From Proofs to Programs, Graphs and Dynamics. Geometric perspectives on computational complexity

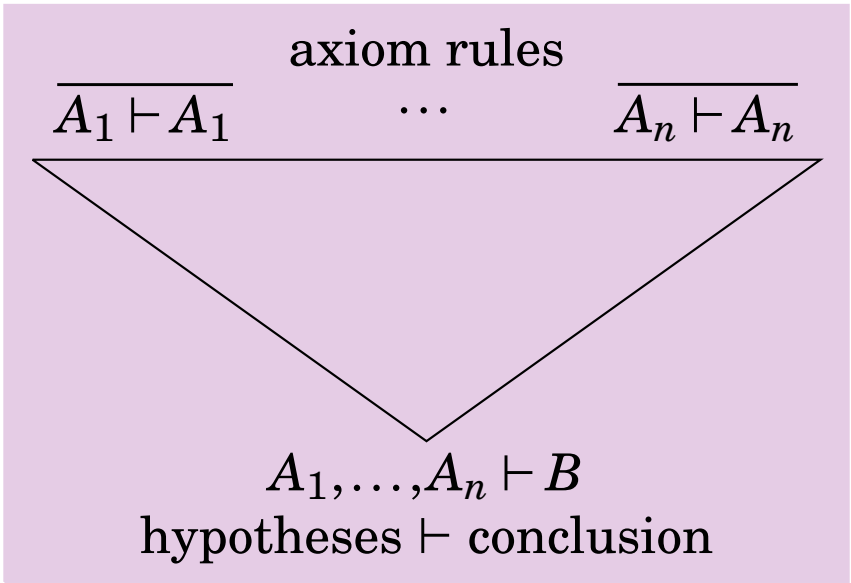
Thomas Seiller
CNRS, LIPN (Paris 13 Univ.)

Cologne-Twente Workshop 2018
June 18th 2018, Paris

- **Logic, Programs and Complexity**
- Graphings and Logic
- Formalising a Conjecture
- The Classification Problem and Barriers
- Entropy and Lower Bounds

Proof Theory

Proof are formalized as trees constructed from a set of rules.



Example

$$\dfrac{\dfrac{\dfrac{\vdots}{A \vdash B} \qquad \dfrac{\dfrac{}{A \vdash A}\,\text{ax} \qquad \dfrac{}{B \vdash B}\,\text{ax}}{A, B \vdash A \wedge B}\,\wedge}{A, A \vdash A \wedge B}\,\text{cut}}{A \vdash A \wedge B}\,\text{ctr}}{\vdash A \to A \wedge B}\,\to$$

Cut Elimination

Cut: using a lemma

$$\dfrac{\begin{array}{cc}\vdots & \vdots \\ \Gamma, A \vdash B & \Delta \vdash A\end{array}}{\Delta, \Gamma \vdash B}\ \text{cut}$$

Theorem (Gentzen)

If A is prouvable, there exists a cut-free proof of A.

The main interest of this result lies in its proof: a (local) procedure of cut-elimination is exhibited.

Proofs as Programs – Curry-Howard – correspondence

- **First Layer: Types | Formulas**

Proofs as Programs – Curry-Howard – correspondence

- **First Layer: Types ! Formulas**
  - Integers: $\mathrm{nat} := \forall X \ \ (X \to X) \to (X \to X)$

Proofs as Programs – Curry-Howard – correspondence

- **First Layer: Types | Formulas**
  - Integers: $\mathrm{nat} := \forall X \ \ (X \to X) \to (X \to X)$
  - Functions from integers to integers: $\mathrm{nat} \to \mathrm{nat}$

Proofs as Programs – Curry-Howard – correspondence

- **First Layer: Types ∣ Formulas**
  - Integers: $\mathrm{nat} := \forall X \ \ (X \to X) \to (X \to X)$
  - Functions from integers to integers: $\mathrm{nat} \to \mathrm{nat}$
- **Second Layer: Programs ∣ Proofs**

Proofs as Programs – Curry-Howard – correspondence

- **First Layer: Types ! Formulas**
  - Integers: $\mathrm{nat} := \forall X \ \ (X \to X) \to (X \to X)$
  - Functions from integers to integers: $\mathrm{nat} \to \mathrm{nat}$
- **Second Layer: Programs ! Proofs**
  - (Cut-free) Proofs of type nat are exactly "Church integers" (iterators $f \mapsto f^k$).

Proofs as Programs – Curry-Howard – correspondence

- **First Layer: Types ! Formulas**
  - Integers: $\mathrm{nat} := \forall X \ \ (X \to X) \to (X \to X)$
  - Functions from integers to integers: $\mathrm{nat} \to \mathrm{nat}$
- **Second Layer: Programs ! Proofs**
  - (Cut-free) Proofs of type nat are exactly "Church integers" (iterators $f \mapsto f^k$).
- **Third Layer: Computation ! Normalisation**

Proofs as Programs – Curry-Howard – correspondence

- **First Layer: Types ! Formulas**
  - Integers: $\mathrm{nat} := \forall X \ \ (X \to X) \to (X \to X)$
  - Functions from integers to integers: $\mathrm{nat} \to \mathrm{nat}$
- **Second Layer: Programs ! Proofs**
  - (Cut-free) Proofs of type nat are exactly "Church integers" (iterators $f \mapsto f^k$).
- **Third Layer: Computation ! Normalisation**
  - Given $[n]$ and $[f]$, proofs of nat and $\mathrm{nat} \to \mathrm{nat}$ respectively, we can define the proof $\mathrm{cut}([f],[n])$:

$$\dfrac{\begin{array}{cc}[f] & [n] \\ \vdots & \vdots \\ \mathrm{nat} \vdash \mathrm{nat} & \vdash \mathrm{nat}\end{array}}{\vdash \mathrm{nat}}\ \mathrm{cut}$$

Proofs as Programs – Curry-Howard – correspondence

- **First Layer: Types ! Formulas**
  - Integers: $\mathrm{nat} := \forall X \ \ (X \to X) \to (X \to X)$
  - Functions from integers to integers: $\mathrm{nat} \to \mathrm{nat}$
- **Second Layer: Programs ! Proofs**
  - (Cut-free) Proofs of type nat are exactly "Church integers" (iterators $f \mapsto f^k$).
- **Third Layer: Computation ! Normalisation**
  - Given $[n]$ and $[f]$, proofs of nat and $\mathrm{nat} \to \mathrm{nat}$ respectively, we can define the proof $\mathrm{cut}([f],[n])$:

$$\dfrac{\begin{array}{cc}[f] & [n] \\ \vdots & \vdots \\ \mathrm{nat} \vdash \mathrm{nat} & \vdash \mathrm{nat}\end{array}}{\vdash \mathrm{nat}}\ \mathrm{cut}$$

  - The cut elimination procedure applied to $[f][n]$ corresponds (step by step) to the computation of $f(n)$. The cut-free proof it produces is equal to $[f(n)]$.

$$\dfrac{\begin{array}{cc}[f] & [n] \\ \vdots & \vdots \\ \mathrm{nat} \vdash \mathrm{nat} & \vdash \mathrm{nat}\end{array}}{\vdash \mathrm{nat}}\ \mathrm{cut} \quad \to * \quad \begin{array}{c}[f(n)] \\ \vdots \\ \vdash \mathrm{nat}\end{array}$$

Fourth Layer: Geometry of Interaction

Geometry of Interaction is a mathematical model of proofs' dynamics.

Logic	Computer Science	
Formulas	Types	
Proof $\pi \vdash \mathbf{Nat} \Rightarrow \mathbf{Nat}$	Program $f : \mathrm{nat} \to \mathrm{nat}$	
Proof $\rho \vdash \mathbf{Nat}$	Data $n : \mathrm{nat}$	
Cut Rule $\mathrm{cut}(\pi, \rho)$	Application $f(n)$	
Cut elimination $\mathrm{cut}(\pi, \rho) \rightsquigarrow \mu \vdash \mathbf{Nat}$	Computation $f(n) \rightsquigarrow m : \mathrm{nat}$	
(Linear) Negation	Test	

Fourth Layer: Geometry of Interaction

Geometry of Interaction is a mathematical model of proofs' dynamics.

Logic	Computer Science	GoI
Formulas	Types	
Proof $\pi \vdash \mathbf{Nat} \Rightarrow \mathbf{Nat}$	Program $f : \mathrm{nat} \to \mathrm{nat}$	
Proof $\rho \vdash \mathbf{Nat}$	Data $n : \mathrm{nat}$	
Cut Rule $\mathrm{cut}(\pi, \rho)$	Application $f(n)$	
Cut elimination $\mathrm{cut}(\pi, \rho) \rightsquigarrow \mu \vdash \mathbf{Nat}$	Computation $f(n) \rightsquigarrow m : \mathrm{nat}$	
(Linear) Negation	Test	

Fourth Layer: Geometry of Interaction

Geometry of Interaction is a mathematical model of proofs' dynamics.

Logic	Computer Science	GoI
Formulas	Types	
Proof $\pi \vdash \mathbf{Nat} \Rightarrow \mathbf{Nat}$	Program $f : \mathrm{nat} \to \mathrm{nat}$	Operator $F \in \mathscr{L}(\mathbb{H} \oplus \mathbb{H})$
Proof $\rho \vdash \mathbf{Nat}$	Data $n : \mathrm{nat}$	Operator $N \in \mathscr{L}(\mathbb{H})$
Cut Rule $\mathrm{cut}(\pi, \rho)$	Application $f(n)$	
Cut elimination $\mathrm{cut}(\pi, \rho) \rightsquigarrow \mu \vdash \mathbf{Nat}$	Computation $f(n) \rightsquigarrow m : \mathrm{nat}$	
(Linear) Negation	Test	

Fourth Layer: Geometry of Interaction

Geometry of Interaction is a mathematical model of proofs' dynamics.

Logic	Computer Science	GoI
Formulas	Types	
Proof $\pi \vdash \mathbf{Nat} \Rightarrow \mathbf{Nat}$	Program $f : \mathrm{nat} \to \mathrm{nat}$	Operator $F \in \mathscr{L}(\mathbb{H} \oplus \mathbb{H})$
Proof $\rho \vdash \mathbf{Nat}$	Data $n : \mathrm{nat}$	Operator $N \in \mathscr{L}(\mathbb{H})$
Cut Rule $\mathrm{cut}(\pi, \rho)$	Application $f(n)$	Functional Equation $\begin{cases} F(x \oplus y) = x' \oplus y' \\ N(x') = x \end{cases}$
Cut elimination $\mathrm{cut}(\pi, \rho) \rightsquigarrow \mu \vdash \mathbf{Nat}$	Computation $f(n) \rightsquigarrow m : \mathrm{nat}$	
(Linear) Negation	Test	

Fourth Layer: Geometry of Interaction

Geometry of Interaction is a mathematical model of proofs' dynamics.

Logic	Computer Science	GoI
Formulas	Types	
Proof $\pi \vdash \mathbf{Nat} \Rightarrow \mathbf{Nat}$	Program $f : \mathrm{nat} \to \mathrm{nat}$	Operator $F \in \mathscr{L}(\mathbb{H} \oplus \mathbb{H})$
Proof $\rho \vdash \mathbf{Nat}$	Data $n : \mathrm{nat}$	Operator $N \in \mathscr{L}(\mathbb{H})$
Cut Rule $\mathrm{cut}(\pi, \rho)$	Application $f(n)$	Functional Equation $\begin{cases} F(x \oplus y) = x' \oplus y' \\ N(x') = x \end{cases}$
Cut elimination $\mathrm{cut}(\pi, \rho) \rightsquigarrow \mu \vdash \mathbf{Nat}$	Computation $f(n) \rightsquigarrow m : \mathrm{nat}$	Construction of a solution $\mathrm{Ex}(F, A)(y) = y' \in \mathscr{L}(\mathbb{H})$
(Linear) Negation	Test	

Fourth Layer: Geometry of Interaction

Geometry of Interaction is a mathematical model of proofs' dynamics.

