prove $\mathrm{A} \subseteq \mathrm{G}$ one seems to need the following semi-constructive principles:

(AP) $\forall x\,.\,(\forall n \in \mathbb{N}\,|x| \leq 2^{-n}) \to x = 0$
(Archimedean Property).

(AC^{ω}) $(\forall n \in \mathbb{N}\,\exists q \in \mathbb{Q}\,A(n,q)) \to \exists f : \mathbb{N} \to \mathbb{Q}\,\forall n \in \mathbb{N}\,A(n, f(n))$
(countable choice for rational numbers).

(MP) $(\forall n \in \mathbb{N}\,.\,A(n) \vee \neg A(n)) \wedge (\neg\neg\exists n \in \mathbb{N}\,A(n)) \to \exists n \in \mathbb{N}\,A(n)$
(Markov's Principle)

AP has a trivial realizer, AC^{ω} is realized by the identity,
MP is realized by unbounded search.

Proving $\mathrm{G} \subseteq \mathrm{C}$

The only really hard part of the proof is to determine the first signed digit of an $x \in \mathrm{G}$, that is, to show

If $x \in \mathrm{G}$, then $x \in \mathbb{I}_d$ for some $d \in \mathrm{SD}$.

For this we use the following *Disjunction Principle*:

(DP) $(A \overset{P}{\vee} B) \wedge (P \overset{Q}{\vee} C) \to (A \vee B) \vee C$

where

$$A \overset{P}{\vee} B := (P \to A \vee B) \wedge (\neg P \to A \wedge B)$$

and A, B, C, P, Q range over propositions without computational content.

Determining the first signed digit of an $x \in \mathrm{G}$

Lemma

If $x \in \mathrm{G}$, then $x \in \mathbb{I}_d$ for some $d \in \mathrm{SD}$.

Proof.

Assume $x \in \mathrm{G}$. Then $\mathrm{D}(x)$ and $\mathrm{D}(\mathrm{t}(x))$. Apply (DP) with

$$A := x \in \mathbb{I}_{-1}, \quad B := x \in \mathbb{I}_1, \quad C := x \in \mathbb{I}_0,$$

$$P := x \neq 0, \quad Q := \mathrm{t}(x) \neq 0.$$

We have to show $A \overset{P}{\vee} B$ and $P \overset{Q}{\vee} C$.

- $P \to A \vee B$ is $\mathrm{D}(x)$.
- To show $\neg P \to A \wedge B$, assume $\neg(x \neq 0)$. Then clearly $x \in \mathbb{I}_{-1}$ and $x \in \mathbb{I}_1$.
- To show $Q \to P \vee C$, assume $\mathrm{t}(x) \neq 0$. Then $\mathrm{t}(x) \leq 0 \vee \mathrm{t}(x) \geq 0$, since $\mathrm{D}(\mathrm{t}(x))$. If $\mathrm{t}(x) \leq 0$, then $|x| \geq \frac{1}{2}$, hence $x \neq 0$. If $\mathrm{t}(x) \geq 0$, then $x \in \mathbb{I}_0$.
- To show $\neg Q \to P \wedge C$, assume $\neg(\mathrm{t}(x) \neq 0)$. Then $|x| = \frac{1}{2}$, hence $x \neq 0$ and $x \in \mathbb{I}_0$.

Concurrent realizability

We add to the domain of realizers a new component $D^{\mathcal{I}}$ for non-deterministic concurrent computations

$$D \simeq 1 + D + D + D \times D + [D \to D] + D^{\mathcal{I}}$$

where $\mathcal{I}$ is the least set containing the constant $*$ and with i, j the elements $(0, i)$, $(1, i)$ and (i, j).

The constructors of D are called Nil, In_0, In_1, Pair, Fun, Fam.

