Extracting non-deterministic concurrent programs

Ulrich Berger
Swansea University

CSL 2016

August 30, Marseille

Overview

Non-deterministic and concurrent extensions of functional programming languages have been intensively studied.

The aim of this talk is to show that programs in such languages can be *specified* and *extracted from proofs* using realizability.

The crucial novelty is a modality permitting non-deterministic concurrent realizers and a semi-constructive Disjunction Principle.

As a concrete example we extract a non-deterministic concurrent program that translates Tsuiki's infinite Gray-code for real numbers to the 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:

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

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

Φ-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

$\mathrm{C}(x) \overset{\nu}{=} \exists d \in \mathrm{SD}\,.\, x \in \mathbb{I}_d \wedge \mathrm{C}(2x - d)$
where $\mathbb{I}_d := [d/2 - 1/2, d/2 + 1/2]$ and $\overset{\nu}{=}$ means 'largest'.

Gray code

$\mathrm{G}(x) \overset{\nu}{=} \mathrm{D}(x) \wedge \mathrm{G}(\mathrm{t}(x))$
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*:

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

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.

Concurrent Fixed Point Logic

DP is classically trivial, but constructively invalid, in particular *not realizable*.

Hence the proof of the Lemma doesn't yield an algorithm to compute the first signed digit of an $x \in \mathrm{G}$.

Concurrent Fixed Point Logic (CFP) comes to our rescue.

We add a modal operator S and the rules

$$\frac{\Gamma \vdash A}{\Gamma \vdash \mathrm{S}(A)}\,(\mathrm{S}^+) \qquad \frac{\Gamma \vdash \mathrm{S}(A) \qquad \Gamma, A \vdash \mathrm{S}(B)}{\Gamma \vdash \mathrm{S}(B)}\,(\mathrm{S}^-)$$

Realizing $\mathrm{S}(A)$

Intuitively, a realizer of $\mathrm{S}(A)$ is a partial family a, indexed by some discrete set $\mathcal{I}$, such that

(i) $a(i)$ yields a result for at least one $i \in \mathcal{I}$,

(ii) for any $i \in \mathcal{I}$, if $a(i)$ yields a result b, then b realizes A.

The formal definition uses a binary constructor Amb representing nondeterministic choice between two alternatives.

Condition (i) is captured by an inductive definition,

condition (ii) by a coinductive definition.

Embedding IFP

To each IFP formula A we assign a CFP formula A^{S} by applying to each disjunctive and existential subformula the modality S.

Theorem (Embedding)

If $\Gamma \vdash_{\mathrm{IFP}} A$*, then* $\Gamma^{\mathrm{S}} \vdash_{\mathrm{CFP}} A^{\mathrm{S}}$*.*

Theorem (Soundness for CFP)

From a proof in CFP *of* $\Delta, \Gamma \vdash A$*, where* Δ *consists of nc formulas, one can extract a concurrent program term* M *such that* $\Delta, \vec{a}\,\mathbf{r}\,\Gamma \vdash (M\,\vec{a})\,\mathbf{r}\,A$*.*

Theorem (Concurrent Soundness)

From a proof of $\Delta, \Gamma \vdash_{\mathrm{IFP}} A$*, where* Δ *is nc, one can extract a concurrent program* M *such that* $\Delta, \vec{a}\,\mathbf{r}\,\Gamma^{\mathrm{S}} \vdash (M \cdot \vec{a})\,\mathbf{r}\,A^{\mathrm{S}}$*.*

Concurrently realizing the Disjunction Principle

Note that DP^{S} is (equivalent to)

$$(A \overset{P}{\vee} B)^{\mathrm{S}} \wedge (P \overset{Q}{\vee} C)^{\mathrm{S}} \to \mathrm{S}(A \vee B \vee C)$$

where
$(A \overset{P}{\vee} B)^{\mathrm{S}} = (P \to \mathrm{S}(A \vee B)) \wedge (\neg P \to A \wedge B)$,
$(P \overset{Q}{\vee} C)^{\mathrm{S}} = (Q \to \mathrm{S}(P \vee C)) \wedge (\neg Q \to P \wedge C)$

The following function realizes DP^{S}

```
fDP (0,b) = -1
fDP (1,b) = 1
fDP (a,1) = 0
```

These equation must be interpreted as nondeterministic rewrite rules.

Extracted programs for Gray code

Lemma 6. If $x \in \mathrm{G}$, then $x \in \mathbb{I}_d$ for some $d \in \mathrm{SD}$.

```
f6 (a:b:s) = fDP(a,b)
```

Lemma 7.

(a) If $x \in \mathrm{G}$, then $-x \in \mathrm{G}$.

(b) If $x \in \mathrm{G}$, then $|x| \in \mathrm{G}$.

```
f7a (a:s) = swap a : s  (swap 0 = 1, swap 1 = 0)
f7b (a:s) = 1:s
```

Extracted programs for Gray code ctd.

Lemma 8. If $0 \leq x \leq 1$ and $\mathrm{G}(x)$, then $\mathrm{G}(2x-1)$.

```
f8 (a:s) = f7a s
```

hence

```
f8 (a:b:s) = swap b : s
```

Lemma 9. If $-1 \leq x \leq 0$ and $\mathrm{G}(x)$, then $\mathrm{G}(2x+1)$.

```
f9 (a:s) = s
```

Extracted programs for Gray code ctd.

Lemma 10. If $0 \leq x \leq 1$ and $\mathrm{G}(x)$, then $\mathrm{G}(1-x)$.

```
f10 (a:s) = 1 : f8(a:s)
```

Hence

```
f10 (a:b:s) = 1 : swap b : s
```

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

```
f11 (a:s) = a : f9(f10 s)
```

Hence

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

Extracted programs for Gray code ctd.

Lemma 12. $\mathrm{G} \subseteq \mathrm{C}$.

```
f12 s = let { d = f9 s}
        in d : case d of { -1 -> f12(f9 s) ;
                            0 -> f12(f11 s) ;
                            1 -> f12(f8 s) }
```

Hence

```
f12 (0:s)     = -1 : f12 s
f15 (1:a:s)   =  1 : f12 (swap a : s)
f15 (a:1:c:s) =  0 : f12 (a : swap c : s)
```

Again, read the equations above as overlapping rewrite rules.

`f12` is exactly Tsuiki's programgtos translating signed digit into to Gray code.

Nondeterminism in Exact Real Number Computation

- Computing with the interval domain as a model of real numbers appears to require a parallel (rather nondeterministic) if-then-else operation (Potts, Edalat, Escardo, 1997).

- In fact, this nondeterminism is unavoidable (Escardo, Hofmann, Streicher, 2004).

- Computing with TTE representations (e.g. Cauchy- or signed digit representation) does *not* require nondeterminism.

- Gray code (though very similar to signed digits) requires nondeterminism.

Next steps (j.w.w. Hideki Tsuiki)

- Replace the Disjunction Principle by a more fundamental axiom.

  Current solution: Generalise $\mathrm{S}(A)$ to $(A \mid B)$ (read “nondetermistically A if B, hence $\mathrm{S}(A) = (A \mid \mathrm{True})$), and replace the Disjunction Principle by nondeterministic elimination for classical disjunction

  $$(C \mid A) \to (C \mid B) \to (C \mid \neg(\neg A \land \neg B))$$

  which is realized by Amb.

- Extract Tsuiki's program translating the signed digit representation into Gray code:

  ```
  stog (-1:s) = 0 : stog s
  stog ( 1:s) = 1 : nh(stog s) -- nh(a:s) = (1-a):s
  stog ( 0:s) = a : 1 : nh t   where a : t = stog s
  ```