Logic	Computer Science	GoI
Formulas	Types	
Proof $\pi \vdash \mathbf{Nat} \Rightarrow \mathbf{Nat}$	Program $f : \mathrm{nat} \to \mathrm{nat}$	Operator $F \in \mathscr{L}(\mathbb{H} \oplus \mathbb{H})$
Proof $\rho \vdash \mathbf{Nat}$	Data $n : \mathrm{nat}$	Operator $N \in \mathscr{L}(\mathbb{H})$
Cut Rule $\mathrm{cut}(\pi, \rho)$	Application $f(n)$	Functional Equation $\begin{cases} F(x \oplus y) = x' \oplus y' \\ N(x') = x \end{cases}$
Cut elimination $\mathrm{cut}(\pi, \rho) \rightsquigarrow \mu \vdash \mathbf{Nat}$	Computation $f(n) \rightsquigarrow m : \mathrm{nat}$	Construction of a solution $\mathrm{Ex}(F, A)(y) = y' \in \mathscr{L}(\mathbb{H})$
(Linear) Negation	Test	Orthogonality $\perp\!\!\!\perp$

Fourth Layer: Geometry of Interaction

Geometry of Interaction is a mathematical model of proofs' dynamics.

Logic	Computer Science	GoI
Formulas	Types	Sets of Operators
Proof $\pi \vdash \mathbf{Nat} \Rightarrow \mathbf{Nat}$	Program $f : \mathrm{nat} \to \mathrm{nat}$	Operator $F \in \mathscr{L}(\mathbb{H} \oplus \mathbb{H})$
Proof $\rho \vdash \mathbf{Nat}$	Data $n : \mathrm{nat}$	Operator $N \in \mathscr{L}(\mathbb{H})$
Cut Rule $\mathrm{cut}(\pi, \rho)$	Application $f(n)$	Functional Equation $\begin{cases} F(x \oplus y) = x' \oplus y' \\ N(x') = x \end{cases}$
Cut elimination $\mathrm{cut}(\pi, \rho) \rightsquigarrow \mu \vdash \mathbf{Nat}$	Computation $f(n) \rightsquigarrow m : \mathrm{nat}$	Construction of a solution $\mathrm{Ex}(F, A)(y) = y' \in \mathscr{L}(\mathbb{H})$
(Linear) Negation	Test	Orthogonality $\perp\!\!\!\perp$

E.g. $\mathrm{nat} \to \mathrm{nat}$ is defined as $\{f \mid \forall n \in \mathrm{nat}, \forall t \in \mathrm{nat}^{\perp\!\!\!\perp}, \mathrm{Ex}(f, n) \perp\!\!\!\perp t\}$.

Logic and Complexity by Layers

- **1st Layer: Types | Formulas**: Descriptive complexity

Informally

Descriptive Complexity (DC) studies types of logics whose individual sentences characterise exactly particular complexity classes.

- **2nd Layer: Programs | Proofs**: Implicit complexity

- **3rd Layer: Computation | Normalisation**: Constrained Linear Logic

Logic and Complexity by Layers

- **1st Layer: Types | Formulas**: Descriptive complexity

Example Theorem (Fagin 74)

The class NPTIME *is the set of problems definable by existential second order boolean formulas.*

- **2nd Layer: Programs | Proofs**: Implicit complexity

- **3rd Layer: Computation | Normalisation**: Constrained Linear Logic

Logic and Complexity by Layers

- **1st Layer: Types | Formulas**: Descriptive complexity

Example Theorem (Fagin 74)

The class NPTIME *is the set of problems definable by existential second order boolean formulas.*

- **2nd Layer: Programs | Proofs**: Implicit complexity

Informally

Implicit Computational Complexity (ICC) studies algorithmic complexity only in terms of restrictions of languages and computational principles, for instance considering restrictions on recursion schemes.

- **3rd Layer: Computation | Normalisation**: Constrained Linear Logic

Logic and Complexity by Layers

- **1st Layer: Types | Formulas**: Descriptive complexity

Example Theorem (Fagin 74)

The class NPTIME *is the set of problems definable by existential second order boolean formulas.*

- **2nd Layer: Programs | Proofs**: Implicit complexity

Example Theorem (Bellantoni and Cook 92)

The class FPTIME *is the smallest class of functions containing constants, projections, successors, predecessor and conditional (if* $a \mod 2 = 0$*), and closed under Predicative Recursion on Notation and Safe Composition.*

- **3rd Layer: Computation | Normalisation**: Constrained Linear Logic

Logic and Complexity by Layers

- **1st Layer: Types | Formulas**: Descriptive complexity

Example Theorem (Fagin 74)

The class NPTIME *is the set of problems definable by existential second order boolean formulas.*

- **2nd Layer: Programs | Proofs**: Implicit complexity

Example Theorem (Bellantoni and Cook 92)

The class FPTIME *is the smallest class of functions containing constants, projections, successors, predecessor and conditional (if* $a \mod 2 = 0$*), and closed under Predicative Recursion on Notation and Safe Composition.*

- **3rd Layer: Computation | Normalisation**: Constrained Linear Logic

Informally

Constrained Linear Logic studies restrictions of proof systems in which the cut-elimination procedure has a bounded complexity.

Logic and Complexity by Layers

- **1st Layer: Types | Formulas**: Descriptive complexity

Example Theorem (Fagin 74)

The class NPTIME *is the set of problems definable by existential second order boolean formulas.*

- **2nd Layer: Programs | Proofs**: Implicit complexity

Example Theorem (Bellantoni and Cook 92)

The class FPTIME *is the smallest class of functions containing constants, projections, successors, predecessor and conditional (if* $a \mod 2 = 0$*), and closed under Predicative Recursion on Notation and Safe Composition.*

- **3rd Layer: Computation | Normalisation**: Constrained Linear Logic

Example Theorem (Girard, Scedrov and Scott 92)

Proofs of $\mathrm{nat} \to \mathrm{nat}$ *in Bounded Linear Logic compute exactly the polynomial-time computable functions* FPTIME.

Fourth Layer: Geometry of Interaction

Theorem (Girard '06)

Let $a, b \in \mathscr{L}(\mathbb{H})$ be operators with norm $\leqslant 1$. Then $\mathrm{Ex}(a,b)$ exists, is unique, and lies in the unit ball of the von Neumann algebra generated by a and b.

(unit ball of) a von Neumann algebra = set of (untyped) programs

Fourth Layer: Geometry of Interaction

Theorem (Girard '06)

Let $a, b \in \mathscr{L}(\mathbb{H})$ be operators with norm $\leqslant 1$. Then $\mathrm{Ex}(a,b)$ *exists, is unique, and lies in the unit ball of the von Neumann algebra generated by a and b.*

(unit ball of) a von Neumann algebra = set of (untyped) programs

Conjecture

Different von Neumann algebras = different degrees of expressivity.

Fourth Layer: Geometry of Interaction

Theorem (Girard '06)

Let $a, b \in \mathscr{L}(\mathbb{H})$ be operators with norm $\leqslant 1$. Then $\mathrm{Ex}(a,b)$ exists, is unique, and lies in the unit ball of the von Neumann algebra generated by a and b.

(unit ball of) a von Neumann algebra = set of (untyped) programs

Conjecture

Different von Neumann algebras = different degrees of expressivity.

This wild idea rests upon the following results:

- $\mathscr{L}(\mathbb{H})$ models pure lambda-calculus (Turing-complete);
- the type II_∞ hyperfinite factor $\mathscr{R}_{0,1}$ models ELEM;
- a sub-algebra of $\mathscr{R}_{0,1}$ characterises LOGSPACE (w/ Aubert);

Fourth Layer: Geometry of Interaction

Theorem (Girard '06)

Let $a, b \in \mathscr{L}(\mathbb{H})$ be operators with norm $\leqslant 1$. Then $\mathrm{Ex}(a,b)$ exists, is unique, and lies in the unit ball of the von Neumann algebra generated by a and b.

(unit ball of) a von Neumann algebra = set of (untyped) programs

Conjecture

Different von Neumann algebras = different degrees of expressivity.

This wild idea rests upon the following results:

- $\mathscr{L}(\mathbb{H})$ models pure lambda-calculus (Turing-complete);
- the type II_∞ hyperfinite factor $\mathscr{R}_{0,1}$ models ELEM;
- a sub-algebra of $\mathscr{R}_{0,1}$ characterises LOGSPACE (w/ Aubert);

Correction: fix a von Neumann algebra *and* a maximal abelian sub-algebra (masa); expressivity then coincides with Dixmier's classification of masas.

Fourth Layer: Geometry of Interaction

Theorem (Girard '06)

Let $a, b \in \mathscr{L}(\mathbb{H})$ be operators with norm $\leqslant 1$. Then $\mathrm{Ex}(a,b)$ exists, is unique, and lies in the unit ball of the von Neumann algebra generated by a and b.

(unit ball of) a von Neumann algebra = set of (untyped) programs

Conjecture

Different pairs $\mathfrak{A} \subset \mathfrak{N}$ = different degrees of expressivity.

This wild idea rests upon the following results:

- $\mathscr{L}(\mathbb{H})$ models pure lambda-calculus (Turing-complete);
- the type II_∞ hyperfinite factor $\mathscr{R}_{0,1}$ models ELEM;
- a sub-algebra of $\mathscr{R}_{0,1}$ characterises LOGSPACE (w/ Aubert);

Correction: fix a von Neumann algebra *and* a maximal abelian sub-algebra (masa); expressivity then coincides with Dixmier's classification of masas.

- Logic, Programs and Complexity
- **Graphings and Logic**
- Formalising a Conjecture
- The Classification Problem and Barriers
- Entropy and Lower Bounds

Towards a logic of programs

Informal Definition

A program is a dynamical process possibly involving exchange/duplication/erasure/modification of information.

[Complexity] Implicit Computational Complexity.

Size-change termination (Lee, Jones, Ben-Amram), mwp-polynomials (Jones, Kristiansen), Loop peeling (Moyen, Rubiano, Seiller).

[Semantics] Dynamic Semantics

Geometry of Interaction (Girard), Game Semantics (Abramsky/Jagadeesan/Malacaria, Hyland/Ong), Interaction Graphs (Seiller).

Towards a logic of programs

Informal Definition

A program is a dynamical process possibly involving exchange/duplication/erasure/modification of information.

[Complexity] Implicit Computational Complexity.
Size-change termination (Lee, Jones, Ben-Amram), mwp-polynomials (Jones, Kristiansen), Loop peeling (Moyen, Rubiano, Seiller).

[Semantics] Dynamic Semantics
Geometry of Interaction (Girard), Game Semantics (Abramsky/Jagadeesan/Malacaria, Hyland/Ong), Interaction Graphs (Seiller).

[Compilation] Compilation techniques.
Work by U. Schöpp (cf. Habilitation thesis), Loop peeling (Moyen, Rubiano, Seiller)

[VLSI design] Synthesis methods for VLSI design.
Geometry of Synthesis programme (Ghica).

Abstract Programs

To make the conjecture more tractable, we consider concrete pairs $\mathfrak{A} \subset \mathfrak{N}$, based on the group measure space construction of Murray and von Neumann:

- Consider a group G acting on a measure space X, then:
  - X induces a hilbert space $L^2(X)$;
  - G induces unitaries acting on $L^2(X)$ generating an algebra $\mathfrak{N}$;
  - X induces a maximal abelian subalgebra $\mathfrak{A} = L^\infty(X)$.

Note: The following setting generalises the construction above.

