On the information carried by programs about the objects they compute

Mathieu Hoyrup

LORIA - Inria, Nancy (France)

joint work with Cristóbal Rojas (Santiago)

The problem

Two ways of providing a computable function $f : \mathbb{N} \to \mathbb{N}$ 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 **semidecide** something about f).

	Decidability	semidecidability
Partial functions		
Total functions		

The problem

Historical results

New results

Limitations

Partial functions

	Decidability	semidecidability
Partial functions	?	
Total functions		

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

	Decidability	semidecidability
Partial functions	$program \equiv graph$	
Total functions		

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

Answers

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

Theorem (Rice, 1953)

Given a program for f, a machine cannot do better.

Partial functions

	Decidability	semidecidability
Partial functions	$program \equiv graph$	$program \equiv graph$
Total functions		

What can be **semidecided** about f?

Answers

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

$$
\begin{aligned}
& (f(a_1) = u_1 \wedge \ldots \wedge f(a_i) = u_i) \\
\vee \quad & (f(b_1) = v_1 \wedge \ldots \wedge f(b_j) = v_j) \\
\vee \quad & (f(c_1) = w_1 \wedge \ldots \wedge f(c_k) = w_k) \\
\vee \quad & \ldots
\end{aligned}
$$

Theorem (Shapiro, 1956)

Given a program for f, a machine cannot do better.

Total functions

	Decidability	semidecidability
Partial functions	*program* $\equiv$ *graph*	*program* $\equiv$ *graph*
Total functions	*program* $\equiv$ *graph*	*program* $>$ *graph*

What can be **decided** about f?

Theorem (Kreisel-Lacombe-Schœnfield/Ceitin, 1957/1962)

For properties of total computable functions,

decidable from a program $\iff$ *decidable from the graph.*

What can be **semidecided** about f?

Theorem (Friedberg, 1958)

For properties of total computable functions,

semidecidable from a program $\not\Rightarrow$ *semidecidable from the graph.*

Friedberg's property

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

Figure: Friedberg's property, taken from the Rogers

Defined in 1958, but easier to define using **Kolmogorov complexity** (1960's).

- $K(n) = \min\{|p| : \text{program } p \text{ computes } n\}$.
- $K(n) \leq \log(n) + O(1)$.
- Say $n \in \mathbb{N}$ is **compressible** if $K(n) < \log(n)$:
    - There are infinitely many incompressible numbers.
    - Whether n is compressible is semidecidable.

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

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

Semideciding $f \in P$

n	0	1	2	3	4	5	6	...
$f(n)$	0	0	0	0	0	0	0	

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.

	Decidability	semidecidability
Partial functions	Markov ≡ Type-2 *Rice*	Markov ≡ Type-2 *Rice-Shapiro*
Total functions	Markov ≡ Type-2 *Kreisel-Lacombe-Shœnfield/Ceitin*	Markov > Type-2 *Friedberg*

Many other results by Selivanov, Spreen, Grassin, Korovina, Kudinov and others.

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

A bound on the Kolmogorov complexity of f!

We define

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

Theorem

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

- *$f \in P$ is Markov-semidecidable,*
- *$f \in P$ is Type-2-semidecidable 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^ω, $\mathbb{R}$, any effective topological space)
- many computability notions other than semidecidability
  (computable functions, n-c.e. properties, Σ^0_2 properties).

We now give 2 such results.

More general results

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

$$f \text{ is Markov-computable } \not\Longrightarrow f \text{ is Type-2-computable.}$$

However,

Theorem (Computable functions)

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

More general results

Theorem (Selivanov, 1984)

For properties of partial functions,

$$2\text{-}c.e.\ \textit{in the}\ \text{Markov}\text{-}model \not\Longrightarrow 2\text{-}c.e.\ \textit{in the}\ \text{Type-2}\text{-}model.$$

However,

Theorem

$$n\text{-}c.e.\ \textit{in the}\ \text{Markov}\text{-}model \iff n\text{-}c.e.\ \textit{in the}\ (\text{Type-2},\text{K})\text{-}model.$$

Better understanding Markov-computability?

- Now the relation between Markov-computability and Type-2-computability is more clear.
- Can we better understand Markov-computability?

Remark

Type-2-computability is well-understood: equivalent to effective topology.

- Type-2-semidecidable property $\equiv$ effective open set (Σ^0_1)
- Type-2-computable function $\equiv$ effectively continuous function

We now investigate the following question:

What do the Markov-semidecidable properties look like?

Complexity of Markov-semidecidable properties

Theorem

Every Markov-semidecidable property is Π_2^0*.*

Proof.

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

$$f \in P \iff \forall k, \exists n \geq k, M \text{ accepts } (f, n). \qquad \square$$

This is tight.

Theorem

There is a Markov-semidecidable property that is not Σ_2^0*:*

$$\forall n, K(f\restriction_n) < n + c.$$

The shape of Markov-semidecidable properties

What do Markov-semidecidable properties look like?

- On $\mathbb{N}^{\mathbb{N}}$, open question.
- On $\mathbb{N}_\infty$, complete answer.
- On the class of primitive recursive functions, complete answer.

The shape of Markov-semidecidable properties

Space of objects : $\mathbb{N}_\infty = \mathbb{N} \cup \{\infty\}$. A program p:

- computes ∞ if p outputs $0000000000\ldots$,
- computes n if p outputs $\underbrace{00\ldots0}_{n}1\ldots$.

