Computational Complexity
Between Algebra and Geometry

Thomas Seiller
University of Copenhagen

seiller@di.ku.dk

November 29th, 2016
Once upon a time, people asked (and answered) the following question:
- What is a computable function?

That's all good in theory, but once first computers were built and in use, people realised there was another important question, namely:

What is an *efficiently* computable function?

I.e. what if we wanted the answer to be produced within our lifetimes (well, quicker than that really if the result is to be used somehow).

This somehow marked the birth of computational complexity: three papers addressed this question within a year. (Cobham 1965; Hartmanis and Stearns 1965; Edmonds 1965)

And now let's fastforward.
Once upon a time, people asked (and answered) the following question:

- What is a computable function?

That’s all good in theory, but once first computers were built and in use, people realised there was another important question, namely:

- What is an *efficiently* computable function?

I.e. what if we wanted the answer to be produced within our lifetimes (well, quicker than that really if the result is to be used somehow).
Computational Complexity

Once upon a time, people asked (and answered) the following question:

- What is a computable function?

That’s all good in theory, but once first computers were built and in use, people realised there was another important question, namely:

- What is an *efficiently* computable function?

I.e. what if we wanted the answer to be produced within our lifetimes (well, quicker than that really if the result is to be used somehow).

- This somehow marked the birth of computational complexity: three papers addressed this question within a year.
  (Cobham 1965; Hartmanis and Stearns 1965; Edmonds 1965)
Once upon a time, people asked (and answered) the following question:

- What is a computable function?

That’s all good in theory, but once first computers were built and in use, people realised there was another important question, namely:

- What is an *efficiently* computable function?

I.e. what if we wanted the answer to be produced within our lifetimes (well, quicker than that really if the result is to be used somehow).

- This somehow marked the birth of computational complexity: three papers addressed this question within a year.
  (Cobham 1965; Hartmanis and Stearns 1965; Edmonds 1965)

And now let’s fastforward.
Complexity Theory, Today
A number of separation results were obtained, most of them in the 70s. But a lot of questions remain open. For instance: we know \( L \subsetneq \text{PSPACE} \), but we don’t know which of these inclusions are strict: \( L \subset P \subset NP \subset \text{PSPACE} \).
Complexity Theory, Today

- A number of separation results were obtained, most of them in the 70s. But a lot of questions remain open. For instance: we know $L \subseteq \text{PSPACE}$, but we don’t know which of these inclusions are strict: $L \subset \text{P} \subset \text{NP} \subset \text{PSPACE}$.

- In fact, the three more important 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).

(Proviso) One research program (but one only) is considered as viable for obtaining new results: Mulmuley’s Geometric Complexity Theory (GCT). However, according to Mulmuley, if GCT produces results, it will not be during our lifetimes (and maybe not our children’s lifetime either), since it requires the development of much involved new techniques in algebraic geometry.
Complexity Theory, Today

- A number of separation results were obtained, most of them in the 70s. But a lot of questions remain open. For instance: we know \( L \subsetneq \text{PSPACE} \), but we don’t know which of these inclusions are strict: \( L \subset \text{P} \subset \text{NP} \subset \text{PSPACE} \).

- In fact, the three more important 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.**
A number of separation results were obtained, most of them in the 70s. But a lot of questions remain open. For instance: we know $L \subsetneq \text{PSPACE}$, but we don’t know which of these inclusions are strict: $L \subset \text{P} \subset \text{NP} \subset \text{PSPACE}$.

In fact, the three more important 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) One research program (but one only) is considered as viable for obtaining new results: Mulmuley’s *Geometric Complexity Theory* (GCT). However, according to Mulmuley, if GCT produces results, it will not be during our lifetimes (and maybe not our childrens’ lifetime either*), since it requires the development of much involved new techniques in algebraic geometry.
Descriptive Complexity (DC) studies types of logics whose individual sentences characterise exactly particular complexity classes. E.g. (Fagin 74) \( \text{NP}^{\text{TIME}} \) is the set of problems definable by existential second order boolean formulas. DC led to the proof that \( \text{NL} = \text{coNL} \) (Immermann 88).
Logic in Complexity

- Descriptive Complexity (DC) studies types of logics whose individual sentences characterise exactly particular complexity classes. E.g. (Fagin 74) $\text{NPTIME}$ is the set of problems definable by existential second order boolean formulas. DC led to the proof that $\text{NL} = \text{coNL}$ (Immermann 88).