Abstract Programs

To make the conjecture more tractable, we consider concrete pairs $\mathfrak{A} \subset \mathfrak{N}$, based on the group measure space construction of Murray and von Neumann:

- Consider a group G acting on a measure space X, then:
  - X induces a hilbert space $L^2(X)$;
  - G induces unitaries acting on $L^2(X)$ generating an algebra $\mathfrak{N}$;
  - X induces a maximal abelian subalgebra $\mathfrak{A} = L^\infty(X)$.

Note: The following setting generalises the construction above.

Definition

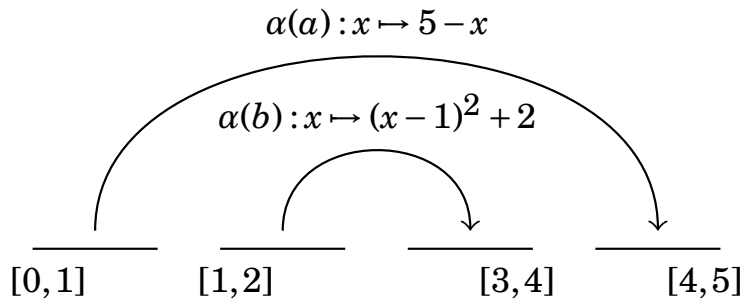
An *abstract model of computation* (AMC) is defined as a monoid action $\alpha : M \curvearrowright \mathbf{X}$ of a monoid $M = \mathrm{M}\langle G, \mathrm{R}\rangle$ on a space $\mathbf{X}$. I.e. a morphism $\alpha : M \to \mathrm{End}(\mathbf{X})$.

Definition

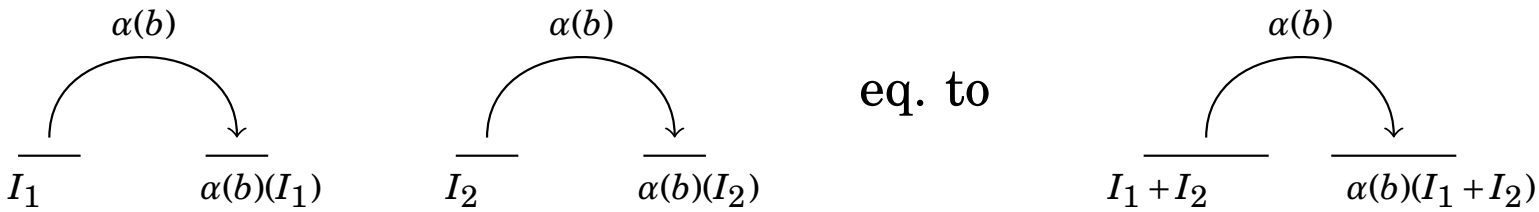
An *abstract program* is a α-graphing.

What's a graphing?

- (Multi)Graph = Collection of edges.
- Graphing = Collection of *realised edges*.
- Replace vertices by subspaces of the underlying space $\mathbf{X}$.
- Decide *how* (i.e. pick an element of M) the edges map sources to targets.



- Then quotient the set of such objects w.r.t. *refinement*:



A bit more on graphings

Graphings were introduced by Adams in the context of ergodic theory. They also appear as certain limits of graphs (cf. Aldous-Lyons conjecture).

A bit more on graphings

Graphings were introduced by Adams in the context of ergodic theory. They also appear as certain limits of graphs (cf. Aldous-Lyons conjecture).

Definition

Given an AMC $\alpha : M \curvearrowright \mathbf{X}$, an α-graphing is a collection of pairs (S, m) where $S \subset \mathbf{X}$ and $m \in M$.

A bit more on graphings

Graphings were introduced by Adams in the context of ergodic theory. They also appear as certain limits of graphs (cf. Aldous-Lyons conjecture).

Definition

Given an AMC $\alpha : M \curvearrowright \mathbf{X}$, an α-graphing is a collection of pairs (S, m) where $S \subset \mathbf{X}$ and $m \in M$.

One can consider restrictions of graphings:

- Discrete space: multigraphs;
- Deterministic Graphings: Dynamical Systems
- Probabilistic Graphings: Markov processes*

Execution

The functional equation:

$$\begin{cases} F(x \oplus y) & = & x' \oplus y' \\ N(x') & = & x \end{cases}$$

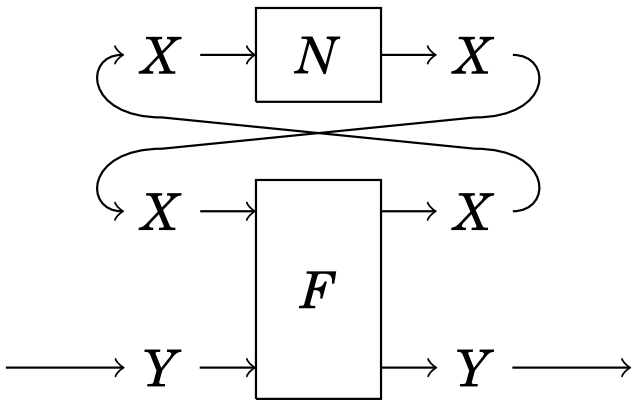
can be expressed diagrammatically:

Execution

The functional equation:

$$\begin{cases} F(x \oplus y) & = & x' \oplus y' \\ N(x') & = & x \end{cases}$$

can be expressed diagrammatically:

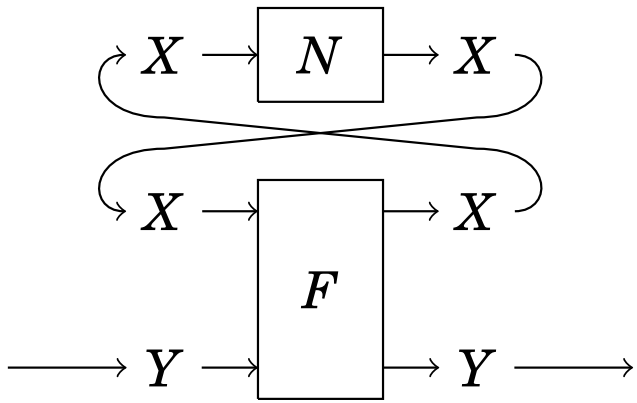


Execution

The functional equation:

$$\begin{cases} F(x \oplus y) & = & x' \oplus y' \\ N(x') & = & x \end{cases}$$

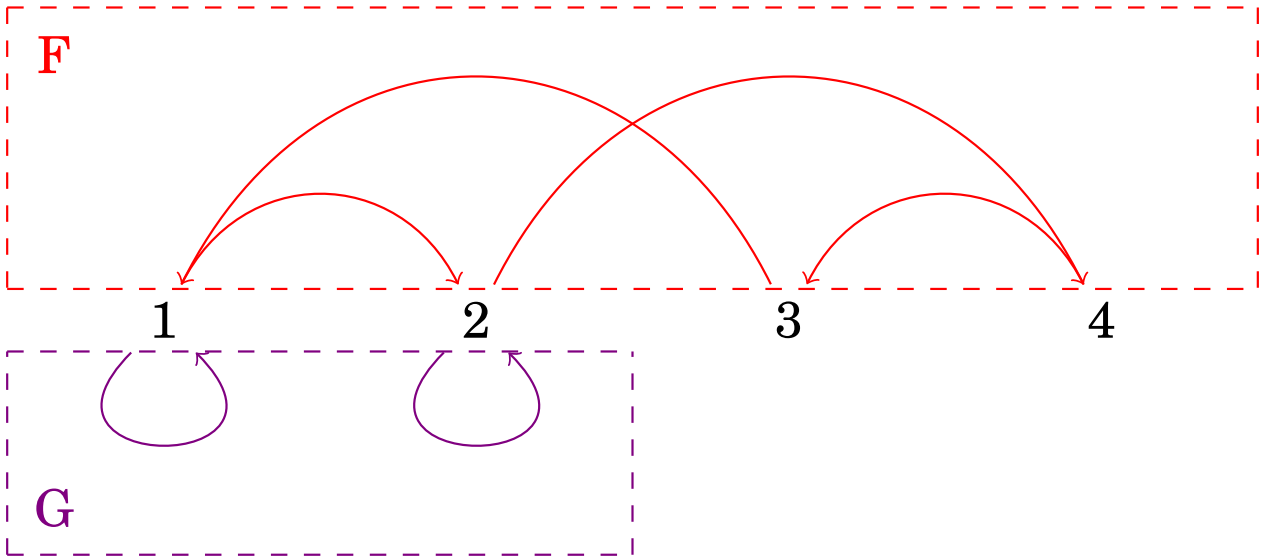
can be expressed diagrammatically:



A solution $F :: N : Y \to Y$ is then computed as a fixpoint.

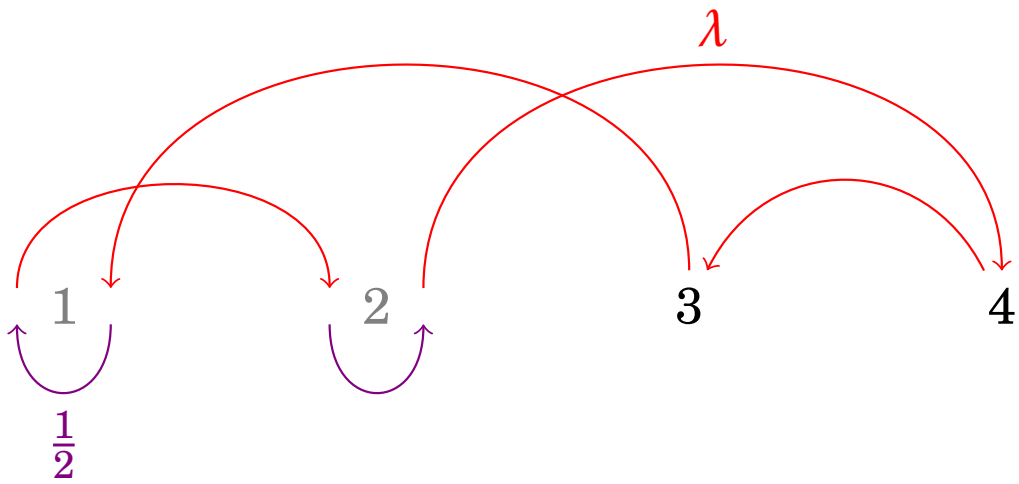
Execution as Paths

The execution $F::G$ of two graphs F, G is the graph of alternating paths of source and target in $V^F \Delta V^G$.



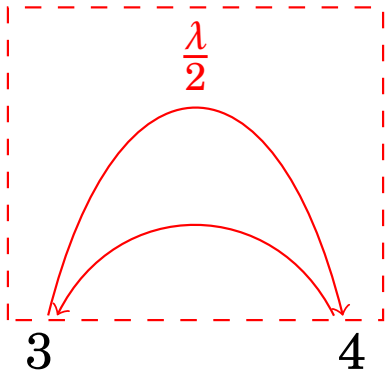
Execution as Paths

The execution $F :: G$ of two graphs F, G is the graph of alternating paths of source and target in $V^F \Delta V^G$.



