Computability Theory

Barbara Csima

May 8, 2026

EMAIL: csima@uwaterloo.ca
WEB: https://sites.google.com/view/barbaracsima/
Department of Pure Mathematics
University of Waterloo

Low Basis Theorem

Theorem (Jockusch, Soare)

Every infinite computable binary branching tree has a low path.

Let T be an infinite computable binary branching tree.
We will use a $\varnothing'$-oracle to build a path through T by stages, while also at each stage restricting to a possibly smaller infinite computable binary branching tree of possible extensions.

Low Basis Theorem

Theorem (Jockusch, Soare)

Every infinite computable binary branching tree has a low path.

Let T be an infinite computable binary branching tree.
We will use a $\varnothing'$-oracle to build a path through T by stages, while also at each stage restricting to a possibly smaller infinite computable binary branching tree of possible extensions.

Stage 0: Let σ_0 be the empty string and let $T_0 = T$.

Low Basis Theorem

Theorem (Jockusch, Soare)

Every infinite computable binary branching tree has a low path.

Let T be an infinite computable binary branching tree.
We will use a $\varnothing'$-oracle to build a path through T by stages, while also at each stage restricting to a possibly smaller infinite computable binary branching tree of possible extensions.

Stage 0: Let σ_0 be the empty string and let $T_0 = T$.

Stage $e+1$: Consider the computable tree
$U_e = \{\tau \succ \sigma_e \mid \tau \in T_e \wedge \Phi^{\tau}_{e,|\tau|}(e)\uparrow\}$.

Low Basis Theorem

Theorem (Jockusch, Soare)

Every infinite computable binary branching tree has a low path.

Let T be an infinite computable binary branching tree.
We will use a $\varnothing'$-oracle to build a path through T by stages, while also at each stage restricting to a possibly smaller infinite computable binary branching tree of possible extensions.

Stage 0: Let σ_0 be the empty string and let $T_0 = T$.

Stage $e+1$: Consider the computable tree $U_e = \{\tau \succ \sigma_e \mid \tau \in T_e \wedge \Phi^{\tau}_{e,|\tau|}(e)\uparrow\}$. We use the $\varnothing'$ oracle to decide whether or not U_e is infinite.

Low Basis Theorem

Theorem (Jockusch, Soare)

Every infinite computable binary branching tree has a low path.

Let T be an infinite computable binary branching tree.
We will use a $\varnothing'$-oracle to build a path through T by stages, while also at each stage restricting to a possibly smaller infinite computable binary branching tree of possible extensions.

Stage 0: Let σ_0 be the empty string and let $T_0 = T$.

Stage $e+1$: Consider the computable tree $U_e = \{\tau \succ \sigma_e \mid \tau \in T_e \wedge \Phi^{\tau}_{e,|\tau|}(e)\uparrow\}$. We use the $\varnothing'$ oracle to decide whether or not U_e is infinite. If U_e is infinite, set $T_{e+1} = U_e$. If not, set $T_{e+1} = T_e$.

Low Basis Theorem

Theorem (Jockusch, Soare)

Every infinite computable binary branching tree has a low path.

Let T be an infinite computable binary branching tree.
We will use a $\varnothing'$-oracle to build a path through T by stages, while also at each stage restricting to a possibly smaller infinite computable binary branching tree of possible extensions.

Stage 0: Let σ_0 be the empty string and let $T_0 = T$.

Stage $e+1$: Consider the computable tree $U_e = \{\tau \succ \sigma_e \mid \tau \in T_e \wedge \Phi^{\tau}_{e,|\tau|}(e)\uparrow\}$. We use the $\varnothing'$ oracle to decide whether or not U_e is infinite. If U_e is infinite, set $T_{e+1} = U_e$. If not, set $T_{e+1} = T_e$. Let $\sigma_{e+1} \succ \sigma_e$ in T_{e+1}.

Low Basis Theorem

Theorem (Jockusch, Soare)

Every infinite computable binary branching tree has a low path.

Let T be an infinite computable binary branching tree.
We will use a $\varnothing'$-oracle to build a path through T by stages, while also at each stage restricting to a possibly smaller infinite computable binary branching tree of possible extensions.

Stage 0: Let σ_0 be the empty string and let $T_0 = T$.

