On the information carried by programs about the objects they compute

Mathieu Hoyrup and Cristóbal Rojas

LORIA - Inria, Nancy (France)
The problem

Two ways of providing a computable function $f : \mathbb{N} \rightarrow \mathbb{N}$ to a machine:

- Via the **graph** of $f$ (*infinite* object),
- Via a **program** computing $f$ (*finite* object).
The problem

Two ways of providing a computable function $f : \mathbb{N} \rightarrow \mathbb{N}$ to a machine:

- Via the **graph** of $f$ (*infinite* object),
- Via a **program** computing $f$ (*finite* object).

Main questions

- Does it make a difference?
- Can the two machines perform the same tasks?
- Does the code of a program give more information about what it computes?
The problem

The answer depends on:

- Whether the functions $f$ are **partial** or **total**, 
- The task to be performed by the machine (e.g. **decide** or **semi-decide** something).

<table>
<thead>
<tr>
<th></th>
<th>Decidability</th>
<th>Semi-decidability</th>
</tr>
</thead>
<tbody>
<tr>
<td>Partial functions</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Total functions</td>
<td></td>
<td></td>
</tr>
<tr>
<td>The problem</td>
<td>Historical results</td>
<td>New results</td>
</tr>
<tr>
<td>-------------</td>
<td>--------------------</td>
<td>-------------</td>
</tr>
</tbody>
</table>

The problem

**Historical results**

New results

Limits
### Partial functions

<table>
<thead>
<tr>
<th></th>
<th>Decidability</th>
<th>Semi-decidability</th>
</tr>
</thead>
<tbody>
<tr>
<td>Partial functions</td>
<td>?</td>
<td></td>
</tr>
<tr>
<td>Total functions</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
# Partial functions

<table>
<thead>
<tr>
<th></th>
<th>Decidability</th>
<th>Semi-decidability</th>
</tr>
</thead>
<tbody>
<tr>
<td>Partial functions</td>
<td>?</td>
<td></td>
</tr>
<tr>
<td>Total functions</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Given (any enumeration of) the graph of $f$, one cannot decide whether $f(0)$ is defined.
Partial functions

<table>
<thead>
<tr>
<th></th>
<th>Decidability</th>
<th>Semi-decidability</th>
</tr>
</thead>
<tbody>
<tr>
<td>Partial functions</td>
<td>?</td>
<td></td>
</tr>
<tr>
<td>Total functions</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Given (any enumeration of) the graph of \( f \), one cannot decide whether \( f(0) \) is defined.

**Theorem (Turing, 1936)**

*Given a program for \( f \), a machine cannot do better.*
Partial functions

<table>
<thead>
<tr>
<th></th>
<th>Decidability</th>
<th>Semi-decidability</th>
</tr>
</thead>
<tbody>
<tr>
<td>Partial functions</td>
<td>?</td>
<td></td>
</tr>
<tr>
<td>Total functions</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

More generally, what can be decided about $f$?
Partial functions

<table>
<thead>
<tr>
<th></th>
<th>Decidability</th>
<th>Semi-decidability</th>
</tr>
</thead>
<tbody>
<tr>
<td>Partial functions</td>
<td>?</td>
<td></td>
</tr>
<tr>
<td>Total functions</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

More generally, what can be **decided** about \( f \)?

**Answers**

Given the graph of \( f \), only trivial properties: the decision about \( \lambda x. \perp \) applies to every \( f \).
Partial functions

<table>
<thead>
<tr>
<th></th>
<th>Decidability</th>
<th>Semi-decidability</th>
</tr>
</thead>
<tbody>
<tr>
<td>Partial functions</td>
<td>program ≡ graph</td>
<td></td>
</tr>
<tr>
<td>Total functions</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

More generally, what can be decided about $f$?

**Answers**

Given the graph of $f$, only trivial properties: the decision about $\lambda x. \perp$ applies to every $f$.

