On the information carried by programs about the objects they compute

Mathieu Hoyrup and Cristóbal Rojas

LORIA - Inria, Nancy (France)

Loria
Laboratoire lorrain de recherche en informatique et ses applications

Inria
informatiques mathématiques

The problem

Let p be a program. Two possible types of access to p:

(i) Running p.

(ii) Reading the code of p.

Having the code of p enables one to execute p, but not vice-versa.

The problem

Let p be a program. Two possible types of access to p:

(i) Running p.

(ii) Reading the code of p.

Having the code of p enables one to execute p, but not vice-versa.

Main questions

- Does it make a difference?
- Does the code of a program give more information about what it computes?

Halting problem

Running p, one can only semi-decide whether p halts.

Halting problem

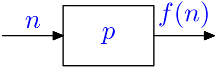
Running p, one can only semi-decide whether p halts.

Theorem (Turing, 1936)

Reading the code of p, a computer cannot do better.

Rice theorem

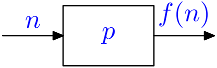
A program p computes a partial function f.



What can be **decided** about f?

Rice theorem

A program p computes a partial function f.



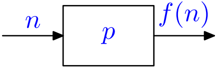
What can be **decided** about f?

Answer

Running p, only trivial properties: the decision about $\lambda x.\bot$ applies to every f.

Rice theorem

A program p computes a partial function f.



What can be **decided** about f?

Answer

Running p, only trivial properties: the decision about $\lambda x.\bot$ applies to every f.

Theorem (Rice, 1953)

Reading the code of p, a computer cannot do better.

Rice-Shapiro theorem

A program p computes a partial function f.

$$\xrightarrow{\;n\;} \boxed{\;p\;} \xrightarrow{\;f(n)\;}$$

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

Rice-Shapiro theorem

A program p computes a partial function f.

$$\xrightarrow{n} \boxed{p} \xrightarrow{f(n)}$$

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

Answer

Running p, exactly the properties of the form:

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

Rice-Shapiro theorem

A program p computes a partial function f.

$$\xrightarrow{\;n\;} \boxed{\;p\;} \xrightarrow{\;f(n)\;}$$

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

Answer

Running p, exactly the properties of the form:

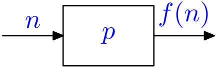
$$\begin{array}{ll} & (f(a_1) = u_1 \wedge \ldots \wedge f(a_i) = u_i) \\ \vee & (f(b_1) = v_1 \wedge \ldots \wedge f(b_j) = v_j) \\ \vee & (f(c_1) = w_1 \wedge \ldots \wedge f(c_k) = w_k) \\ \vee & \ldots \end{array}$$

Theorem (Rice-Shapiro, 1956)

Reading the code of p, a computer cannot do better.

Kreisel-Lacombe-Schœnfield/Ceitin theorem

Now assume that program p computes a **total** function f.



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

Kreisel-Lacombe-Schœnfield/Ceitin theorem

Now assume that program p computes a **total** function f.

$$\xrightarrow{\;n\;} \boxed{\;p\;} \xrightarrow{f(n)}$$

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

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

For properties of total computable functions,

read-*decidable* $\iff$ run-*decidable.*

Kreisel-Lacombe-Schœnfield/Ceitin theorem

Now assume that program p computes a **total** function f.

$$\xrightarrow{n} \boxed{\;p\;} \xrightarrow{f(n)}$$

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

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

For properties of total computable functions,

read-*decidable* $\iff$ run-*decidable.*

It makes a difference!

Theorem (Friedberg, 1958)

For properties of total computable functions,

read-*semi-decidable* $\not\Longrightarrow$ run-*semi-decidable.*

Sum up

Two computation models: read[1] and run[2].

Class of functions	Decidability	Semi-decidability
Partial	read $\equiv$ run *Rice*	read $\equiv$ run *Rice-Shapiro*
Total	read $\equiv$ run *Kreisel-Lacombe-Schœnfield/Ceitin*	read $>$ run *Friedberg*

[1] usually called Markov computability
[2] usually called Type-2 computability

Sum up

Two computation models: read[1] and run[2].

Class of functions	Decidability	Semi-decidability
Partial	read $\equiv$ run *Rice*	read $\equiv$ run *Rice-Shapiro*
Total	read $\equiv$ run *Kreisel-Lacombe-Schœnfield/Ceitin*	read $>$ run *Friedberg*

Let's now look at Friedberg's example.

[1] usually called Markov computability
[2] usually called Type-2 computability

Kolmogorov complexity

Introduced by Solomonoff (1960), Kolmogorov (1965), Chaitin (1966).