Realizability of disjunctions is redefined as

$$\begin{aligned} a \,\mathbf{r}\, (A_0 \vee A_1) \quad := \quad & (\exists \varphi \, \exists p \in \{0,1\} \, \exists i \, \exists b \,.\, a = \mathrm{Fam}\, \varphi \wedge \varphi\, i = \mathrm{In}_p\, b) \wedge \\ & (\forall \varphi \, \forall p \, \forall i \, \forall b \,.\, a = \mathrm{Fam}\, \varphi \wedge \varphi\, i = \mathrm{In}_p\, b \to b \,\mathbf{r}\, A_p) \end{aligned}$$

Compare this with the original definition

$$a \,\mathbf{r}\, (A_0 \vee A_1) \quad := \quad \exists p \in \{0,1\} \, \exists b \,.\, a = \mathrm{In}_p\, b \wedge b \,\mathbf{r}\, A_p$$

Weakening disjunction elimination

With the new realizability interpretation of disjunction the usual disjunction elimination is no longer realizable.

We replace it by two rules:

Disjunction Transformation (DT):

$$\frac{\Gamma \vdash A \vee B \qquad \Gamma, A \vdash A' \qquad \Gamma, B \vdash B'}{\Gamma \vdash A' \vee B'}$$

Disjunction Contraction (DC):

$$\frac{\Gamma \vdash A \vee A}{\Gamma \vdash A} \qquad (A \text{ a disjunction})$$

Weak disjunction elimination

Disjunction elimination can be partly recovered by the following derived rule:

$$\frac{\Gamma \vdash A \vee B \qquad \Gamma, A \vdash C \qquad \Gamma, B \vdash C}{\Gamma \vdash C} \qquad (C \text{ a quasi disjunction})$$

where quasi disjunctions are inductively defined as follows:

(i) Every disjunction is a quasi disjunction.

(ii) If A is a quasi disjunction, so are $\forall x\, A$ and $B \to A$ (for arbitrary variables x and formulas B).

(iii) If A and B are quasi disjunctions, so is $A \wedge B$.

(iv) If Φ is a quasi disjunction, strictly positive in X, then $(\mu\,(\lambda X\, \lambda \vec{x}\,.\,\Phi))\,\vec{t}$ and $(\nu\,(\lambda X\, \lambda \vec{x}\,.\,\Phi))\,\vec{t}$ are quasi disjunctions.

Soundness for concurrent realizability

Let Conc be the logical system where Disjunction Elimination is replaced by Disjunction Transformation and Disjunction Contraction, and the Archimedean Principle, Countable Choice and Markov's Principle are added.

Let $a \,\mathbf{cr}\, A$ mean that a realizes A where disjunction is realized concurrently.

Theorem (Soundness Theorem for concurrent realizability)

If $\mathrm{Conc} \vdash A$, *then* $\mathrm{Conc} \vdash M \,\mathbf{cr}\, A$ *for some concurrent program term* $M : \delta$.

Proof.

We only look at the disjunction principle . . . □

Realizing the Disjunction Principle

Assume $a \,\mathbf{cr}\, (A \overset{P}{\vee} B)$ and $b \,\mathbf{cr}\, (P \overset{Q}{\vee} C)$.

Then $a = \mathrm{Fam}\, \varphi$ and $b = \mathrm{Fam}\, \psi$.

Define $\chi \in D^{\mathcal{I}}$ by

$$\begin{aligned} \chi(0, i) &:= \text{if } \varphi(i) = \mathrm{In}_p\, b \text{ then } \mathrm{In}_0\, a \text{ else } \bot \\ \chi(1, i) &:= \text{if } \psi(i) = \mathrm{In}_1\, c \text{ then } \mathrm{In}_1\, c \text{ else } \bot \\ \chi(j) &:= \bot \quad \text{otherwise} \end{aligned}$$

Then $(\mathrm{Fam}\, \chi) \,\mathbf{cr}\, ((A \vee B) \vee C)$.

This can be checked by a lengthy, but elementary argument involving classical logic.