**Theorem (Rice, 1953)**

*Given a program for $f$, a machine cannot do better.*
**Partial functions**

<table>
<thead>
<tr>
<th></th>
<th>Decidability</th>
<th>Semi-decidability</th>
</tr>
</thead>
<tbody>
<tr>
<td>Partial functions</td>
<td><code>program ≡ graph</code></td>
<td>?</td>
</tr>
<tr>
<td>Total functions</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

What can be **semi-decided** about $f$?
Partial functions

<table>
<thead>
<tr>
<th></th>
<th>Decidability</th>
<th>Semi-decidability</th>
</tr>
</thead>
<tbody>
<tr>
<td>Partial functions</td>
<td><em>program</em> ≡ <em>graph</em></td>
<td>?</td>
</tr>
<tr>
<td>Total functions</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

What can be **semi-decided** about $f$?

**Answers**

Given the **graph** of $f$, exactly the properties of the form:

$$(f(a_1) = u_1 \land \ldots \land f(a_i) = u_i)$$

$\lor (f(b_1) = v_1 \land \ldots \land f(b_j) = v_j)$$

$\lor (f(c_1) = w_1 \land \ldots \land f(c_k) = w_k)$$

$\lor \ldots$$
Partial functions

<table>
<thead>
<tr>
<th></th>
<th>Decidability</th>
<th>Semi-decidability</th>
</tr>
</thead>
<tbody>
<tr>
<td>Partial functions</td>
<td><em>program ≡ graph</em></td>
<td><em>program ≡ graph</em></td>
</tr>
<tr>
<td>Total functions</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

What can be **semi-decided** about $f$?

**Answers**

Given the **graph** of $f$, exactly the properties of the form:

$$
(f(a_1) = u_1 \land \ldots \land f(a_i) = u_i) \\
\lor \quad (f(b_1) = v_1 \land \ldots \land f(b_j) = v_j) \\
\lor \quad (f(c_1) = w_1 \land \ldots \land f(c_k) = w_k) \\
\lor \quad \ldots
$$

**Theorem (Shapiro, 1956)**

*Given a program for $f$, a machine cannot do better.*
Total functions

<table>
<thead>
<tr>
<th></th>
<th>Decidability</th>
<th>Semi-decidability</th>
</tr>
</thead>
<tbody>
<tr>
<td>Partial functions</td>
<td>( \text{program} \equiv \text{graph} )</td>
<td>( \text{program} \equiv \text{graph} )</td>
</tr>
<tr>
<td>Total functions</td>
<td>?</td>
<td></td>
</tr>
</tbody>
</table>

What can be decided/semi-decided about \( f \)?
Total functions

<table>
<thead>
<tr>
<th></th>
<th>Decidability</th>
<th>Semi-decidability</th>
</tr>
</thead>
<tbody>
<tr>
<td>Partial functions</td>
<td>$\text{program } \equiv \text{graph}$</td>
<td>$\text{program } \equiv \text{graph}$</td>
</tr>
<tr>
<td>Total functions</td>
<td>$\text{program } \equiv \text{graph}$</td>
<td>?</td>
</tr>
</tbody>
</table>

What can be decided/semi-decided about $f$?

**Theorem (Kreisel-Lacombe-Schoenfield/Ceitin, 1957/1962)**

*For properties of total computable functions,*

\[
\text{decidable from a program} \iff \text{decidable from the graph}.
\]
Total functions

<table>
<thead>
<tr>
<th></th>
<th>Decidability</th>
<th>Semi-decidability</th>
</tr>
</thead>
<tbody>
<tr>
<td>Partial functions</td>
<td><code>program \equiv graph</code></td>
<td><code>program \equiv graph</code></td>
</tr>
<tr>
<td>Total functions</td>
<td><code>program \equiv graph</code></td>
<td><code>program &gt; graph</code></td>
</tr>
</tbody>
</table>

What can be decided/semi-decided about $f$?

**Theorem (Kreisel-Lacombe-Schoenfield/Ceitin, 1957/1962)**

For properties of total computable functions,

\[
\text{decidable from a program } \iff \text{decidable from the graph.}
\]

It does make a difference!