Stage $e+1$: Consider the computable tree $U_e = \{\tau \succ \sigma_e \mid \tau \in T_e \wedge \Phi^{\tau}_{e,|\tau|}(e)\uparrow\}$. We use the $\varnothing'$ oracle to decide whether or not U_e is infinite. If U_e is infinite, set $T_{e+1} = U_e$. If not, set $T_{e+1} = T_e$. Let $\sigma_{e+1} \succ \sigma_e$ in T_{e+1}.

Let $X = \bigcup_s \sigma_s$.

Low Basis Theorem

Theorem (Jockusch, Soare)

Every infinite computable binary branching tree has a low path.

Let T be an infinite computable binary branching tree.
We will use a $\varnothing'$-oracle to build a path through T by stages, while also at each stage restricting to a possibly smaller infinite computable binary branching tree of possible extensions.

Stage 0: Let σ_0 be the empty string and let $T_0 = T$.

Stage $e+1$: Consider the computable tree $U_e = \{\tau \succ \sigma_e \mid \tau \in T_e \wedge \Phi^{\tau}_{e,|\tau|}(e)\uparrow\}$. We use the $\varnothing'$ oracle to decide whether or not U_e is infinite. If U_e is infinite, set $T_{e+1} = U_e$. If not, set $T_{e+1} = T_e$. Let $\sigma_{e+1} \succ \sigma_e$ in T_{e+1}.

Let $X = \bigcup_s \sigma_s$.

Note that $e \in X'$ iff $\Phi^X_e(e)\downarrow$ iff U_e is finite. So $X' \leq_T \varnothing'$.

Reverse Mathematics

In Reverse Mathematics, the "difficulty" of various mathematical statements are compared by looking at what axioms of second order arithmetic are required to prove them.

Reverse Mathematics

In Reverse Mathematics, the "difficulty" of various mathematical statements are compared by looking at what axioms of second order arithmetic are required to prove them.

There are some natural groups of axioms, into which many common mathematical statements fall.

Reverse Mathematics

In Reverse Mathematics, the "difficulty" of various mathematical statements are compared by looking at what axioms of second order arithmetic are required to prove them.

There are some natural groups of axioms, into which many common mathematical statements fall.

For most of these axiom groups, there is a computability theory statement that must hold of any theorem that belongs to the group. So computability theory is often a useful tool for deciding the reverse mathematical strength of a theorem.

Infinitary formulas

Definition

Let τ be a countable language. The $\mathcal{L}_{\omega_1\omega}(\tau)$-formulas are defined inductively as follows:

- every atomic τ-formula is an $\mathcal{L}_{\omega_1\omega}(\tau)$-formula,
- if φ is an $\mathcal{L}_{\omega_1\omega}(\tau)$-formula, then so are $\neg\varphi$, $\exists x\varphi$ and $\forall x\varphi$,
- if $(\varphi_i)_{i\in\omega}$ are $\mathcal{L}_{\omega_1\omega}(\tau)$-formulas with finitely many free variables, then so are $\bigwedge_{i\in\omega}\varphi_i$ and $\bigvee_{i\in\omega}\varphi_i$.

Complexity of infinitary formulas

Each formula of $\mathcal{L}_{\omega_1\omega}$ is logically equivalent to a formula in normal form, and this can be seen as a measure of it's complexity.

Definition

- The Σ_0 and Π_0 formulas are the finitary open formulas,
- For $\alpha > 0$, if φ is a countable disjunction of formulas of the form $\exists u\, \psi$, where ψ is a Π_β formula for some $\beta < \alpha$, then φ is a Σ_α formula.
- For $\alpha > 0$, if φ is a countable conjunction of formulas of the form $\forall u\, \psi$, where ψ is a Σ_β formula for some $\beta < \alpha$, then φ is a Π_α formula.

Complexity of infinitary formulas

Each formula of $\mathcal{L}_{\omega_1\omega}$ is logically equivalent to a formula in normal form, and this can be seen as a measure of it's complexity.