- Implicit Computational Complexity (ICC) studies algorithmic complexity only in terms of restrictions of languages and computational principles, for instance considering restrictions on recursion schemes (e.g. Bellantoni and Cook 92).
Logic in Complexity

- Descriptive Complexity (DC) studies types of logics whose individual sentences characterise exactly particular complexity classes. E.g. (Fagin 74) NPTIME is the set of problems definable by existential second order boolean formulas. DC led to the proof that NL = coNL (Immermann 88).

- Implicit Computational Complexity (ICC) studies algorithmic complexity only in terms of restrictions of languages and computational principles, for instance considering restrictions on recursion schemes (e.g. Bellantoni and Cook 92).

- Constrained Linear Logics is a Curry-Howard approach to ICC, where one defines “subsystems” of Girard’s linear logic which capture complexity classes (i.e. one can write less proofs, thus less programs, thus one can compute less functions).
What is an algorithm?

- One could argue that to obtain a better understanding of computational complexity, one needs to answer the question "What is an algorithm?" Currently, two main propositions: Gurevich (model-theoretic), Moschovakis (syntactic).
What is an algorithm?

- One could argue that to obtain a better understanding of computational complexity, one needs to answer the question "What is an algorithm?" Currently, two main propositions: Gurevich (model-theoretic), Moschovakis (syntactic).

- In fact, there is a third: Girard’s Geometry of interaction.
What is an algorithm?

- One could argue that to obtain a better understanding of computational complexity, one needs to answer the question "What is an algorithm?" Currently, two main propositions: Gurevich (model-theoretic), Moschovakis (syntactic).

- In fact, there is a third: Girard’s Geometry of interaction.

To quote Girard:

This paper is the main piece in a general program of mathematisation of algorithmics, called geometry of interaction. We would like to define independently of any concrete machine, any extant language, the mathematical notion of an algorithm (maybe with some proviso, e.g. deterministic algorithms), so that it would be possible to establish general results which hold once for all.

Girard, Geometry of Interaction II (1988)
Algebra

(Geometry of Interaction and Complexity)
## Algebras as models of computation

### Table:

<table>
<thead>
<tr>
<th>Logic</th>
<th>Computer Science</th>
<th>Algebras</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Proof</strong></td>
<td><strong>Program</strong></td>
<td><strong>Operator</strong>*</td>
</tr>
<tr>
<td>( \pi \vdash \text{Nat} \Rightarrow \text{Nat} )</td>
<td>( f : \text{nat} \rightarrow \text{nat} )</td>
<td>( F \in \mathcal{L}(H \oplus H) )</td>
</tr>
<tr>
<td><strong>Proof</strong></td>
<td><strong>Data</strong></td>
<td><strong>Operator</strong></td>
</tr>
<tr>
<td>( \rho \vdash \text{Nat} )</td>
<td>( n : \text{nat} )</td>
<td>( N \in \mathcal{L}(H) )</td>
</tr>
<tr>
<td><strong>Cut Rule</strong></td>
<td><strong>Application</strong></td>
<td><strong>Functional Equation</strong></td>
</tr>
</tbody>
</table>
| \( \text{cut}(\pi, \rho) \) | \( f(n) \) | \( \begin{align*}
F(x \oplus y) &= x' \oplus y' \\
N(x') &= x
\end{align*} \) |
| **Cut elimination** | **Computation** | **Construction of a solution** |
| \( \text{cut}(\pi, \rho) \leadsto \mu \vdash \text{Nat} \) | \( f(n) \leadsto m : \text{nat} \) | \( \text{Ex}(F,A)(y) = y' \in \mathcal{L}(H) \) |

*Boundeds/Continuous linear map. Think of matrices.
Theorem (Girard ’06)

If $a \in \mathcal{L}(\mathbb{H} \oplus \mathbb{K})$, $b \in \mathcal{L}(\mathbb{H})$ are operators of norm at most 1, the solution to the feedback equation involving $a$ and $b$ exists, is unique, and is an operator of norm at most 1 in the von Neumann algebra generated by $a$ and $b$.