**Theorem (Friedberg, 1958)**

For properties of total computable functions,

\[
\text{semi-decidable from a program } \not\iff \text{semi-decidable from the graph.}
\]
Friedberg’s property

\[
\psi(x) = \begin{cases} 
0, & \text{if either } (\forall y)[y \leq x \Rightarrow \varphi_x(y) = 0] \text{ or } (\exists z)[\varphi_x(z) \neq 0] \text{ and } (\forall y)[y < z \Rightarrow \varphi_x(y) = 0] \text{ and } (\exists x')[x' < z \text{ and } (\forall u)[u \leq z \Rightarrow \varphi_{x'}(u) = \varphi_x(u)]]; \\
\text{divergent,} & \text{otherwise.}
\end{cases}
\]

Figure: Taken from Rogers

- Invented in 1958, easier to express using Kolmogorov complexity (1960’s).
- Say \( n \in \mathbb{N} \) is **compressible** if \( K(n) < \log(n) \).
Friedberg’s property

Given a total function \( f \neq \lambda x.0 \), let

\[
n_f = \min \{ n : f(n) \neq 0 \}.\]

Friedberg’s property is

\[
P = \{ \lambda x.0 \} \cup \{ f : n_f \text{ is compressible} \}.
\]
Friedberg’s property

Given a total function \( f \neq \lambda x.0 \), let

\[
  n_f = \min\{n : f(n) \neq 0\}.
\]

Friedberg’s property is

\[
P = \{\lambda x.0\} \cup \{f : n_f \text{ is compressible}\}.
\]

Semi-deciding \( f \in P \)

<table>
<thead>
<tr>
<th>( n )</th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>...</th>
</tr>
</thead>
<tbody>
<tr>
<td>( f(n) )</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Friedberg’s property

Given a total function $f \neq \lambda x.0$, let

$$n_f = \min\{n : f(n) \neq 0\}.$$

Friedberg’s property is

$$P = \{\lambda x.0\} \cup \{f : n_f \text{ is compressible}\}.$$
Friedberg’s property

Given a total function \( f \neq \lambda x.0 \), let

\[
n_f = \min\{ n : f(n) \neq 0 \}.
\]

Friedberg’s property is

\[
P = \{ \lambda x.0 \} \cup \{ f : n_f \text{ is compressible} \}.
\]

Semi-deciding \( f \in P \)

<table>
<thead>
<tr>
<th>( n )</th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>( \ldots )</th>
</tr>
</thead>
<tbody>
<tr>
<td>( f(n) )</td>
<td>0</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

When is it time to accept \( f \)?

- If \( f \) is given by its graph, we can never know.
- If \( f \) is given by a program \( p \) then evaluate on inputs \( 0, \ldots, |p| \).
Friedberg’s property

Given a total function $f \neq \lambda x.0$, let

$$n_f = \min\{n : f(n) \neq 0\}.$$  

Friedberg’s property is

$$P = \{\lambda x.0\} \cup \{f : n_f \text{ is compressible}\}.$$  

Semi-deciding $f \in P$

<table>
<thead>
<tr>
<th>$n$</th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>...</th>
</tr>
</thead>
<tbody>
<tr>
<td>$f(n)$</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Friedberg’s property

Given a total function \( f \neq \lambda x.0 \), let

\[
n_f = \min\{n : f(n) \neq 0\}.
\]

Friedberg’s property is

\[
P = \{\lambda x.0\} \cup \{f : n_f \text{ is compressible}\}.
\]

Semi-deciding \( f \in P \)

<table>
<thead>
<tr>
<th>( n )</th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>...</th>
</tr>
</thead>
<tbody>
<tr>
<td>( f(n) )</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>4</td>
<td>5</td>
<td>6</td>
<td></td>
</tr>
</tbody>
</table>
Friedberg’s property

Given a total function $f \neq \lambda x.0$, let

$$n_f = \min\{n : f(n) \neq 0\}.$$

Friedberg’s property is

$$P = \{\lambda x.0\} \cup \{f : n_f \text{ is compressible}\}.$$

Semi-deciding $f \in P$

<table>
<thead>
<tr>
<th>$n$</th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>...</th>
</tr>
</thead>
<tbody>
<tr>
<td>$f(n)$</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
**Friedberg’s property**