Concurrent operational semantics

Intuitively, a family $\varphi \in D^{\mathcal{I}}$ represents a choice between all family members. Hence, roughly speaking, we have for each $i \in \mathcal{I}$ a reduction rule rule

$$\varphi \longrightarrow \varphi(i)$$

The Soundness Theorem is a result about the denotational semantics of realizers.

For the extraction of runnable programs we must link the denotational with the operational semantics.

Computational Adequacy and Program Extraction

Theorem (Computational Adequacy)

If d is in the set $\mathrm{data}(M)$ of data denoted by M, then M reduces to d.

Furthermore, one can show that the two notions of realizability coincide for data formulas A.

Theorem (Faithfulness)

*If M **cr** A, then $\mathrm{data}(M) \neq \emptyset$ and d **r** A for all $d \in \mathrm{data}(M)$.*

All together one obtains:

Theorem (Program Extraction)

*From a proof $\mathrm{Conc} \vdash A$ one can extract a concurrent program term M such that M redues to some data d with d **r** A.*

Parallelism in Exact Real Number Computation

- Computing with the interval domain as a model of real numbers appears to require a parallel if-then-else operation (Potts, Edalat, Escardo, 1997).
- In fact, this parallelism is unavoidable (Escardo, Hofmann, Streicher, 2004).
- Computing with TTE representations (e.g. Cauchy- or signed digit representation) does *not* require parallelism.
- Gray code (though very similar to signed digits) requires parallelism.

Extracted programs for Gray code

Lemma 9. If $x \in \mathrm{G}$, then $x \in \mathbb{I}_d$ for some $d \in \mathrm{SD}$.

```
f9 (a:s) = conv a   if  a /= bot
f9 (a:1:s) = 0

  where conv 0 = -1
        conv 1 = 1
        conv bot = bot
```

The equations above should be read as overlapping reduction rules.

Lemma 10. If $x \in \mathrm{G}$, then $-x \in \mathrm{G}$.

```
f10 (a:s) = swap a : s

  where swap 0 = 1
        swap 1 = 0
        swap bot = bot
```

Extracted programs for Gray code ctd.

Lemma 11. If $0 \leq x \leq 1$ and $\mathrm{G}(x)$, then $\mathrm{G}(2x - 1)$.

```
f11 (a:s) = f10 s
```

hence

```
f11 (a:b:s) = swap b : s
```

Lemma 12. If $-1 \leq x \leq 0$ and $\mathrm{G}(x)$, then $\mathrm{G}(2x + 1)$.

```
f12 (a:s) = s
```

Extracted programs for Gray code ctd.

Lemma 13. If $0 \leq x \leq 1$ and $\mathrm{G}(x)$, then $\mathrm{G}(1 - x)$.

```
f13 (a:s) = 1 : f11 (a:s)
```

Hence

```
f13 (a:b:s) = 1 : swap b : s
```

Lemma 14. If $-\frac{1}{2} \leq x \leq \frac{1}{2}$ and $\mathrm{G}(x)$, then $\mathrm{G}(2x)$.

```
f14 (a:s) = a : f13 s
```

Hence

```
f14 (a:b:c:s) = a : swap c : s
```

Extracted programs for Gray code ctd.

Lemma 15. $\mathrm{G} \subseteq \mathrm{C}$.

```
f15 s = let { d = f9 s}
        in d : case d of { -1 -> f12 s ;
                            0 -> f14 s ;
                            1 -> f11 s }
```

Hence

```
f15 (0:s)     = -1 : f15 s
f15 (1:a:s)   =  1 : f15 (swap a : s)
f15 (a:1:c:s) =  0 : f15 (a : swap c : s)
```

Again, read the equations above as overlapping rewrite rules.

One observes that the equations for `f15` correspond exactly to those given by Tsuiki.

Next steps

- Replace the Disjunction Principle by a more fundamental axiom.
- Implement everything and do more case studies.