Definition

- The Σ_0 and Π_0 formulas are the finitary open formulas,
- For $\alpha > 0$, if φ is a countable disjunction of formulas of the form $\exists u\, \psi$, where ψ is a Π_β formula for some $\beta < \alpha$, then φ is a Σ_α formula.
- For $\alpha > 0$, if φ is a countable conjunction of formulas of the form $\forall u\, \psi$, where ψ is a Σ_β formula for some $\beta < \alpha$, then φ is a Π_α formula.

Definition

A formula is said to be d-Σ_α if it is a conjunction of a Σ_α and a Π_α formula.

Scott's isomorphism theorem

Theorem (Scott)

For any countable structure $\mathcal{A}$ there is a sentence φ of $\mathcal{L}_{\omega_1\omega}$ that characterizes $\mathcal{A}$ up to isomorphism among countable structures, i.e., for all countable $\mathcal{B}$,

$$\mathcal{B} \models \varphi \iff \mathcal{A} \cong \mathcal{B}.$$

Scott's isomorphism theorem

Theorem (Scott)

For any countable structure $\mathcal{A}$ there is a sentence φ of $\mathcal{L}_{\omega_1\omega}$ that characterizes $\mathcal{A}$ up to isomorphism among countable structures, i.e., for all countable $\mathcal{B}$,

$$\mathcal{B} \models \varphi \iff \mathcal{A} \cong \mathcal{B}.$$

Definition

We call such a sentence a Scott sentence for $\mathcal{A}$.

Scott's isomorphism theorem

Theorem (Scott)

For any countable structure $\mathcal{A}$ there is a sentence φ of $\mathcal{L}_{\omega_1\omega}$ that characterizes $\mathcal{A}$ up to isomorphism among countable structures, i.e., for all countable $\mathcal{B}$,

$$\mathcal{B} \models \varphi \iff \mathcal{A} \cong \mathcal{B}.$$

Definition

We call such a sentence a Scott sentence for $\mathcal{A}$.

Definition

We call the least α for which $\mathcal{A}$ has a $\Pi_{\alpha+1}$ Scott sentence the Scott rank of $\mathcal{A}$.

Scott families

Definition

A Scott family for a structure $\mathcal{A}$ is a countable family Φ of formulas over a finite parameter such that

- for each $\bar{a} \in \mathcal{A}$, there exists $\varphi \in \Phi$ such that $\mathcal{A} \vDash \varphi(\bar{a})$
- if $\phi \in \Phi$, $\mathcal{A} \vDash \varphi(\bar{a})$, and $\mathcal{A} \vDash \varphi(\bar{b})$, then there is an automorphism of $\mathcal{A}$ taking $\bar{a}$ to $\bar{b}$.

Scott families

Definition

A Scott family for a structure $\mathcal{A}$ is a countable family Φ of formulas over a finite parameter such that

- for each $\bar{a} \in \mathcal{A}$, there exists $\varphi \in \Phi$ such that $\mathcal{A} \vDash \varphi(\bar{a})$
- if $\phi \in \Phi$, $\mathcal{A} \vDash \varphi(\bar{a})$, and $\mathcal{A} \vDash \varphi(\bar{b})$, then there is an automorphism of $\mathcal{A}$ taking $\bar{a}$ to $\bar{b}$.

Theorem (Scott)

Every countable structure has a Scott family. Moreover, there is some countable ordinal α such that every formula in the Scott family is Σ^0_α.

Back-and-forth relations

Definition (Back-and-forth relations)

Let $\mathcal{A}$ and $\mathcal{B}$ be countable structures in a finite language, and let $\vec{a}$ and $\vec{b}$ be a finite tuples of the same length from $\mathcal{A}$ and $\mathcal{B}$, respectively. For all ordinals α, define the back-and-forth relations, $\leq_\alpha$, inductively as follows:

Back-and-forth relations

Definition (Back-and-forth relations)

Let $\mathcal{A}$ and $\mathcal{B}$ be countable structures in a finite language, and let $\vec{a}$ and $\vec{b}$ be a finite tuples of the same length from $\mathcal{A}$ and $\mathcal{B}$, respectively. For all ordinals α, define the back-and-forth relations, $\leq_\alpha$, inductively as follows:

1. $(\mathcal{A}, \vec{a}) \leq_0 (\mathcal{B}, \vec{b})$ if and only if $\vec{a}$ and $\vec{b}$ satisfy the same atomic formulas in $\mathcal{A}$ and $\mathcal{B}$ respectively, and