Translation

(the unit ball of) a von Neumann algebra = set of (untyped) programs.
**Theorem (Girard ’06)**

If $a \in \mathcal{L}(\mathcal{H} \oplus \mathcal{K})$, $b \in \mathcal{L}(\mathcal{H})$ are operators of norm at most 1, the solution to the feedback equation involving $a$ and $b$ exists, is unique, and is an operator of norm at most 1 in the von Neumann algebra generated by $a$ and $b$.

**Translation**

(the 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 two results: the algebras $\mathcal{L}(\mathcal{H})$ and $\mathcal{R} \subset \mathcal{L}(\mathcal{H})$ (the type II$_\infty$ hyperfinite factor) model respectively pure lambda-calculus (Turing-complete) and "elementary programs".
Sub-algebras and complexity classes

Reality is a bit more subtle: in fact the set of programs represented by a von Neumann algebra $\mathcal{N}$ depends also on a (maximal commutative) sub-algebra $\mathcal{A} \subset \mathcal{N}$.

**Theorem (Seiller 2016)**

*Depending on the subalgebra $\mathcal{A} \subset \mathcal{R}$ chosen, the expressivity of the set of programs modelled by the von Neumann algebra $\mathcal{R}$ (the type $\text{II}_\infty$ hyperfinite factor) varies.*

The theorem is more precise than that, but the main point of interest is that the conjecture, as stated above is false.

**(False) Conjecture**

*Different von Neumann algebras = different degrees of expressivity.*
Sub-algebras and complexity classes

Reality is a bit more subtle: in fact the set of programs represented by a von Neumann algebra $\mathcal{N}$ depends also on a (maximal commutative) sub-algebra $\mathcal{A} \subset \mathcal{N}$.

**Theorem (Seiller 2016)**

*Depending on the subalgebra $\mathcal{A} \subset \mathcal{N}$ chosen, the expressivity of the set of programs modelled by the von Neumann algebra $\mathcal{N}$ (the type $\text{II}_\infty$ hyperfinite factor) varies.*

The theorem is more precise than that, but the main point of interest is that the conjecture, as stated above is false. We can try to correct it as follows.

**(Corrected) Conjecture**

*Different pairs $\mathcal{A} \subset \mathcal{N}$ = different degrees of expressivity.*
Limits of this approach

- **It (seems to) lead to dead-ends.** The different degrees of expressivity are linked with subtle properties of the pair $\mathcal{A} \subset \mathcal{V}$; and there are more open questions than answers in this particular domain of mathematics;

- **It is NOT intuitive AT ALL.** How one can express complexity constraints as choices of subalgebras?

- No non-determinism (limited by continuity/boundedness), only (subsets of) complex numbers as coefficients.

These problems can be solved by using *Interaction Graphs* models. These models have the advantage of being both more combinatorial and more general than the approach taken above. (And we’ll see, they are also more *geometric*.)
Towards Geometry
(Complexity Constraints as Measured Group Actions)
Defining concrete algebras

**Principle**

Replace pairs $A \subset N$ by pairs $(X, m)$ of a measured space $X$ and a monoid (group) $m$ of measurable maps $X \to X$.

The correspondence with the previous presentation is as follows (when $m$ is a group of measure-preserving maps):

- the algebra $A$ corresponds to the space $X$. 
  I.e. $A := L^\infty(X) \subset \mathcal{L}(L^2(X))$.
- the algebra $N$ corresponds to the group $m$. 
  I.e. each measure-preserving map $\phi$ defines a unitary $u_\phi \in \mathcal{L}(L^2(X))$ by precomposition, and $N$ is the algebra generated by the set \{ $u_\phi \mid \phi \in m$ \}.
- elements of the algebra $N$ correspond to *graphings* (generalized graphs).
Programs as graphings

What’s a graphing?

Pick a directed graph, add weights (from a monoid $\Omega$) on the edges. Consider that vertices are measurable sets, e.g. intervals. Decide how the edges map sources to targets.
Programs as graphings

What’s a graphing?

- Pick a directed graph, add weights (from a monoid $\Omega$) on the edges.

$$\frac{1}{2} \cos(\theta)$$
Programs as graphings

What’s a graphing?