Examples of Type-2-semidecidable properties

- Singletons: e.g. $\{6\}$,
- Semi-lines: e.g. $[10, \infty]$,

Examples of Markov-semidecidable properties

- Friedberg's set $F = \{n \in \mathbb{N} : K(n) < \log(n)\} \cup \{\infty\}$,
- More generally $F_h = \{n \in \mathbb{N} : K(n) < h(n)\} \cup \{\infty\}$.

Theorem

That's it!

The shape of Markov-semidecidable properties

Space of objects : primitive recursive functions. Here, **only primitive recursive programs** are allowed.

Example of Type-2-decidable property

A cylinder:

$$f(2) = 4 \quad \wedge \quad f(3) = 9 \quad \wedge \quad f(4) = 16$$

Example of Markov-decidable property

$$\forall n, K_{pr}(f\restriction_n) < h(n)$$

Theorem

They generate all the Markov-semidecidable properties.

Idem for FPTIME, provably total functions, etc.

The shape of Markov-semidecidable properties

On the class of total computable functions,

Type-2-semidecidable properties

The effective open sets.

Example of Markov-semidecidable property

$$\forall n, K(f\restriction_n) < h(n)$$

Theorem

They do not generate all the Markov-semidecidable properties.

"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 main result holds relative to any oracle?

- On partial functions, NO.
- On total functions, YES.

Relativization

Properties of **partial** functions.

Reminder: Rice-Shapiro theorem

$$\text{Markov-semidecidable} \iff \text{(Type-2,K)-semidecidable}$$
$$\iff \text{Type-2-semidecidable}$$

However,

Proposition

$$\text{Markov-}\textit{semidecidable}^{\emptyset'} \not\Longrightarrow \text{(Type-2,K)-}\textit{semidecidable}^{\emptyset'}$$
$$\text{(Type-2,K)-}\textit{semidecidable}^{\emptyset''} \not\Longrightarrow \text{Type-2-}\textit{semidecidable}^{\emptyset''}$$

Relativization

Properties of **total** functions.

Theorem

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

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

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 $f : X \to Y$.

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

Does it still hold if Y 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-semidecidable properties of total functions?
- Precise limits of the equivalence Markov≡(Type-2,K)?
- Does the implication hold?

  ω-c.e. in the Markov model $\implies$ ω-c.e. in the (Type-2,K) model?
- The objects always lived in countably-based topological spaces. What about other represented spaces? For instance, $\mathbb{N}^{\mathbb{N}^{\mathbb{N}}}$?

Thank you for your attention!

Proof of the main result

Theorem

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

- $f \in P$ *is* Markov*-semidecidable,*
- $f \in P$ *is* Type-2*-semidecidable given any upper bound on* $K(f)$.

Proof: main ingredient

Let P be a property of total computable functions containing $\lambda x.0$.

- If P is Type-2-semi-decidable then
  $$\exists n, \forall g, [g(0) = \ldots = g(n) = 0 \text{ implies } g \in P],$$
  and n can be computed.

- If P is Markov-semi-decidable then
  $$\underline{\forall g, \exists n}, [g(0) = \ldots = g(n) = 0 \text{ implies } g \in P],$$
  and n can be computed from a program for g.

- As a result for all k,
  $$\exists n, \forall g \ \underline{\text{s.t. } K(g) \leq k}, [g(0) = \ldots = g(n) = 0 \text{ implies } g \in P],$$
  and n can be computed from k.

Proof: main ingredient

Let P be a property of total computable functions containing $\lambda x.0$. Assume that P is Markov-semi-decidable.

Lemma

One has $\forall g, \exists n$ *s.t.* $[g(0) = \ldots = g(n) = 0$ *implies* $g \in P]$, *and* n *can be computed from a program for* g.

Proof.

Let M be the machine Markov-semideciding P. Define a program p:

$$p(t) = \begin{cases} 0 & \text{if } M(p) \text{ does not halt within } t \text{ steps,} \\ g(t) & \text{otherwise.} \end{cases}$$

- $M(p)$ must halt.
- Taking $n =$ halting time of $M(p)$ works. □

Bonus: let's play

Game

- Player: tries to guess a number n.
- Opponent: produces *in some way* a list of all the programs that eventually print n.

Version 0 (warm-up)

The opponent simply writes down the list of programs.

The player has a winning strategy: wait for a program "`print i`", then announce $n = i$.

Bonus: let's play

Game

- Player: tries to guess a number n.
- Opponent: produces *in some way* a list of all the programs that eventually print n.

Version 1 (Type-2)

The opponent writes down a list of programs and is allowed to remove some of them later (definitively). The list is what remains.

The player does not have a winning strategy.

Bonus: let's play

Game

- Player: tries to guess a number n.
- Opponent: produces *in some way* a list of all the programs that eventually print n.

Version 2 (Markov)

Idem, but the opponent is a program, known by the player.

The player has a winning strategy. For each $i \in \mathbb{N}$, it is possible to define a program p_i that prints only i and *will not* be removed by the opponent.
The strategy is as before: wait for a program p_i, then announce $n = i$.

p_i is defined this way: print i and if p_i is eventually removed by the opponent, print every $j \in \mathbb{N}$.

Bonus: let's play

Game

- Player: tries to guess a number n.
- Opponent: produces *in some way* a list of all the programs that eventually print n.

Version 3 (Type-2,K)

Again the opponent is a program. The player just has an upper bound on its size.

The player has a winning strategy.

Let k be the upper bound. Define programs $p_{i,j}$ that print i and if program j eventually halts, prints every natural number.
The strategy is: look for i such that $p_{i,j}$ appears for every $j \leq k$, then announce $n = i$.