Given a total function $f \neq \lambda x.0$, let

$$n_f = \min\{n : f(n) \neq 0\}.$$ 

Friedberg’s property is

$$P = \{\lambda x.0\} \cup \{f : n_f \text{ is compressible}\}.$$ 

**Semi-deciding $f \in P$**

<table>
<thead>
<tr>
<th>$n$</th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>...</th>
</tr>
</thead>
<tbody>
<tr>
<td>$f(n)$</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td></td>
</tr>
</tbody>
</table>
Friedberg’s property

Given a total function \( f \neq \lambda x.0 \), let

\[
n_f = \min\{n : f(n) \neq 0\}.
\]

Friedberg’s property is

\[
P = \{\lambda x.0\} \cup \{f : n_f \text{ is compressible}\}.
\]

### Semi-deciding \( f \in P \)

<table>
<thead>
<tr>
<th>( n )</th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>...</th>
</tr>
</thead>
<tbody>
<tr>
<td>( f(n) )</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>( f(n) )</td>
</tr>
</tbody>
</table>
Friedberg’s property

Given a total function $f \neq \lambda x.0$, let

$$n_f = \min\{n : f(n) \neq 0\}.$$

Friedberg’s property is

$$P = \{\lambda x.0\} \cup \{f : n_f \text{ is compressible}\}.$$

<table>
<thead>
<tr>
<th>$n$</th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>...</th>
</tr>
</thead>
<tbody>
<tr>
<td>$f(n)$</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>...</td>
</tr>
</tbody>
</table>
Friedberg’s property

Given a total function $f \neq \lambda x.0$, let

$$n_f = \min\{n : f(n) \neq 0\}.$$

Friedberg’s property is

$$P = \{\lambda x.0\} \cup \{f : n_f \text{ is compressible}\}.$$

Semi-deciding $f \in P$

<table>
<thead>
<tr>
<th>$n$</th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>...</th>
</tr>
</thead>
<tbody>
<tr>
<td>$f(n)$</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td></td>
</tr>
</tbody>
</table>

When is it time to accept $f$?

- If $f$ is given by its graph, we can never know.
Friedberg’s property

Given a total function $f \neq \lambda x.0$, let

$$n_f = \min\{n : f(n) \neq 0\}.$$  

Friedberg’s property is

$$P = \{\lambda x.0\} \cup \{f : n_f \text{ is compressible}\}.$$  

Semi-deciding $f \in P$

<table>
<thead>
<tr>
<th>$n$</th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>...</th>
</tr>
</thead>
<tbody>
<tr>
<td>$f(n)$</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>...</td>
</tr>
</tbody>
</table>

When is it time to accept $f$?

- If $f$ is given by its graph, we can never know.
- If $f$ is given by a program $p$ then evaluate $f$ on inputs $0, \ldots, 2^{|p|}$. 
Sum up

Two computation models:
- **Markov-computability**: given a program,
- **Type-2-computability**: given the graph.

<table>
<thead>
<tr>
<th></th>
<th>Decidability</th>
<th>Semi-decidability</th>
</tr>
</thead>
<tbody>
<tr>
<td>Partial functions</td>
<td>Markov $\equiv$ Type-2 $^\text{Rice}$</td>
<td>Markov $\equiv$ Type-2 $^\text{Rice-Shapiro}$</td>
</tr>
<tr>
<td>Total functions</td>
<td>Markov $\equiv$ Type-2 $^\text{Kreisel-Lacombe-Schænfield/Ceitin}$</td>
<td>Markov $&gt; \text{Type-2} ^\text{Friedberg}$</td>
</tr>
<tr>
<td>The problem</td>
<td>Historical results</td>
<td>New results</td>
</tr>
<tr>
<td>-------------</td>
<td>--------------------</td>
<td>-------------</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

The problem

Historical results

New results

Limits
Let $f$ be a computable function. All the programs computing $f$ share some common information about $f$:

- The information needed to recover the graph of $f$,
- Plus some extra information about $f$.

**Question**

What is the extra information?
Let $f$ be a computable function. All the programs computing $f$ share some common information about $f$:

- The information needed to recover the graph of $f$,
- Plus some extra information about $f$.

**Question**

What is the extra information?