- Pick a directed graph, add weights (from a monoid $\Omega$) on the edges.
- Consider that vertices are measurable sets, e.g. intervals.

```
[0, 1]   [1, 2]   [3, 4]   [4, 5]
   ^     \cos(\theta)     \frac{1}{2}
```

T. Seiller (DIKU)
What’s a graphing?

- Pick a directed graph, add weights (from a monoid $\Omega$) on the edges.
- Consider that vertices are measurable sets, e.g. intervals.
- Decide how the edges map sources to targets.
Programs as graphings

What’s a graphing?
- Pick a directed graph, add weights (from a monoid $\Omega$) on the edges.
- Consider that vertices are measurable sets, e.g. intervals.
- Decide how (i.e. which element of $m$) the edges map sources to targets.

The parameters of the construction:
- A measure space $(X, \mathcal{B}, \mu)$;
- A monoid $\Omega$;
- A monoid $m$ of measurable maps $X \to X$ – called a microcosm;
- A type of graphing (e.g. deterministic, probabilistic);
- A measurable map $m : \Omega \to \mathbb{R}_{\geq 0} \cup \{\infty\}$. 
Digression: measurable games

Consider the following (naive) games:

- Same moves for opponent and player; all sequences of moves allowed;
- (measure spaces) Set of moves = a measurable subset \( V \) of \( \mathbf{X} \);

Then, strategies are just graphs over the set \( V \) (seen as a set). One could then consider the following \textit{measure-theoretic} constraints:

- we restrict to \textbf{measurable} strategies, i.e. strategies that define a measurable map;
- we identify measurable strategies w.r.t. almost-everywhere equality.
Digression: measurable games

Consider the following (naive) games:

- Same moves for opponent and player; all sequences of moves allowed;
- (measure spaces) Set of moves = a measurable subset $V$ of $X$;

Then, strategies are just graphs over the set $V$ (seen as a set). One could then consider the following measure-theoretic constraints:

- we restrict to measurable strategies, i.e. strategies that define a measurable map;
- we identify measurable strategies w.r.t. almost-everywhere equality.

**Proposition**

*Graphings and measurable strategies are in one-to-one correspondence.*
A \textit{m-graphing} $G$ is then a measurable strategy satisfying the following additional property:

- \textbf{(m-measurability)} The measurable function corresponding to the strategy can be defined as a countable union of measurable maps in m;

Since the composition and hiding of strategies corresponds in this case to the \textit{execution} between graphings, we get the following proposition for free.

\textbf{Proposition}

The composition and hiding of two m-measurable strategies is still a m-measurable strategy.
Hierarchies of models

**Theorem (Seiller)**

*For every monoid of measurable maps \( m \) (and every monoid \( \Omega \), and every measurable map \( m : \Omega \to \mathbb{R}_{\geq 0} \cup \{\infty\} \)), the set of \( m \)-graphings defines a non-degenerate model of Multiplicative-Additive Linear Logic.*

Varying the parameters of the construction and/or on graphings, we obtain a rich hierarchy of models.

- The weights model quantitative aspects (e.g. probabilities, effects);
- The microcosms capture complexity constraints (cf. a few slides below);
- The notion of graphing can be further constrained.
Constraints on graphings

Example (Deterministic graphings)

A Ω-weighted graphing $G$ is *deterministic* when:

- for all $e \in E^G$, $\omega^G_e = 1$;
- the following set is of null measure:

$$\{ x \in X \mid \sum_{e \in E^G, x \in s^G(e)} \omega^G(e) > 1 \}$$

Proposition

A deterministic graphing is a partial dynamical system.

Proposition

Deterministic graphings are closed under composition.
Constraints on graphings

Example (Deterministic graphings)

A $\Omega$-weighted graphing $G$ is deterministic when:

- for all $e \in E^G$, $\omega_e^G = 1$;
- the following set is of null measure:

$$\{x \in X \mid \sum_{e \in E^G, x \in s^G(e)} \omega^G(e) > 1\}$$

Proposition

A deterministic graphing is a partial dynamical system.