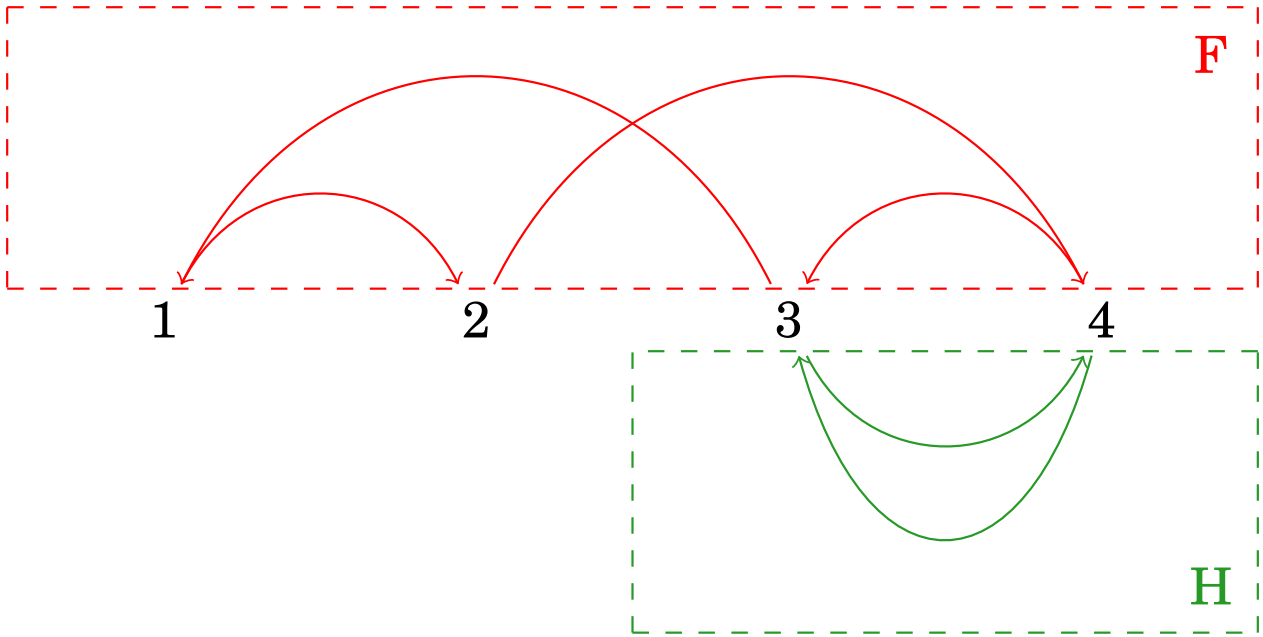
Execution as Paths

The execution $F::G$ of two graphs F, G is the graph of alternating paths of source and target in $V^F \Delta V^G$.



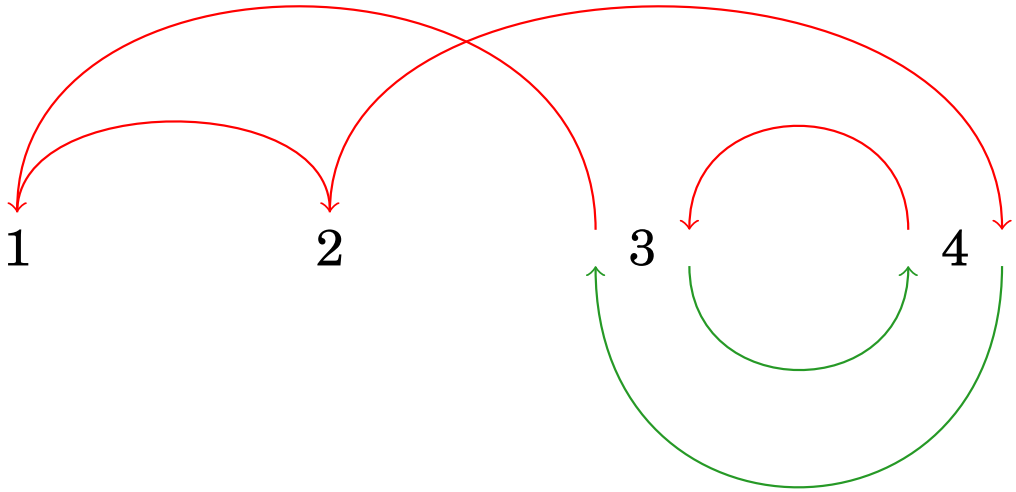
Cycles

In some cases, cycles appear during this operation.



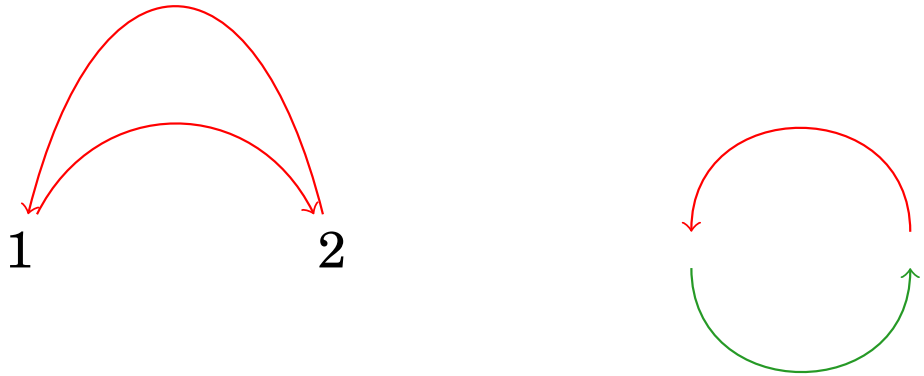
Cycles

In some cases, cycles appear during this operation.



Cycles

In some cases, cycles appear during this operation.

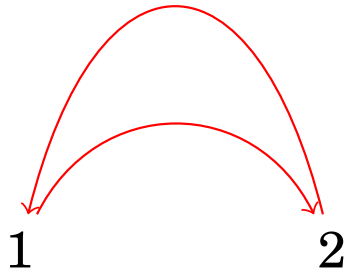


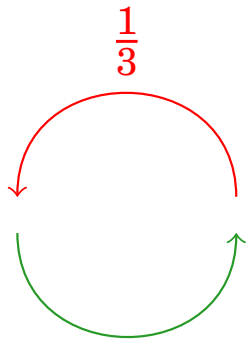
Remark

To define types, one needs to decide what these cycles mean.

Cycles

In some cases, cycles appear during this operation.





Remark

To define types, one needs to decide what these cycles mean.

Orthogonality and Zeta

- Orthogonality in IG: defined by *measuring* cycles.

$$[\![F,G]\!]_m = \sum_{\pi \in \mathscr{C}(F,G)} m(\pi)$$

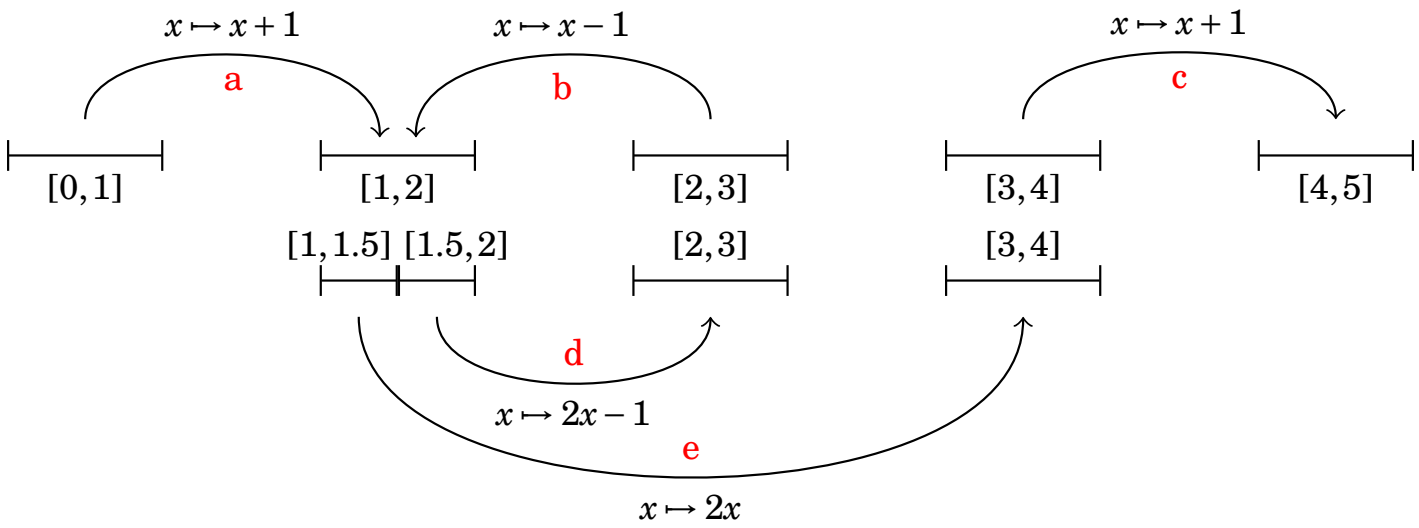
- This is related to Zeta functions of graphs (Ihara):

$$\zeta_G(z) = \prod_{\pi \in \mathscr{C}(G)} (1 - z^{\omega(\pi)})^{-1}$$

- The following property insures that types are defined properly:

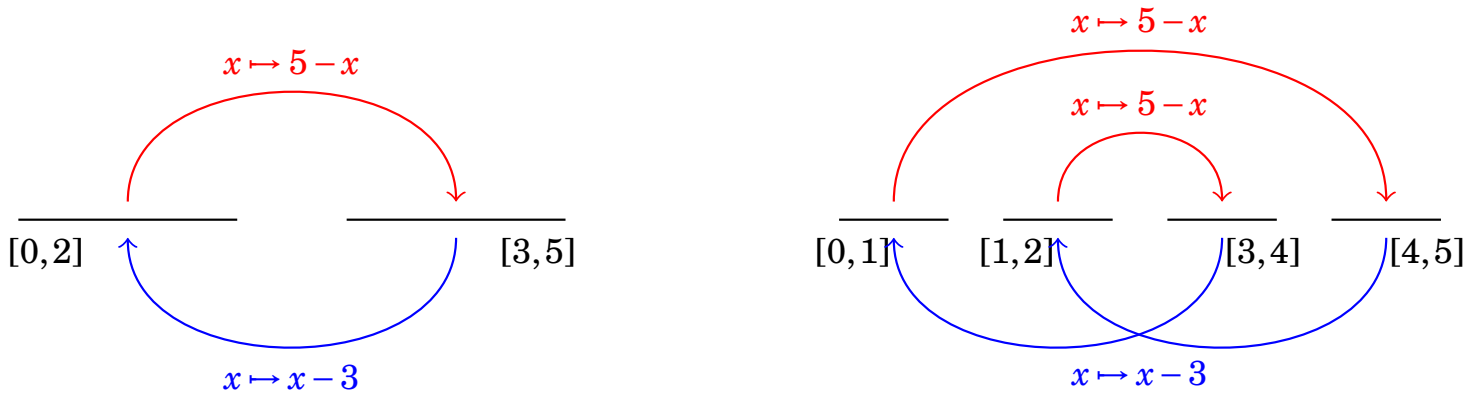
$$\zeta_{F \circ (G+H)}(z) = \zeta_{F \circ G}(z) \zeta_{(F::G) \circ H}(z)$$

Execution in Graphings

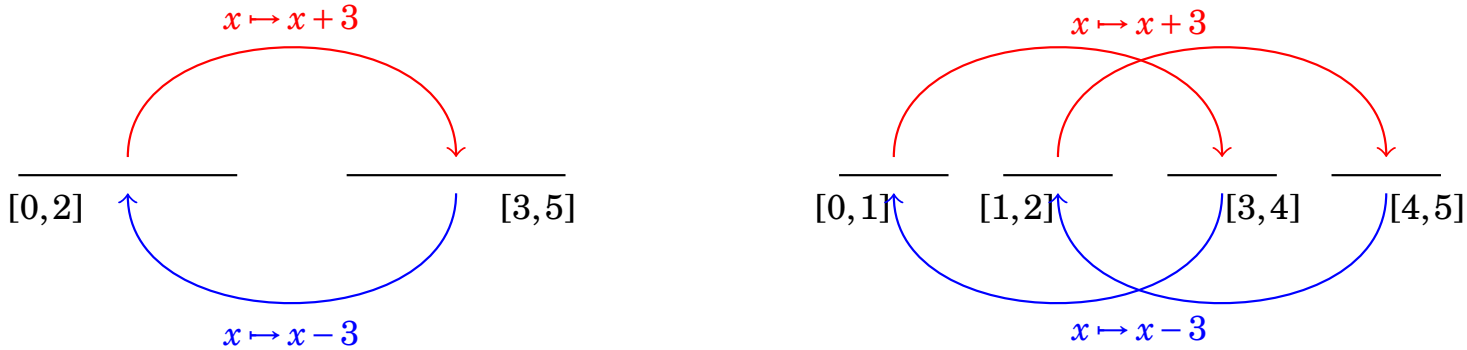


Here the result of the execution is the graphing defined by the following set of edges: $\{a(db)^k ec\}_{k=0}^{\infty}$.