**Answer**

They bound the Kolmogorov complexity of $f$!
First main result

Let

\[ K(f) = \min \{ |p| : p \text{ computes } f \} \].

Theorem

Let \( P \) be a property of total functions. The following are equivalent:

- \( f \in P \) is Markov-semi-decidable,
- \( f \in P \) is Type-2-semi-decidable given any upper bound on \( K(f) \).
First main result

Let

\[ K(f) = \min\{|p| : p \text{ computes } f\}. \]

Theorem

Let \( P \) be a property of total functions. The following are equivalent:

- \( f \in P \) is Markov-semi-decidable,
- \( f \in P \) is Type-2-semi-decidable given any upper bound on \( K(f) \).

In other words, the only useful information provided by a program \( p \) for \( f \) is:

- the graph of \( f \) (by running \( p \)),
- an upper bound on \( K(f) \) (namely, \( |p| \)).
More general results

The result is much more general and holds for:

- many classes of objects other than total functions:
  \( 2^\omega, \mathbb{R}, \text{any effective topological space} \)
- many notions other than semi-decidability:
  computable functions, \( n \)-c.e. properties, \( \Sigma^0_2 \) properties
More general results

The result is much more general and holds for:

- many classes of objects other than total functions: $2^\omega$, $\mathbb{R}$, any effective topological space
- many notions other than semi-decidability:
  
  computable functions, $n$-c.e. properties, $\Sigma^0_2$ properties

For instance,

**Theorem (Computable functions)**

Let $X, Y$ be effective topological spaces and $f : X \to Y$.

$f$ is Markov-computable $\iff f$ is (Type-2, K)-computable.
More general results

Example: $n$-c.e. properties of partial functions.

**Theorem (Selivanov, 1984)**

*There is a property of partial functions that is*

- 2-c.e. in the Markov-model,
- not 2-c.e. (and not even $\Pi^0_2$) in the Type-2-model.
More general results

Example: $n$-c.e. properties of partial functions.

**Theorem (Selivanov, 1984)**

There is a property of partial functions that is

- 2-c.e. in the Markov-model,
- not 2-c.e. (and not even $\Pi^0_2$) in the Type-2-model.

Again,

**Theorem**

Let $P$ be a property. The following are equivalent:

- $P$ is $n$-c.e. in the Markov-model,
- $P$ is $n$-c.e. in the (Type-2,K)-model.
Better understanding Markov-semi-decidable sets?

Type-2-computability

Well-understood, equivalent to effective topology:

- Type-2-semi-decidable set = effective open set
- Type-2-computable function = effectively continuous function

Markov-computability

No such correspondence.

- Can we get a better understanding of Markov-computability?
- E.g., what do the Markov-semi-decidable properties look like?
Better understanding Markov-semi-decidable sets?

Effective Borel complexity.

**Theorem**

*Every Markov-semi-decidable property is $\Pi^0_2$.***

**Proof.**

The property is (Type-2,K)-semi-decidable, via a machine $M$. $M$ behaves the same on $(f, n)$ for all $n \geq K(f)$. As a result,

$$f \in P \text{ iff } \forall k, \exists n \geq k, \text{ the machine accepts } (f, n).$$

\[\Box\]
Better understanding Markov-semi-decidable sets?

Effective Borel complexity.

**Theorem**

*Every Markov-semi-decidable property is $\Pi^0_2$.***

**Proof.**

The property is *(Type-2,K)-semi-decidable*, via a machine $M$. $M$ behaves the same on $(f, n)$ for all $n \geq K(f)$. As a result,

$$ f \in P \text{ iff } \forall k, \exists n \geq k, \text{ the machine accepts } (f, n). $$

This is tight.

**Theorem**

*There is a Markov-semi-decidable property that is not $\Sigma^0_2$:*

$$ \forall n, Km(f|_n) < n + c. $$
Better understanding **Markov-semi-decidable sets**?

What do the **Markov-semi-decidable** properties look like?

- For total computable functions: open problem.
- For subrecursive classes: answer now!
Primitive recursive functions

What can be decided/semi-decided about a primitive recursive function \( f \), given a primitive recursive program for it?

