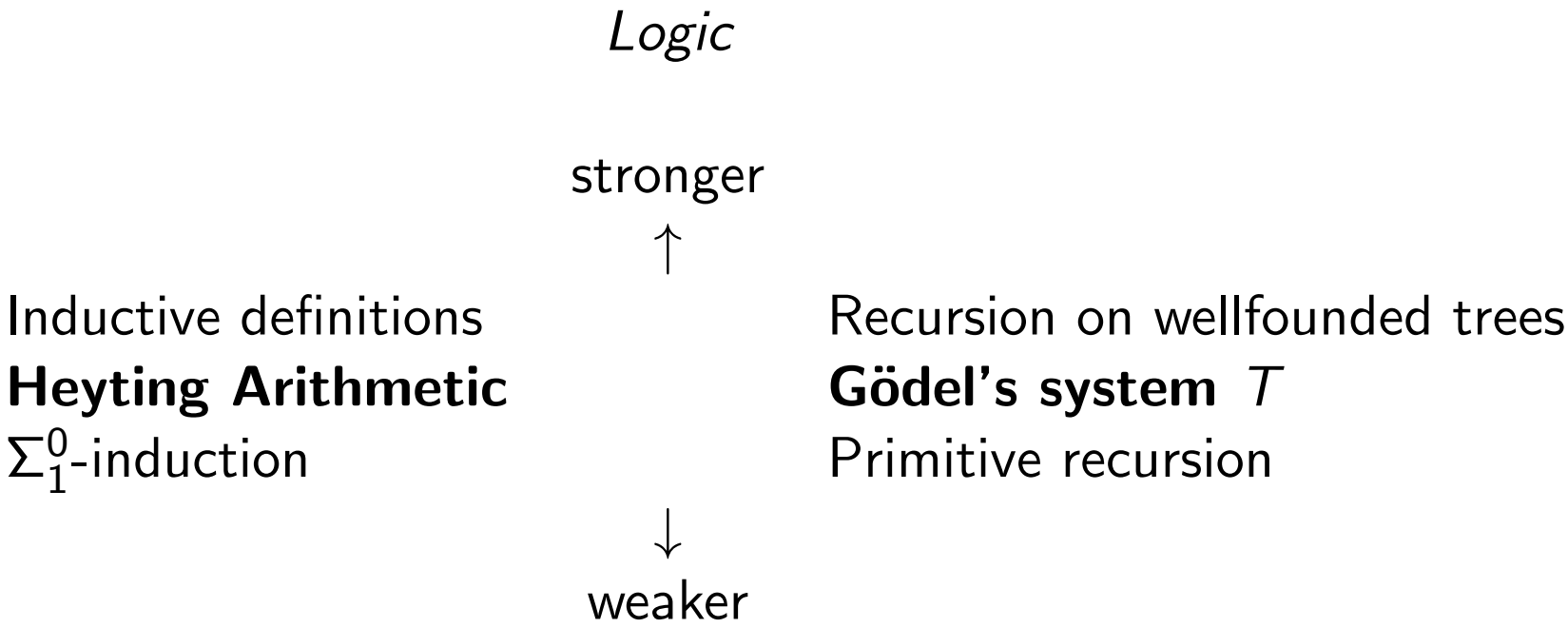
Concurrent and nondeterministic program extraction

Ulrich Berger
Swansea University

Workshop on Intersection Types

June 13-14, Paris

Proofs as Programs



Proofs as Programs

Logic

stronger

↑

Inductive definitions	Recursion on wellfounded trees
Heyting Arithmetic	**Gödel's system** T
Σ^0_1-induction	Primitive recursion

↓

weaker

CS

Classical logic	Non-local control
?	Side effects
?	Concurrency, nondeterminism

Intuitionistic Fixed Point Logic (IFP)

- Intuitionistic many-sorted logic in finite types.
- Sorts represent (carriers of) abstract mathematical structures given by *non-computational (nc)*, that is, disjunction-free axioms.
- Inductive and coinductive definitions of predicates as least and greatest fixed points of monotone predicate transformers.

Intuitionistic Fixed Point Logic (IFP)

- Intuitionistic many-sorted logic in finite types.
- Sorts represent (carriers of) abstract mathematical structures given by *non-computational (nc)*, that is, disjunction-free axioms.
- Inductive and coinductive definitions of predicates as least and greatest fixed points of monotone predicate transformers.

Our leading example will be from computable analysis.

Realizability

In this talk, program extraction will be based on realizability.

Realizability

In this talk, program extraction will be based on realizability.

The definition of realizability is as usual (Kleene) 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 $\forall x\, (a \,\mathbf{r}\, A(x))$.

Realizability

In this talk, program extraction will be based on realizability.

The definition of realizability is as usual (Kleene) 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 $\forall x\, (a \,\mathbf{r}\, A(x))$.

A realizer of an implication $A \to B$ is a function f such that whenever a realizes A, then $f(a)$ realizes B.

f may be partial, i.e. $f(a)$ need not be defined for all a.

Realizability involving nc formulas

If A is nc, then $a\,\mathbf{r}\,A$ means just A.

Hence, if A and B are nc formulas, then

$$a\,\mathbf{r}\,(A \vee B) \text{ means } (a = 0 \wedge A) \vee (a = 1 \wedge B)$$

If A is nc, then $a\,\mathbf{r}\,(A \to C)$ means $A \to a\,\mathbf{r}\,C$.