Cycles in Graphings



(a) A cycle can become a longer cycle



(b) A cycle can become two (or more) cycles

Figure: Evolution of cycles through refinement

Hierarchies of models

Theorem (Seiller, APAL 2017)

For every AMC α*, the set of* α*-graphings defines a non-degenerate model of Multiplicative-Additive Linear Logic.*

Hierarchies of models

Theorem (Seiller, APAL 2017)

For every AMC α*, the set of* α*-graphings defines a non-degenerate model of Multiplicative-Additive Linear Logic.*

Back to complexity:

- Within these models, consider the set of programs of type nat $\to$ bool;
- Larger AMC implies more programs, i.e. more expressivity;
- Find some AMC characterising complexity classes.

- Logic, Programs and Complexity
- Graphings and Logic
- **Formalising a Conjecture**
- The Classification Problem and Barriers
- Entropy and Lower Bounds

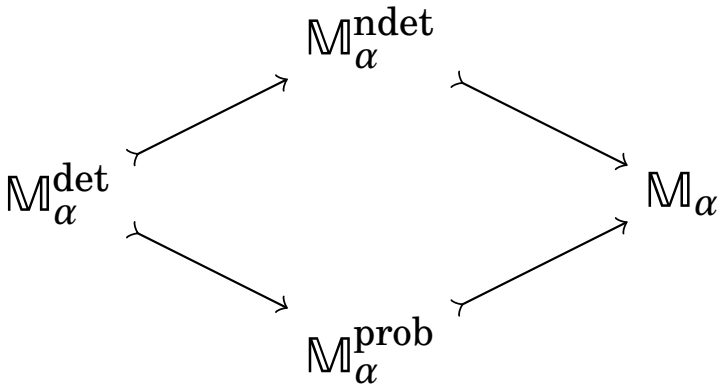
Some Results

AMC	det. model	non-det. model		prob. model
α_1	REGULAR	REGULAR	REGULAR	STOCHASTIC
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$
α_k	D_k	N_k	co-N_k	P_k
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$
α_∞	LOGSPACE	NLOGSPACE	coNLOGSPACE	PLOGSPACE
β	PTIME	PTIME	PTIME	PTIME?
γ	PTIME	NPTIME	coNPTIME	PP?

- Only known correspondence between infinite hierarchies of mathematical objects and complexity classes.
- Indicates a strong connection between *geometry* and complexity: cf. AMC generalise *group actions*, use of (generalised) Zeta functions, (homotopy) equivalence between microcosms implies equality of the classes.

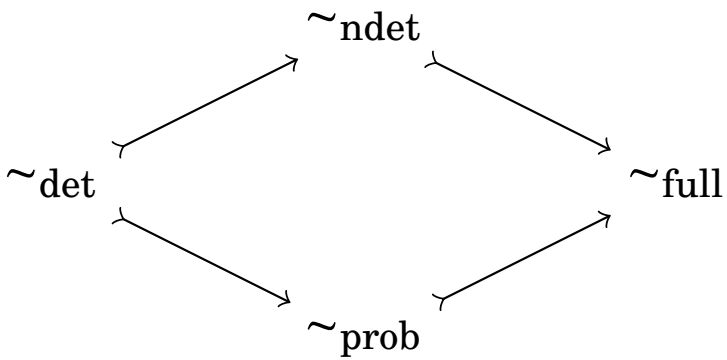
Some Results

AMC	det. model	non-det. model		prob. model
α_1	REGULAR	REGULAR	REGULAR	STOCHASTIC
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$
α_k	D_k	N_k	co-N_k	P_k
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$
α_∞	LOGSPACE	NLOGSPACE	coNLOGSPACE	PLOGSPACE
β	PTIME	PTIME	PTIME	PTIME?
γ	PTIME	NPTIME	coNPTIME	PP?



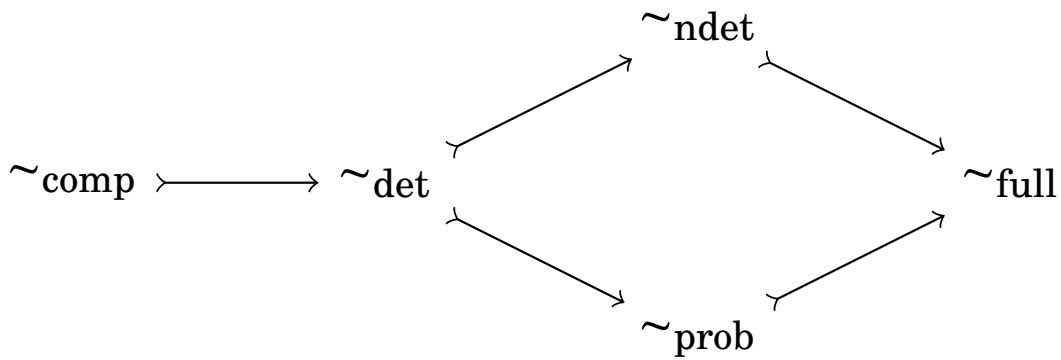
Some Results

AMC	det. model	non-det. model		prob. model
α_1	REGULAR	REGULAR	REGULAR	STOCHASTIC
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$
α_k	D_k	N_k	co-N_k	P_k
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$
α_∞	LOGSPACE	NLOGSPACE	coNLOGSPACE	PLOGSPACE
β	PTIME	PTIME	PTIME	PTIME?
γ	PTIME	NPTIME	coNPTIME	PP?



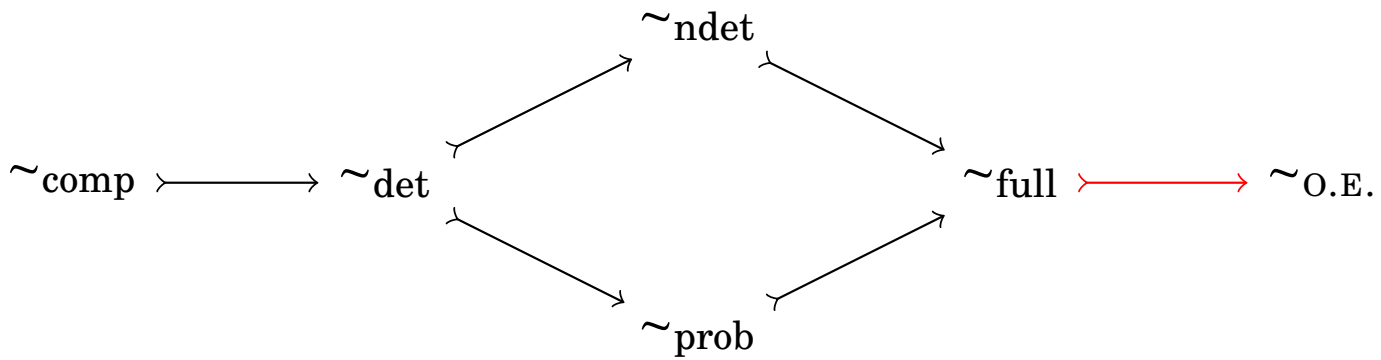
Some Results

AMC	det. model	non-det. model		prob. model
α_1	REGULAR	REGULAR	REGULAR	STOCHASTIC
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$
α_k	D_k	N_k	co-N_k	P_k
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$
α_∞	LOGSPACE	NLOGSPACE	coNLOGSPACE	PLOGSPACE
β	PTIME	PTIME	PTIME	PTIME?
γ	PTIME	NPTIME	coNPTIME	PP?



Some Results

AMC	det. model	non-det. model		prob. model
α_1	REGULAR	REGULAR	REGULAR	STOCHASTIC
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$
α_k	D_k	N_k	co-N_k	P_k
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$
α_∞	LOGSPACE	NLOGSPACE	coNLOGSPACE	PLOGSPACE
β	PTIME	PTIME	PTIME	PTIME?
γ	PTIME	NPTIME	coNPTIME	PP?



Some Results

AMC	det. model	non-det. model		prob. model
α_1	REGULAR	REGULAR	REGULAR	STOCHASTIC
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$
α_k	D_k	N_k	co-N_k	P_k
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$
α_∞	LOGSPACE	NLOGSPACE	coNLOGSPACE	PLOGSPACE
β	PTIME	PTIME	PTIME	PTIME?
γ	PTIME	NPTIME	coNPTIME	PP?

Conjecture (Weak Conjecture)

If $\alpha \sim_{\text{full}} \beta$ then $\alpha \sim_{\text{O.E.}} \beta$.

Corollary

If $\alpha : M \curvearrowright \mathbf{X}$ and $\beta : N \curvearrowright \mathbf{Y}$ are separable (by e.g. ℓ^2-Betti numbers), they characterise different complexity classes.

Strong Conjecture

Definition

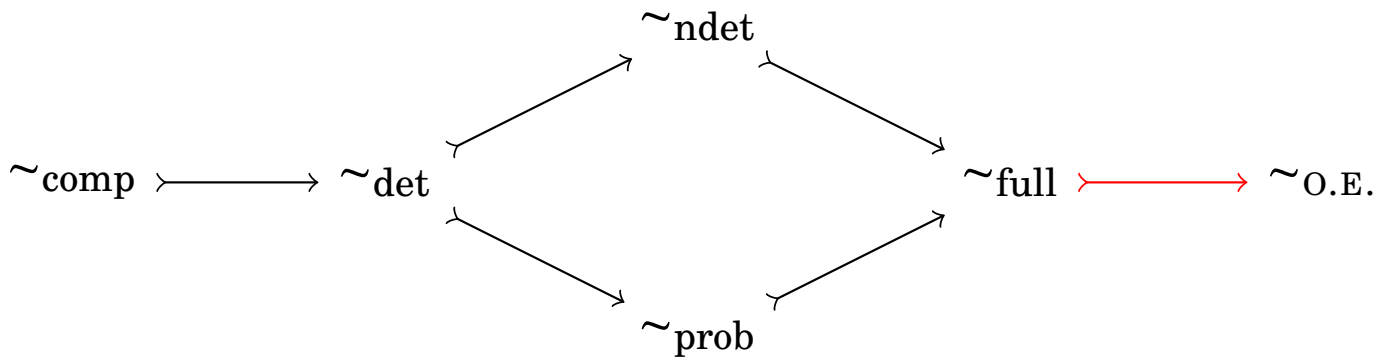
An *abstract model of computation* (AMC) is defined as a monoid action $\alpha : M \curvearrowright \mathbf{X}$ of a monoid M on a space $\mathbf{X}$. I.e. a morphism $\alpha : M \to \mathrm{End}(\mathbf{X})$.

Definition

An *abstract program* is a α-graphing.

Conjecture (Strong Conjecture)

Computational complexity of programs coincides with some tractable equivalences on monoid actions.