**Example of Type-2-decidable property**

\[
f(3) = 9 \land f(4) = 16 \land f(5) = 25
\]
Primitive recursive functions

What can be decided/semi-decided about a primitive recursive function \( f \), given a primitive recursive program for it?

Example of **Type-2-decidable property**

\[
f(3) = 9 \land f(4) = 16 \land f(5) = 25
\]

Example of **Markov-decidable property**

\[
AC_h = \{ f : \forall n, K_{pr}(f|_n) < h(n) \}
\]
Primitive recursive functions

What can be decided/semi-decided about a primitive recursive function \( f \), given a primitive recursive program for it?

**Example of Type-2-decidable property**

\[
f(3) = 9 \land f(4) = 16 \land f(5) = 25
\]

**Example of Markov-decidable property**

\[
AC_h = \{ f : \forall n, K_{pr}(f \upharpoonright n) < h(n) \}
\]

**Theorem**

*That’s it!*
Primitive recursive functions

What can be decided/semi-decided about a primitive recursive function $f$, given a primitive recursive program for it?

Example of Type-2-decidable property

$$f(3) = 9 \land f(4) = 16 \land f(5) = 25$$

Example of Markov-decidable property

$$AC_h = \{ f : \forall n, K_{pr}(f|_n) < h(n) \}$$

Theorem

That’s it! All the Markov-semi-decidable properties are unions of cylinders and sets $AC_h$.

Idem for FPTIME, provably total functions, etc.

Fails for the class of all total computable functions.
<table>
<thead>
<tr>
<th>The problem</th>
<th>Historical results</th>
<th>New results</th>
<th>Limits</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>The problem</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Historical results</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>New results</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Limits</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
“The only extra information shared by programs computing an object is bounding its Kolmogorov complexity.”

True to a large extent
See previous results.

Not always true
See next results.
Relativization

Does the result hold relative to any oracle?

- On partial functions, NO.
- On total functions, YES.
Properties of **partial** functions.

**Reminder: Rice-Shapiro theorem**

\[
\text{Markov-}\text{semi-decidable} \iff (\text{Type-2,K})\text{-semi-decidable} \\
\iff \text{Type-2-semi-decidable}
\]

However,

**Proposition**

\[
\text{Markov-}\text{semi-decidable}^0' \implies (\text{Type-2,K})\text{-semi-decidable}^0'
\]
\[
(\text{Type-2,K})\text{-semi-decidable}^0'' \implies \text{Type-2-semi-decidable}^0''
\]
Relativization

Properties of total functions.

Theorem

For each oracle $A \subseteq \mathbb{N}$,

$$\text{Markov-semi-decidable}^A \iff \text{(Type-2,K)-semi-decidable}^A$$

There are two cases, whether $A$ computes $\emptyset'$ or not.

Theorem

There is no uniform argument.
**Computable functions**

**Reminder**

Let \( X, Y \) be **countably-based** topological spaces and \( f : X \to Y \).

\[
f \text{ is Markov}-\text{computable} \iff f \text{ is (Type-2,K)-computable}.
\]

Still holds if \( Y \) is not countably-based? For instance,

\[
Y = \{\text{open subsets of } \mathbb{N}^\mathbb{N}\}.
\]
Computable functions

Reminder

Let $X, Y$ be countably-based topological spaces and $f : X \rightarrow Y$.

$f$ is Markov-computable $\iff f$ is (Type-2,K)-computable.

Still holds if $Y$ is not countably-based? For instance,

$Y = \{\text{open subsets of } \mathbb{N}^\mathbb{N}\}$.

- When $X = \{\text{partial functions}\}$, NO.
- When $X = \{\text{total functions}\}$, open question.
Future work

- What are the Markov-semi-decidable properties of total functions?
- Precise limits of the equivalence $\text{Markov} \equiv (\text{Type-2}, K)$?
- If a property is $\omega$-c.e. in the Markov model, is it $\omega$-c.e. in the $(\text{Type-2}, K)$ model?
- The objects always lived in effective topological spaces. What about other represented spaces? For instance, the computable functionals from $\mathbb{N}^\mathbb{N}$ to $\mathbb{N}^\mathbb{N}$?