Real, natural and rational numbers

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

Real, natural and rational numbers

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

Real, natural and rational numbers

The structure of the *real numbers* $\mathbb{R} = (0, 1, +, *, -, /, <, |\cdot|)$ is treated as a sort with constants and functions specified by nc 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 $n \in \mathbb{N}$

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

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.

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.

Tsuiki's partial infinite Gray code for real numbers

The **Gray code** 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 $\mathrm{t}^n(x) < 0$ resp. > 0.

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

Tsuiki's partial infinite Gray code for real numbers

The **Gray code** 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 $\mathrm{t}^n(x) < 0$ resp. > 0.

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

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

Since $\mathrm{t}(0) = 1$ and $\mathrm{t}(1) = \mathrm{t}(-1) = -1$, 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*.

Translating Gray code to signed digit representation

Tsuiki's Two-Head Turing Machines can execute the following translation of Gray code to signed digit representation.

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

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

Problem: Devise a logic that can extract concurrent nondeterministic programs.

Related work: Realizing Gray Code deterministically

The closest to Gray code one seems to get with traditional program extraction is a program that works on so-called pre Gray code, i.e., streams representing constructions of Gray code.

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

Related work: Realizing Gray Code deterministically

The closest to Gray code one seems to get with traditional program extraction is a program that works on so-called pre Gray code, i.e., streams representing constructions of Gray code.

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

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

Cauchy representation in logical form

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

Then $f \mathbf{r} A(x)$ iff f is a Cauchy representation of x.

We call the predicate $A(x)$ the *Cauchy representation in logical form*.

Signed digit representation in logical form

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

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

Signed digit representation in logical form

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

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

A realizer of $\mathrm{C}(x)$ is a stream of signed digits whose head realizes $x \in \mathbb{I}_d$ for some $d \in \mathrm{SD}$ (which essentially means the head equals d) and whose tail realizes $\mathrm{C}(2x - d)$.

It follows that the realizers of $\mathrm{C}(x)$ are exactly the signed digit representations of x.

Gray code in logical form

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

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

Gray code in logical form

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

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

Note that $a \,\mathbf{r}\, D(x)$ means
$x \neq 0 \to (a = 0 \wedge x \leq 0) \vee (a = 1 \wedge x \geq 0)$.

Hence a realizer of $\mathrm{G}(x)$ is a stream of partial binary digits whose head realizes $\mathrm{D}(x)$ and whose tail realizes $\mathrm{G}(\mathrm{t}(x))$.

This means that the Gray code of x is a realizer of $\mathrm{G}(x)$.

Gray code in logical form

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

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

Note that $a \mathbf{r} D(x)$ means
$x \neq 0 \to (a = 0 \wedge x \leq 0) \vee (a = 1 \wedge x \geq 0)$.

Hence a realizer of $\mathrm{G}(x)$ is a stream of partial binary digits whose head realizes $\mathrm{D}(x)$ and whose tail realizes $\mathrm{G}(\mathrm{t}(x))$.

This means that the Gray code of x is a realizer of $\mathrm{G}(x)$.

It is our goal to show constructively $\mathrm{A} = \mathrm{C} = \mathrm{G}$ which will give us programs translating between the three representations.

Gray code in logical form

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

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

Note that $a \mathbf{r} D(x)$ means
$x \neq 0 \to (a = 0 \wedge x \leq 0) \vee (a = 1 \wedge x \geq 0)$.

Hence a realizer of $\mathrm{G}(x)$ is a stream of partial binary digits whose head realizes $\mathrm{D}(x)$ and whose tail realizes $\mathrm{G}(\mathrm{t}(x))$.

This means that the Gray code of x is a realizer of $\mathrm{G}(x)$.

It is our goal to show constructively $\mathrm{A} = \mathrm{C} = \mathrm{G}$ which will give us programs translating between the three representations.

We concentrate on the inclusion $\mathrm{G} \subseteq \mathrm{C}$.

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

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

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 nc propositions.

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

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 nc propositions.

We apply (DP) with

$$A := x \in \mathbb{I}_{-1}, \quad B := x \in \mathbb{I}_1, \quad C := x \in \mathbb{I}_0,$$
$$P := x \neq 0, \quad Q := \mathrm{t}(x) \neq 0.$$

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

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 $\mathrm{S}(A)$

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.

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, similar to the program `gtos`.

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
f9 (a:1:s) = 0

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

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
```

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.

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 nondeterministic parallelism.

Nondeterministic realizability

This talks was based on the paper

- B. Extracting nondeterministic concurrent programs (accepted for CSL 2016)

which considers realizing program language with an infinite choice construct, $\langle\alpha\rangle M$, and the reduction rules

$$\langle\alpha\rangle M \longrightarrow M[i/\alpha] \qquad (i \in \mathcal{I})$$

where $\mathcal{I}$ is an infinite index set.

Nondeterministic realizability

This talks was based on the paper

- B. Extracting nondeterministic concurrent programs (accepted for CSL 2016)

which considers realizing program language with an infinite choice construct, $\langle\alpha\rangle M$, and the reduction rules

$$\langle\alpha\rangle M \longrightarrow M[i/\alpha] \qquad (i \in \mathcal{I})$$

where $\mathcal{I}$ is an infinite index set.

This can be reduced to binary choice

$$M + N \longrightarrow M, \qquad M + N \longrightarrow N$$

as in

- Bucciarelly, Ehrhard, Manzonetto. A relational semantics for parallelism and non-determinism in a functional setting. APAL 163(7): 918-934 (2012)

Thank You

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

Lemma

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

Proof.

Assume $\mathrm{G}(x)$. 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 $x \in \mathbb{I}_0$.