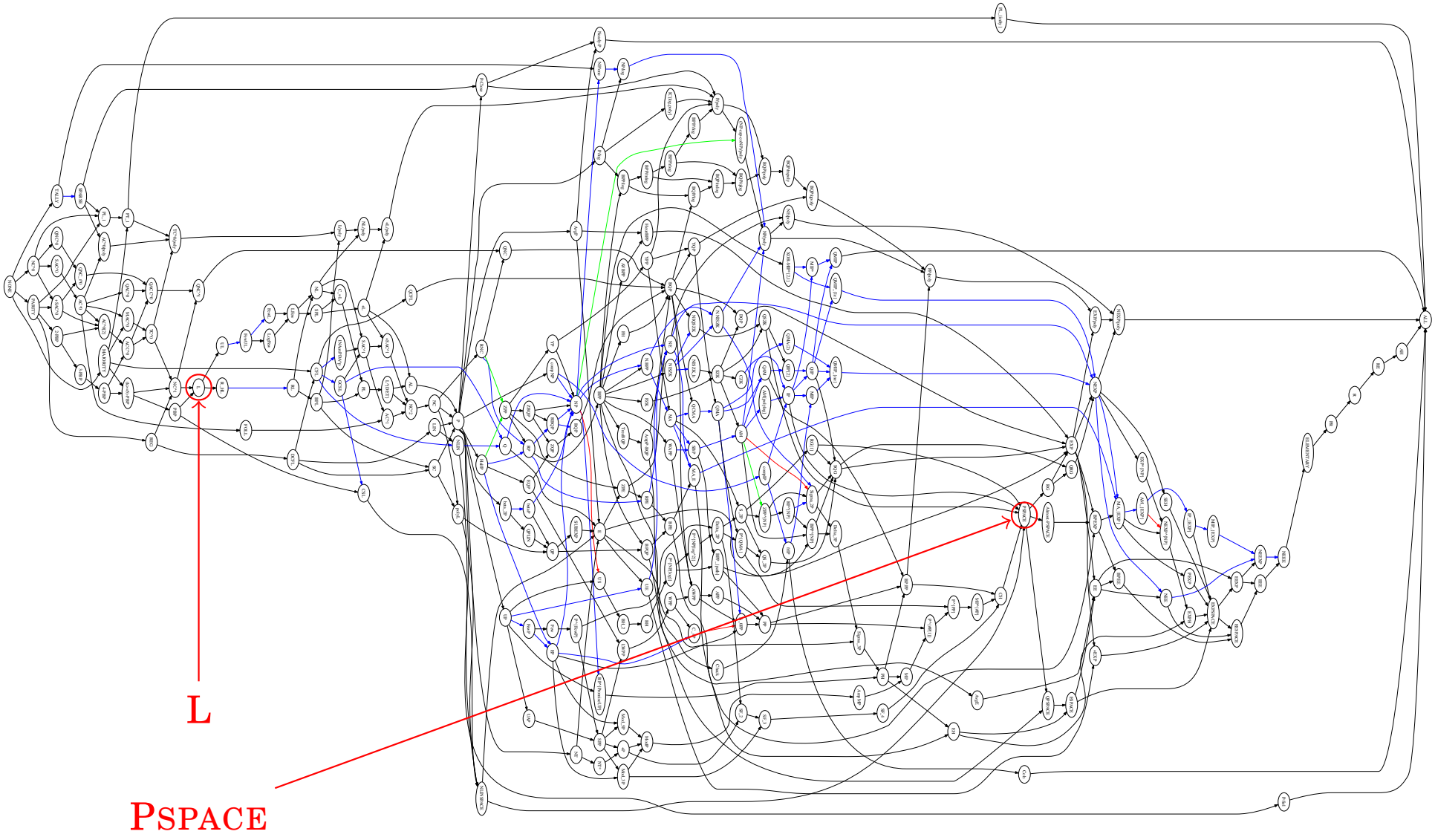


- Logic, Programs and Complexity
- Graphings and Logic
- Formalising a Conjecture
- **The Classification Problem and Barriers**
- Entropy and Lower Bounds

Complexity Theory, Today

- A number of separation results exist, most of them are from the 70s, but a lot of questions remain open. E.g. we know $\textsc{Logspace} \subsetneq \textsc{Pspace}$, but not which of these are strict:
  $\textsc{Logspace} \subset \textsc{NLogspace} \subset \textsc{NC} \subset \textsc{P} \subset \textsc{NP} \subset \textsc{Pspace}$.

Complexity Theory, Today (well, in 2006)



Complexity Theory, Today

- Separation results were obtained, most of them in the 70s, but a lot of questions remain open. E.g. we know $\text{LOGSPACE} \subsetneq \text{PSPACE}$, but not which of these are strict: $\text{LOGSPACE} \subset \text{NLOGSPACE} \subset \text{NC} \subset \text{P} \subset \text{NP} \subset \text{PSPACE}$.
- In fact, three major results are *negative results* (called *barriers*) showing that known proof methods for separation of complexity classes are inefficient w.r.t. currently open problems. They are: relativisation (1975), natural proofs (1995), and algebrization (2008).
- **Thus: no proof methods for (new) separation results exist today.**
- (Proviso) A single research program is considered as viable for obtaining new results: Mulmuley's *Geometric Complexity Theory* (GCT).

Barriers in Computational Complexity.

Morally, there are two barriers (here for PTIME vs. NPTIME):

- Relativization/Algebrization: Proof methods that are oblivious to the use/disuse of oracles are ineffective.
  - There exists oracles $\mathscr{A}, \mathscr{B}$ such that:

$$\text{PTIME}^{\mathscr{A}^{\sim}} \neq \text{NPTIME}^{\mathscr{A}}$$

$$\text{PTIME}^{\mathscr{B}} = \text{NPTIME}^{\mathscr{B}^{\sim}}$$

- Natural Proofs: Proof methods expressible as (Large, Constructible) predicates on boolean functions are ineffective.
  - A natural proof for P/POLY implies that no pseudo-random generators (in P/POLY) have exponential hardness.

Conclusion: Lack of proof methods for separation.

Barriers in Computational Complexity.

Morally, there are two barriers (here for PTIME vs. NPTIME):

- Relativization/Algebrization: Proof methods that are oblivious to the use/disuse of oracles are ineffective.
  - There exists oracles $\mathscr{A}, \mathscr{B}$ such that:

$$\text{PTIME}^{\mathscr{A}^{\sim}} \neq \text{NPTIME}^{\mathscr{A}}$$

$$\text{PTIME}^{\mathscr{B}} = \text{NPTIME}^{\mathscr{B}^{\sim}}$$

- Natural Proofs: Proof methods expressible as (**Large**, Constructible) predicates on boolean functions are ineffective.
  - A natural proof for P/POLY implies that no pseudo-random generators (in P/POLY) have exponential hardness.

Conclusion: Lack of proof methods for separation.

Geometric Complexity Theory

- GCT's aim is to prove PTIME $\neq$ NPTIME using techniques from algebraic geometry (Mulmuley and Sohoni).

Geometric Complexity Theory

- GCT's aim is to prove PTIME $\neq$ NPTIME using techniques from algebraic geometry (Mulmuley and Sohoni).
- Related to Algebraic complexity and Valiant's classes, GCT proposes a strategy for proving the permanent (VNPTIME-complete) cannot be embedded in the determinant (in VPTIME).

Geometric Complexity Theory

- GCT's aim is to prove PTIME $\neq$ NPTIME using techniques from algebraic geometry (Mulmuley and Sohoni).
- Related to Algebraic complexity and Valiant's classes, GCT proposes a strategy for proving the permanent (VNPTIME-complete) cannot be embedded in the determinant (in VPTIME).
- Mulmuley did not expect results within the next 100 years. Recently several drawbacks, closing the easiest path to GCT (Ikenmeyer).

Geometric Complexity Theory

- GCT's aim is to prove PTIME $\neq$ NPTIME using techniques from algebraic geometry (Mulmuley and Sohoni).
- Related to Algebraic complexity and Valiant's classes, GCT proposes a strategy for proving the permanent (VNPTIME-complete) cannot be embedded in the determinant (in VPTIME).
- Mulmuley did not expect results within the next 100 years. Recently several drawbacks, closing the easiest path to GCT (Ikenmeyer).
- Initiated after a proof of lower bound for a restricted algebraic PRAM model, which we note PRAM$^-$. This model defines a class NC$^-$ lying within NC (still quite large) and shows it is strictly contained within PTIME.
  Considered by some as the strongest lower bounds result so far.

Our conjecture and Barriers

Why barriers do not apply to this approach:

- (Relativisation/Algebrization)

Our conjecture and Barriers

Why barriers do not apply to this approach:

- (Relativisation/Algebrization)
  - How to describe oracles in this setting?

Our conjecture and Barriers

Why barriers do not apply to this approach:

- (Relativisation/Algebrization)
  - How to describe oracles in this setting?
  - It has to be defined *explicitly*, i.e. extend the AMC by adding a new computational principle as a map $o : \mathbf{O} \to \mathbf{O}$;

Our conjecture and Barriers

Why barriers do not apply to this approach:

- (Relativisation/Algebrization)
  - How to describe oracles in this setting?
  - It has to be defined *explicitly*, i.e. extend the AMC by adding a new computational principle as a map $o : \mathbf{O} \to \mathbf{O}$;
  - Impact the invariants: if $\alpha : M \curvearrowright \mathbf{X}$ and $\beta : N \curvearrowright \mathbf{Y}$ are separable, there are no reasons to believe that $\alpha + o : M \curvearrowright \mathbf{X}$ and $\beta + o : N \curvearrowright \mathbf{Y}$ are separable.

Our conjecture and Barriers

Why barriers do not apply to this approach:

- (Relativisation/Algebrization)
  - How to describe oracles in this setting?
  - It has to be defined *explicitly*, i.e. extend the AMC by adding a new computational principle as a map $o : \mathbf{O} \to \mathbf{O}$;
  - Impact the invariants: if $\alpha : M \curvearrowright \mathbf{X}$ and $\beta : N \curvearrowright \mathbf{Y}$ are separable, there are no reasons to believe that $\alpha + o : M \curvearrowright \mathbf{X}$ and $\beta + o : N \curvearrowright \mathbf{Y}$ are separable.
- (Natural Proofs)

Our conjecture and Barriers

Why barriers do not apply to this approach:

- (Relativisation/Algebrization)
  - How to describe oracles in this setting?
  - It has to be defined *explicitly*, i.e. extend the AMC by adding a new computational principle as a map $o : \mathbf{O} \to \mathbf{O}$;
  - Impact the invariants: if $\alpha : M \curvearrowright \mathbf{X}$ and $\beta : N \curvearrowright \mathbf{Y}$ are separable, there are no reasons to believe that $\alpha + o : M \curvearrowright \mathbf{X}$ and $\beta + o : N \curvearrowright \mathbf{Y}$ are separable.
- (Natural Proofs)
  - The approach violates the constructivity axiom of the Natural Proof barrier.

Our conjecture and Barriers

Why barriers do not apply to this approach:

- (Relativisation/Algebrization)
  - How to describe oracles in this setting?
  - It has to be defined *explicitly*, i.e. extend the AMC by adding a new computational principle as a map $o : \mathbf{O} \to \mathbf{O}$;
  - Impact the invariants: if $\alpha : M \curvearrowright \mathbf{X}$ and $\beta : N \curvearrowright \mathbf{Y}$ are separable, there are no reasons to believe that $\alpha + o : M \curvearrowright \mathbf{X}$ and $\beta + o : N \curvearrowright \mathbf{Y}$ are separable.
- (Natural Proofs)
  - The approach violates the constructivity axiom of the Natural Proof barrier.
  - More importantly, we can argue that if barriers exists in this setting then the separation problem is undecidable.