Proposition

Deterministic graphings are closed under composition.
Example (Probabilistic graphings)

A \( \Omega \)-weighted graphing \( G \) is *probabilistic* when:

- for all \( e \in E^G \), \( \omega^G_e \in ]0, 1] \);
- the following set is of null measure:

\[
\{ x \in X \mid \sum_{e \in E^G, x \in s^G(e)} \omega^G(e) \neq 1 \}
\]
Constraints on graphings

Example (Probabilistic graphings)

A $\Omega$-weighted graphing $G$ is probabilistic when:

- for all $e \in E^G$, $\omega^G_e \in ]0, 1]$;
- the following set is of null measure:

$$\{ x \in X \mid \sum_{e \in E^G, x \in s^G(e)} \omega^G(e) \neq 1 \}$$

Proposition

A probabilistic graphing is a continuous (discrete-time) Markov process.

Proposition

Probabilistic graphings are closed under composition.
Constraints on parameters

Figure: Inclusions of models
We define models with (very) weak exponential connectives.

In this respect, this approach is somehow a semantic version of constrained linear logics, although the freeing from syntax makes it more flexible.

We can define microcosms \( m_1 \subset m_2 \subset \cdots \subset m_\infty \subset n \subset p \) in order to obtain the following characterisations.

<table>
<thead>
<tr>
<th>Microcosm</th>
<th>( M^\text{det}_m )</th>
<th>( M^n\text{det}_m )</th>
<th>( M^n\text{det}_m )</th>
<th>( M^\text{prob}_m )</th>
<th>Logic</th>
<th>Machines</th>
</tr>
</thead>
<tbody>
<tr>
<td>( m_1 )</td>
<td>REG</td>
<td>REG</td>
<td>REG</td>
<td>STOC</td>
<td>MALL</td>
<td>2-way Automata (2FA)</td>
</tr>
<tr>
<td>( \vdots )</td>
<td>( \vdots )</td>
<td>( \vdots )</td>
<td>( \vdots )</td>
<td>( \vdots )</td>
<td>( \vdots )</td>
<td>( \vdots )</td>
</tr>
<tr>
<td>( m_k )</td>
<td>( D_k )</td>
<td>( N_k )</td>
<td>( \text{CON}_k )</td>
<td>( P_k )</td>
<td>(…)</td>
<td>( k )-heads 2FA</td>
</tr>
<tr>
<td>( \vdots )</td>
<td>( \vdots )</td>
<td>( \vdots )</td>
<td>( \vdots )</td>
<td>( \vdots )</td>
<td>( \vdots )</td>
<td>( \vdots )</td>
</tr>
<tr>
<td>( m_\infty )</td>
<td>L</td>
<td>NL</td>
<td>( \text{CONL} )</td>
<td>PL</td>
<td>(…)</td>
<td>multihead-head 2FA (2MHFA)</td>
</tr>
<tr>
<td>( n )</td>
<td>P</td>
<td>P</td>
<td>P</td>
<td>P</td>
<td>(…)</td>
<td>2MHFA + Pushdown Stack</td>
</tr>
<tr>
<td>( p )</td>
<td>P</td>
<td>NP</td>
<td>( \text{CONP} )</td>
<td>PP?</td>
<td>ELL</td>
<td>Ptime Turing Machines</td>
</tr>
</tbody>
</table>

Conjecture: microcosms correspond to complexity constraints.
The conjecture, formally

- We define an equivalence relation on monoid of measurable maps.
- Notation: we fix every parameter but the monoid $m$, and write $\text{Pred}(m)$ the complexity class (predicates) thus characterised.

**Theorem**

*If* $m \equiv n$, *then* $\text{Pred}(m) = \text{Pred}(n)$.

**Conjecture**

*The converse holds, i.e. $\text{Pred}(m) = \text{Pred}(n)$ implies* $m \equiv n$.