Back-and-forth relations

Definition (Back-and-forth relations)

Let $\mathcal{A}$ and $\mathcal{B}$ be countable structures in a finite language, and let $\vec{a}$ and $\vec{b}$ be a finite tuples of the same length from $\mathcal{A}$ and $\mathcal{B}$, respectively. For all ordinals α, define the back-and-forth relations, $\leq_\alpha$, inductively as follows:

1. $(\mathcal{A}, \vec{a}) \leq_0 (\mathcal{B}, \vec{b})$ if and only if $\vec{a}$ and $\vec{b}$ satisfy the same atomic formulas in $\mathcal{A}$ and $\mathcal{B}$ respectively, and
2. for $\gamma \geq 1$, $(\mathcal{A}, \vec{a}) \leq_\gamma (\mathcal{B}, \vec{b})$ if and only if for each $\vec{d} \in \mathcal{B}$ and each $0 \leq \beta < \gamma$ there exists $\vec{c} \in \mathcal{A}$ such that $(\mathcal{B}, \vec{b}, \vec{d}) \leq_\beta (\mathcal{A}, \vec{a}, \vec{c})$, where $\vec{c}$ and $\vec{d}$ are of the same length.

Back-and-forth relations

Definition (Back-and-forth relations)

Let $\mathcal{A}$ and $\mathcal{B}$ be countable structures in a finite language, and let $\vec{a}$ and $\vec{b}$ be a finite tuples of the same length from $\mathcal{A}$ and $\mathcal{B}$, respectively. For all ordinals α, define the back-and-forth relations, $\leq_\alpha$, inductively as follows:

1. $(\mathcal{A}, \vec{a}) \leq_0 (\mathcal{B}, \vec{b})$ if and only if $\vec{a}$ and $\vec{b}$ satisfy the same atomic formulas in $\mathcal{A}$ and $\mathcal{B}$ respectively, and
2. for $\gamma \geq 1$, $(\mathcal{A}, \vec{a}) \leq_\gamma (\mathcal{B}, \vec{b})$ if and only if for each $\vec{d} \in \mathcal{B}$ and each $0 \leq \beta < \gamma$ there exists $\vec{c} \in \mathcal{A}$ such that $(\mathcal{B}, \vec{b}, \vec{d}) \leq_\beta (\mathcal{A}, \vec{a}, \vec{c})$, where $\vec{c}$ and $\vec{d}$ are of the same length.

Note that this definition includes the case where $\vec{a}$ and $\vec{b}$ are both the empty tuple. We denote $(\mathcal{A}, \varnothing) \leq_\alpha (\mathcal{B}, \varnothing)$ by $\mathcal{A} \leq_\alpha \mathcal{B}$.

Back-and-forth relations

Theorem (Ash and Knight)

Let $\mathcal{A}$ and $\mathcal{B}$ be structures in the same language and let $\vec{a}$ and $\vec{b}$ be tuples, from $\mathcal{A}$ and $\mathcal{B}$ respectively, with $|\vec{a}| = |\vec{b}|$. Then, for all ordinals α, the following are equivalent.

(i) $(\mathcal{A}, \vec{a}) \leq_\alpha (\mathcal{B}, \vec{b})$

(ii) *Every Σ_α formula true of $\vec{b}$ in $\mathcal{B}$ is also true of $\vec{a}$ in $\mathcal{A}$.*

(iii) *Every Π_α formula true of $\vec{a}$ in $\mathcal{A}$ is also true of $\vec{b}$ in $\mathcal{B}$.*

Computable Structures and Computable Infinitary Formulas

Definition

We say a structure $\mathcal{A}$ is computable if it has domain $\mathbb{N}$ and all functions and relations on $\mathcal{A}$ are computable.

Computable Structures and Computable Infinitary Formulas

Definition

We say a structure $\mathcal{A}$ is computable if it has domain $\mathbb{N}$ and all functions and relations on $\mathcal{A}$ are computable.