Our conjecture and Barriers

Why barriers do not apply to this approach:

- (Relativisation/Algebrization)
  - How to describe oracles in this setting?
  - It has to be defined *explicitly*, i.e. extend the AMC by adding a new computational principle as a map $o : \mathbf{O} \to \mathbf{O}$;
  - Impact the invariants: if $\alpha : M \curvearrowright \mathbf{X}$ and $\beta : N \curvearrowright \mathbf{Y}$ are separable, there are no reasons to believe that $\alpha + o : M \curvearrowright \mathbf{X}$ and $\beta + o : N \curvearrowright \mathbf{Y}$ are separable.
- (Natural Proofs)
  - The approach violates the constructivity axiom of the Natural Proof barrier.
  - More importantly, we can argue that if barriers exists in this setting then the separation problem is undecidable.
  - Uniformity seems to be easily expressed (while the Natural Proofs barrier applies to non-uniform classes).

- Logic, Programs and Complexity
- Graphings and Logic
- Formalising a Conjecture
- The Classification Problem and Barriers
- **Entropy and Lower Bounds**

Entropy and Cells

We now only consider *deterministic topological graphings*.

Definition

Let $\mathbf{X}$ be a topological space and $f : \mathbf{X} \to \mathbf{X}$ be a continuous partial map. For any finite open cover $\mathscr{U}$ of $\mathbf{X}$, we define:

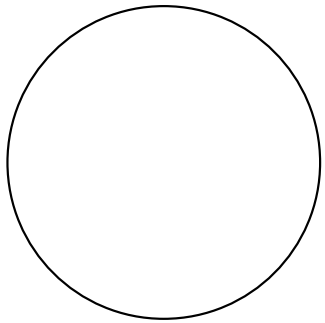
$$H^k_{\mathbf{X}}(f, \mathscr{U}) = \frac{1}{k} H^0_{f^{-k+1}(\mathbf{X})}(\mathscr{U} \vee f^{-1}(\mathscr{U}) \vee \cdots \vee f^{-(k-1)}(\mathscr{U})).$$

The *entropy* of f is then defined as $h(f) = \sup_{\mathscr{U} \in \mathrm{FCov}(\mathbf{X})} h(f, \mathscr{U})$, where $h(f, \mathscr{U})$ is again defined as the limit $\lim_{n \to \infty} H^n_{\mathbf{X}}(f, \mathscr{U})$.

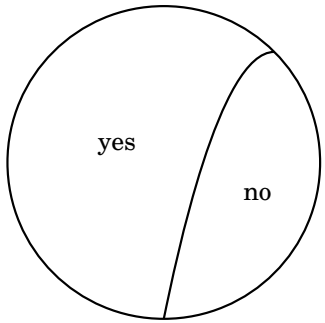
Proposition

Let G be a deterministic graphing, with entropy $h(G)$. The cardinality of the k-th cell decomposition of $\mathbf{X}$ w.r.t. G, as a function $c(k)$ of k, is asymptotically bounded by $g(k) = 2^k 2^{h([G])}$, i.e. $c(k) = O(g(k))$.