If this conjecture holds, it would provide new proof techniques for separation through, e.g., (co)homological invariants such as $\ell^{(2)}$-Betti numbers:

$$\text{Pred}(m) = \text{Pred}(n) \Rightarrow m \equiv n \Rightarrow \mathcal{P}(m) \simeq \mathcal{P}(n) \Rightarrow \ell^{(2)}(\mathcal{P}(m)) = \ell^{(2)}(\mathcal{P}(n))$$

where $\mathcal{P}(m) = \{(x, y) \mid \exists m \in m, m(x) = y\}$ is a “measurable preorder”.
Geometry

or Why should we believe in this conjecture?
Orthogonality in Interaction Graphs

- Initially defined to provide a *geometric* counterpart to Girard’s hyperfinite GoI orthogonality expressed by a determinant. (And it does, but this was not a new result.)
- Defined in terms of the following quantity:

\[ \sum_{\pi \in \mathcal{C}(F \Box G)} m(\omega(\pi)) \]

where \( \mathcal{C}(F \Box G) \) is the set of 1-circuits in the graph \( F \Box G \).
Orthogonality in Interaction Graphs

- Initially defined to provide a *geometric* counterpart to Girard’s hyperfinite GoI orthogonality expressed by a determinant. (And it does, but this was not a new result.)

- Defined in terms of the following quantity:

  \[ \sum_{\pi \in C(F \square G)} m(\omega(\pi)) \]

  where \( C(F \square G) \) is the set of 1-circuits in the graph \( F \square G \).

- It turns out that 1-circuits were already defined somewhere in the literature, and named *prime cycles*; they are used to define Ihara’s *zeta functions* of graphs (and their generalisations).
Orthogonality in Interaction Graphs

- Initially defined to provide a *geometric* counterpart to Girard’s hyperfinite GoI orthogonality expressed by a determinant. (And it does, but this was not a new result.)
- Defined in terms of the following quantity:

\[ \sum_{\pi \in \mathcal{C}(F \Box G)} m(\omega(\pi)) \]

where \( \mathcal{C}(F \Box G) \) is the set of 1-circuits in the graph \( F \Box G \).
- It turns out that 1-circuits were already defined somewhere in the literature, and named *prime cycles*; they are used to define Ihara’s *zeta functions* of graphs (and their generalisations).
- (More generally) Orthogonality in graphings is related to Ruelle’s zeta functions for dynamical systems.
Zeta functions

Where is the geometry?

- Zeta functions (defined from e.g. graphs, manifolds, dynamical systems) are extensively used in mathematics because they carry – as (complex) analytic properties – lots of information about the geometry of the objects they are defined from.

- For instance, one can “read” in the zeta function if a graph is bipartite, cyclic, regular;

- Another illustration: the following expression (Bass) for the inverse of the zeta function of a graph.

\[
\zeta_G(z)^{-1} = (1 - z^2)^{r-1} \det(I - zB + z^2Q),
\]

let appear the Euler characteristic \( r \) of the graph, i.e. the alternating sum of the Betti numbers (dimension of the i-th homology group).

(Here \( B \) is the adjacency matrix of \( G \), \( I \) is the identity matrix and \( Q = D - I \) with \( D \) a "degree (diagonal) matrix".)
In the results above about complexity: the orthogonality is used in the characterisations of complexity classes. Indeed, it is used to test the result of a computation, e.g. it is used to decide whether the program accepts or rejects the entry.

This supports a geometric approach to computational complexity. In particular, we can hope to express separation problems in geometric terms: since those are defined through zeta functions, we can produce geometric objects (e.g. (families of) algebraic manifolds) capturing these problems.

In the end, this might provide a proof of the conjecture, and provide new methods for proving separation results.

(Note: this may be related to Mulmuley’s techniques as he associates complexity classes to families of algebraic manifolds).

Is this viable? (w.r.t. the barriers)
The aim is to formally relate the following two problems:

- **C.S.** Are two complexity classes $A, B$ equal?
  - Difficult to answer negatively;
  - No proof methods available.
The aim is to formally relate the following two problems:

- **C.S.** Are two complexity classes $A, B$ equal?
  - Difficult to answer negatively;
  - No proof methods available.

- **Mathematics.** Are two spaces $X, Y$ homotopy equivalent?
  - Difficult to answer negatively;
The aim is to formally relate the following two problems:

- **C.S.** Are two complexity classes $A, B$ equal?
  - Difficult to answer negatively;
  - No proof methods available.

- **Mathematics.** Are two spaces $X, Y$ homotopy equivalent?
  - Difficult to answer negatively;
  - Some proof methods available (e.g. (co)homological invariants).