Definition

- The computable Σ_0 and computable Π_0 formulas are the finitary open formulas,
- For $\alpha > 0$, if φ is a disjunction of a computably enumerable set of formulas of the form $\exists u\, \psi$, where ψ is a computable Π_β formula for some $\beta < \alpha$, then φ is a computable Σ_α formula.
- For $\alpha > 0$, if φ is a conjunction of a computably enumerable set of formulas of the form $\forall u\, \psi$, where ψ is a computable Σ_β formula for some $\beta < \alpha$, then φ is a computable Π_α formula.

Computable Scott Complexity

If a computable structure has a Π_α Scott Sentence for some computable α, then following the proof of Scott's Isomorphism Theorem, one can see that it must also have a Scott Sentence that is computable $\Pi_{2\alpha}$. (Note that a computable structure is not guaranteed to have a Π_α Scott Sentence for some computable α - counterexamples exist.)

Computable Scott Complexity

If a computable structure has a Π_α Scott Sentence for some computable α, then following the proof of Scott's Isomorphism Theorem, one can see that it must also have a Scott Sentence that is computable $\Pi_{2\alpha}$. (Note that a computable structure is not guaranteed to have a Π_α Scott Sentence for some computable α - counterexamples exist.)

Theorem (Alvir, Knight, McCoy)

There is a computable structure with a Π_2 Scott Sentence but with no computable Π_2 Scott Sentence.

Computable Scott Complexity

If a computable structure has a Π_α Scott Sentence for some computable α, then following the proof of Scott's Isomorphism Theorem, one can see that it must also have a Scott Sentence that is computable $\Pi_{2\alpha}$. (Note that a computable structure is not guaranteed to have a Π_α Scott Sentence for some computable α - counterexamples exist.)

Theorem (Alvir, Knight, McCoy)

There is a computable structure with a Π_2 Scott Sentence but with no computable Π_2 Scott Sentence.

Theorem (Alvir, Csima, Harrison-Trainor)

There is a computable structure with a Π_2 Scott sentence but no computable Σ_4 Scott sentence.

Computable Scott Complexity

Using Marker extensions, we can extend our theorem to obtain the following:

Corollary (Alvir, Csima, Harrison-Trainor)

For each computable ordinal α there is a computable structure with a $\Pi_{\alpha+2}$ Scott sentence but with no computable $\Sigma_{\alpha+4}$ Scott sentence.

Computable Scott Complexity

Using Marker extensions, we can extend our theorem to obtain the following:

Corollary (Alvir, Csima, Harrison-Trainor)

For each computable ordinal α there is a computable structure with a $\Pi_{\alpha+2}$ Scott sentence but with no computable $\Sigma_{\alpha+4}$ Scott sentence.

More recently, using "unfriendly jump inversions", this has been improved:

Theorem (Harrison-Trainor)

For every n, there is a computable structure $\mathcal{A}$ with a Π_n Scott sentence but with no computable Σ_{2n} Scott sentence.

Algorithmic Randomness

Classically, we know that if an infinite binary sequence is randomly generated, then every infinite binary string has an equal probability of occurring.

Algorithmic Randomness

Classically, we know that if an infinite binary sequence is randomly generated, then every infinite binary string has an equal probability of occurring.

In our hearts we feel that the infinite string

00000000000000000000...

is less random than the string

11010001010111001010 01...

Algorithmic Randomness

Classically, we know that if an infinite binary sequence is randomly generated, then every infinite binary string has an equal probability of occurring.

In our hearts we feel that the infinite string

00000000000000000000...

is less random than the string

11010001010111001010 01...

In the area of Algorithmic Randomness, ideas from Kolmogorov complexity and effective probability theory come together to study the relative randomness of infinite binary sequences.

Want to learn more?

Ash, Chris J., and Julia Knight. *Computable structures and the hyperarithmetical hierarchy*. Vol. 144. Elsevier, 2000. Downey,

Rodney G., and Denis R. Hirschfeldt. *Algorithmic randomness and complexity*. Springer Science & Business Media, 2010. Hirschfeldt,

Denis R. *Slicing the truth: On the computable and reverse mathematics of combinatorial principles*. 2015.

Montalbán, Antonio. *Computable structure theory: Within the arithmetic*. Cambridge University Press, 2021.

Soare, Robert I. *Turing computability*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2016.

Stillwell, John. "Reverse mathematics: Proofs from the inside out." (2018): 1-200.

Thank You!