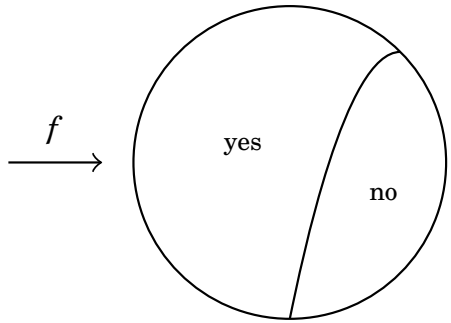
Lower Bounds and Entropy



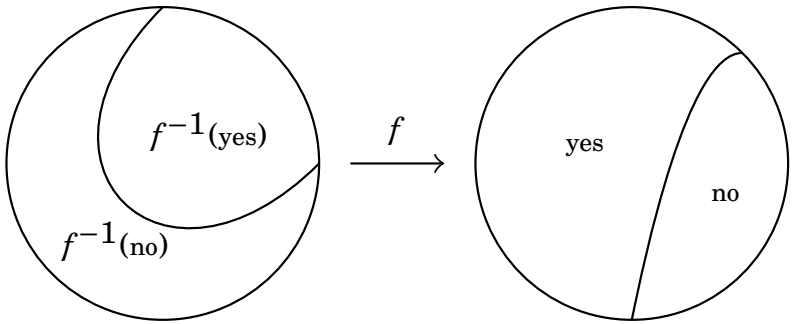
Lower Bounds and Entropy



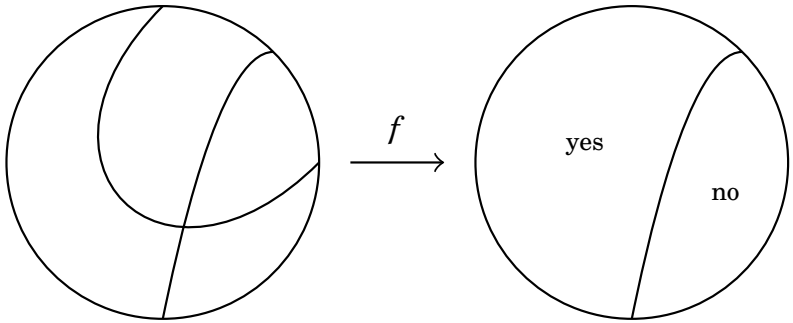
Lower Bounds and Entropy



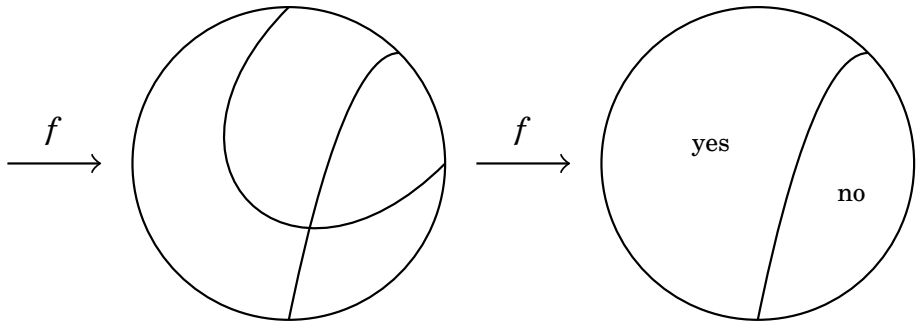
Lower Bounds and Entropy



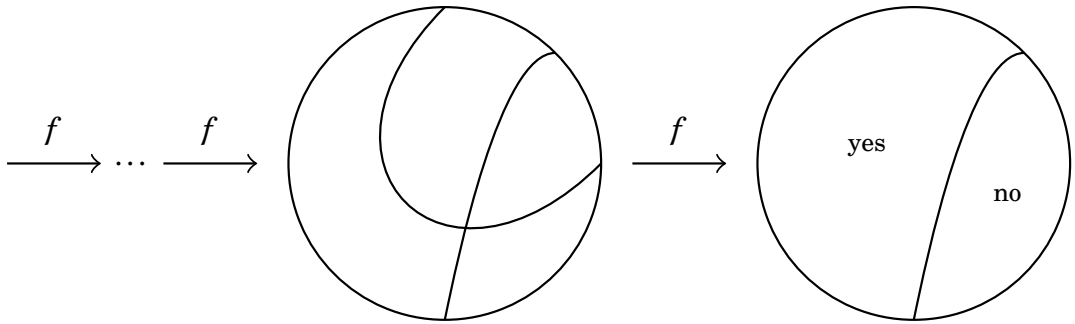
Lower Bounds and Entropy



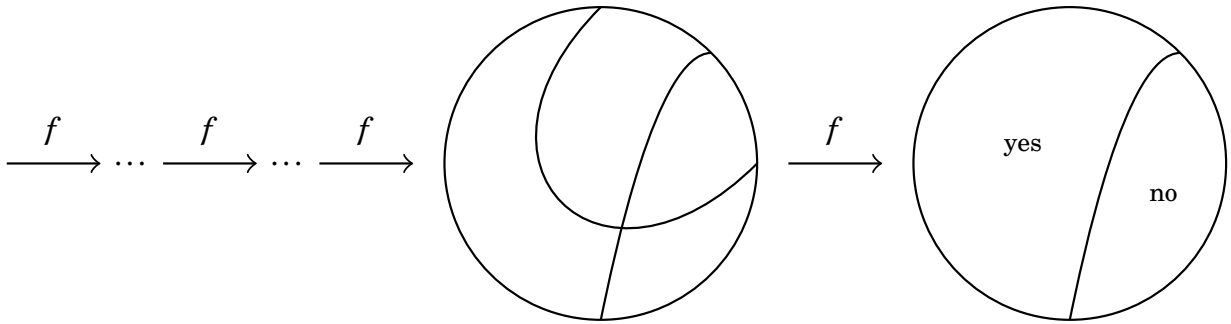
Lower Bounds and Entropy



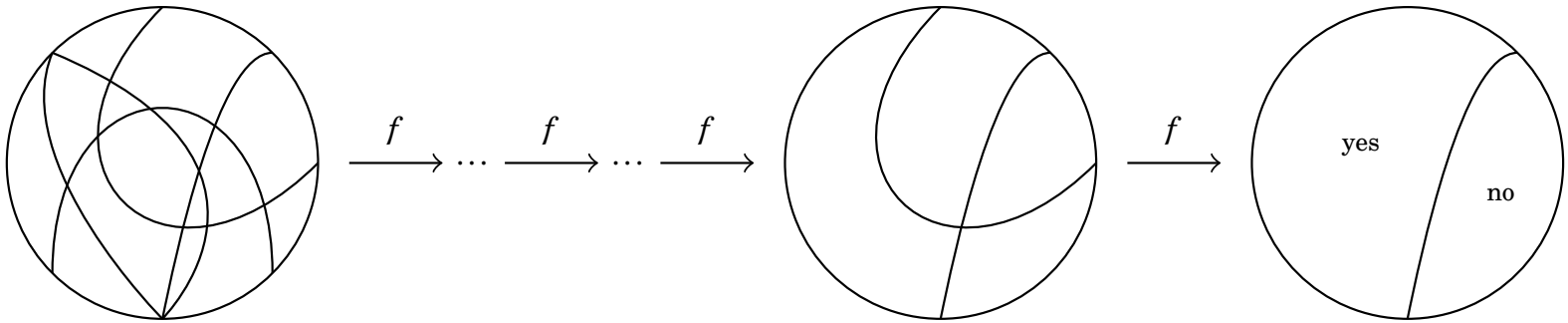
Lower Bounds and Entropy



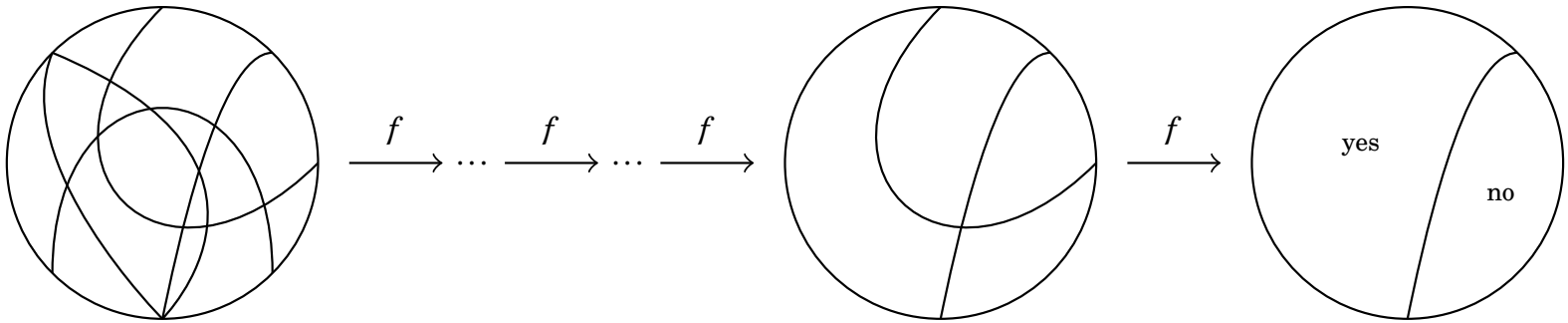
Lower Bounds and Entropy



Lower Bounds and Entropy

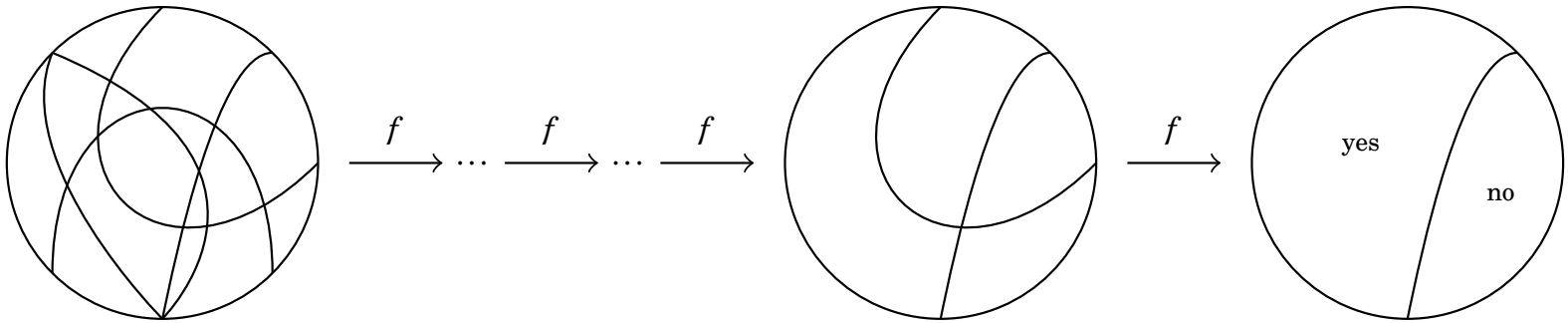


Lower Bounds and Entropy



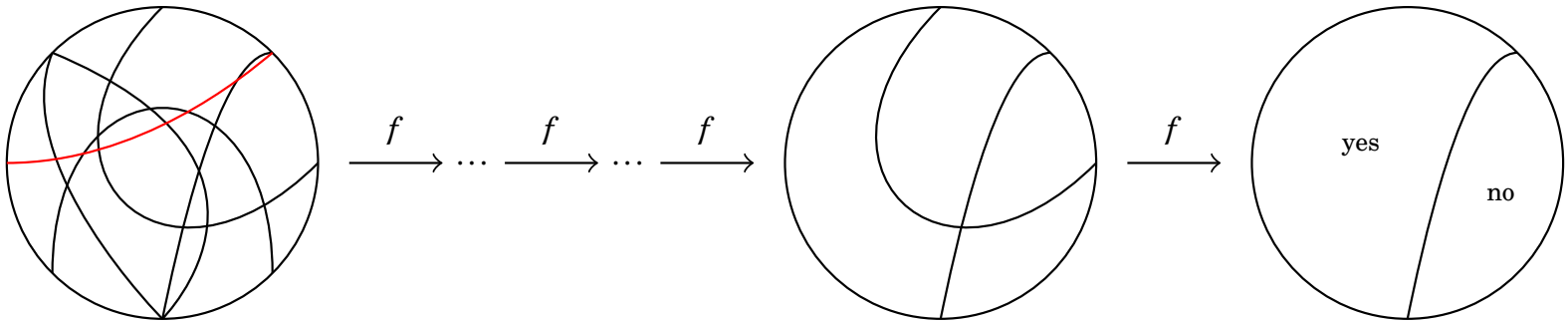
- The k-cell decomposition has the following property: if two points belong to the same cell, they are both accepted or both rejected.

Lower Bounds and Entropy



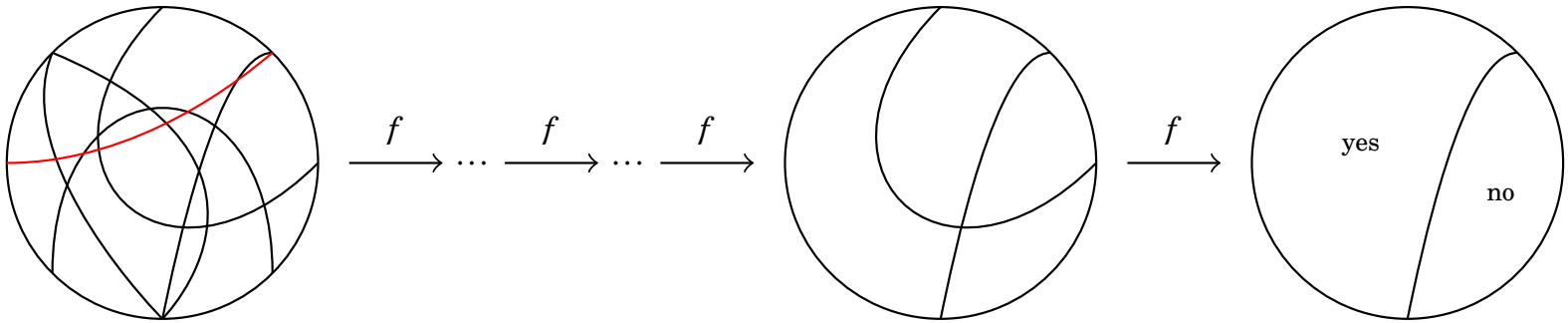
- The k-cell decomposition has the following property: if two points belong to the same cell, they are both accepted or both rejected.
- The graphing f computes a language $\mathscr{L}$ in k steps if the k-cell decomposition is a refinement of the partition corresponding to $\mathscr{L}$.

Lower Bounds and Entropy



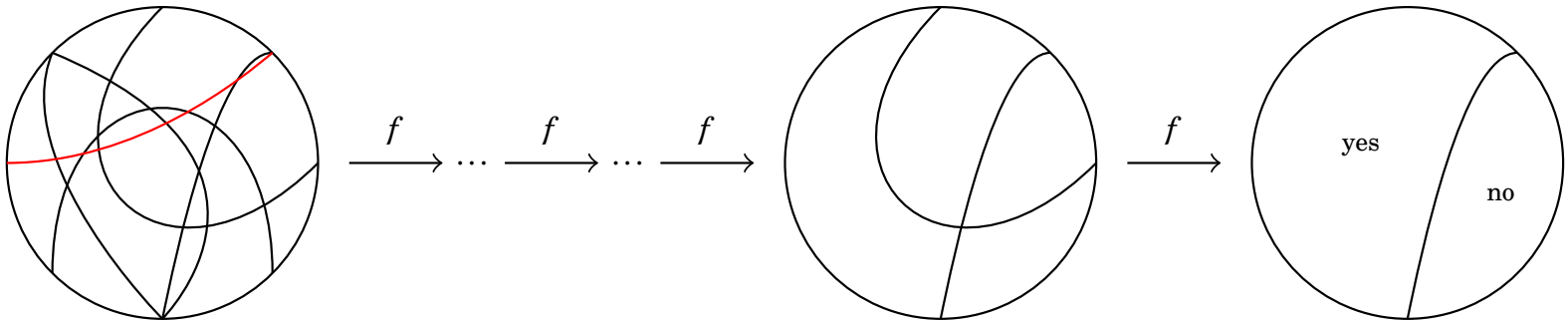
- The k-cell decomposition has the following property: if two points belong to the same cell, they are both accepted or both rejected.
- The graphing f computes a language $\mathscr{L}$ in k steps if the k-cell decomposition is a refinement of the partition corresponding to $\mathscr{L}$.

Lower Bounds and Entropy



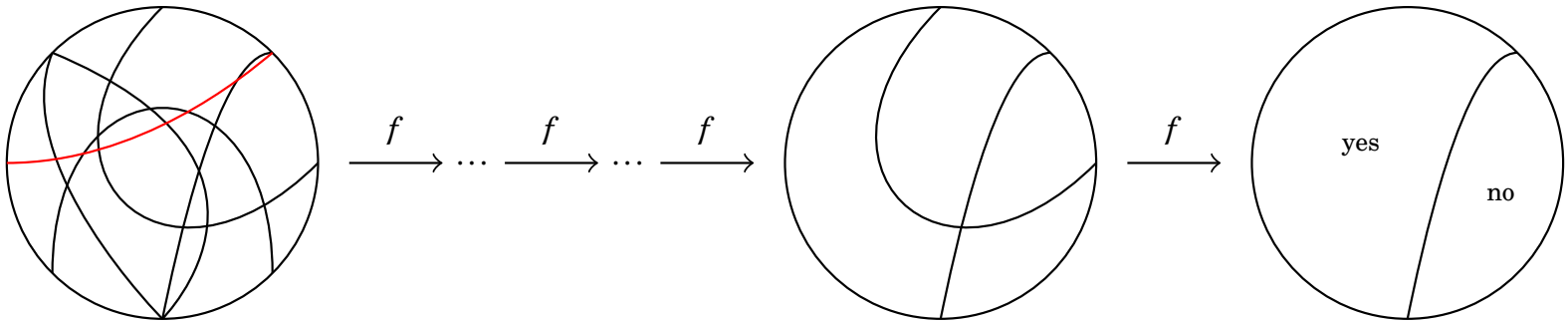
- The k-cell decomposition has the following property: if two points belong to the same cell, they are both accepted or both rejected.
- The graphing f computes a language $\mathscr{L}$ in k steps if the k-cell decomposition is a refinement of the partition corresponding to $\mathscr{L}$.
- Show lower bounds by proving a given language is *too complex* for being refined this way. E.g.:

Lower Bounds and Entropy



- The k-cell decomposition has the following property: if two points belong to the same cell, they are both accepted or both rejected.
- The graphing f computes a language $\mathscr{L}$ in k steps if the k-cell decomposition is a refinement of the partition corresponding to $\mathscr{L}$.
- Show lower bounds by proving a given language is *too complex* for being refined this way. E.g.:
  - Bound the entropy of the graphing and deduce a bound on the number of connected components of the k-cell decomposition;

Lower Bounds and Entropy



- The k-cell decomposition has the following property: if two points belong to the same cell, they are both accepted or both rejected.
- The graphing f computes a language $\mathscr{L}$ in k steps if the k-cell decomposition is a refinement of the partition corresponding to $\mathscr{L}$.
- Show lower bounds by proving a given language is *too complex* for being refined this way. E.g.:
  - Bound the entropy of the graphing and deduce a bound on the number of connected components of the k-cell decomposition;
  - Produce a given language requiring more connected components.

Lower Bounds results

- Bound the number of connected components of the k-cell decomposition (Entropy, Regularity and Milnor-Thom theorem);
- Show a given language cannot be computed by a graphing in this class because it requires more connected components (Specific Proof).

The following theorems are proved using the strategy above (w/ L. Pellissier).

Theorem (Ben-Or 1983)

A set $W \subset \mathbf{R}^n$ with N connected components cannot be decided by a degree d algebraic decision tree of height less than $\log(N) - n$.

Theorem (Cucker 1992)

$$\mathrm{NC}_{\mathbf{R}} \subsetneq \mathrm{Ptime}_{\mathbf{R}}$$

Theorem (Mulmuley 1999)

$$\mathrm{NC}^{\mathrm{PRAM}^-} \subsetneq \mathrm{Ptime}$$

From Proofs to Programs, Graphs and Dynamics. Geometric perspectives on computational complexity

Thomas Seiller
CNRS, LIPN (Paris 13 Univ.)

Cologne-Twente Workshop 2018
June 18th 2018, Paris