- Let $K(n) = \min\{|p| :$ program p computes $n\}$.
- $K(n) \leq \log(n) + O(1)$.
- n is **compressible** if $K(n) < \log(n)$.
- There are infinitely many incompressible numbers.
- Inequality $K(n) \leq k$ is semi-decidable.

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 $f \in P$

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

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

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

Semi-deciding $f \in P$

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

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 $f \in P$

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

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 $f \in P$

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

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 $f \in P$

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

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 $f \in P$

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

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 $f \in P$

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

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 $f \in P$

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

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 $f \in P$

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

When is it time to accept f?

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 $f \in P$

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

When is it time to accept f?

- If f is given by running p, 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 $f \in P$

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

When is it time to accept f?

- If f is given by running p, we can never know.
- If f is given by the code of p then evaluate f up to $2^{|p|}$.

Let x be an object. All the programs computing x share some common information about x:

- The information needed to recover x,
- Plus some extra information about x.

Question

What is the extra information?

Let x be an object. All the programs computing x share some common information about x:

- The information needed to recover x,
- Plus some extra information about x.

Question

What is the extra information?

Answer

They bound the Kolmogorov complexity of x!

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 read-semi-decidable,*
- *$f \in P$ is run-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
  (real numbers, subsets of $\mathbb{N}$*, points of a countably-based topological space)*
- many computability notions other than semi-decidability
  (computable functions, n*-c.e. properties,* Σ_2^0 *properties).*

More general results

The result is much more general and holds for:

- many classes of objects other than total functions
  (real numbers, subsets of $\mathbb{N}$, points of a countably-based topological space)
- many computability notions other than semi-decidability
  (computable functions, n-c.e. properties, Σ^0_2 properties).

For instance,

Theorem (Computable functions)

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

$$f \text{ is read-computable} \iff f \text{ is (run,K)-computable.}$$

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* read*-model,*
- *not* 2*-c.e. (and not even* Π^0_2*) in the* run*-model.*

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* read*-model,*
- *not* 2*-c.e. (and not even* Π^0_2*) in the* run*-model.*

Again,

Theorem

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

- P *is* n*-c.e. in the* read*-model,*
- P *is* n*-c.e. in the* (run,K)*-model.*

Applications

Effective Borel complexity of semi-decidable properties

Theorem

Every property that is read-semi-decidable is Π^0_2.

Applications

Effective Borel complexity of semi-decidable properties

Theorem

Every property that is read-semi-decidable is Π^0_2.

This is tight.

Theorem

There is a read-semi-decidable property of binary sequences that is not Σ^0_2.

$$x \in P \text{ iff } \forall n, K(x_0 \ldots x_{n-1}) < \log(n).$$

Applications

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

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

Examples of run-semi-decidable sets

- Singleton $\{n\}$, $n \in \mathbb{N}$,
- Semi-line $[n, \infty]$, $n \in \mathbb{N}$,

Applications

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

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

Examples of run-semi-decidable sets

- Singleton $\{n\}$, $n \in \mathbb{N}$,
- Semi-line $[n, \infty]$, $n \in \mathbb{N}$,

Examples of read-semi-decidable sets

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

Applications

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

- computes ∞ if p outputs $0000000000\dots$,
- computes n if p outputs $\underbrace{00\dots0}_{n}1\dots$.

Examples of run-semi-decidable sets

- Singleton $\{n\}$, $n \in \mathbb{N}$,
- Semi-line $[n, \infty]$, $n \in \mathbb{N}$,

Examples of read-semi-decidable sets

- 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!

A Rice-like theorem for primitive recursive functions

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

Example of run-decidable property

$$f(0) = 1 \wedge f(1) = 2 \wedge f(2) = 4$$

A Rice-like theorem for primitive recursive functions

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

Example of run-decidable property

$$f(0) = 1 \wedge f(1) = 2 \wedge f(2) = 4$$

Example of read-decidable property

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

A Rice-like theorem for primitive recursive functions

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

Example of run-decidable property

$$f(0) = 1 \wedge f(1) = 2 \wedge f(2) = 4$$

Example of read-decidable property

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

Theorem

That's it!

Idem for FPTIME, provably total functions, etc.

"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 holds relative to any oracle?

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

Relativization

Properties of **partial** functions.

Reminder: Rice-Shapiro theorem

$$\begin{aligned}\text{read-semi-decidable} &\iff \text{(run,K)-semi-decidable}\\ &\iff \text{run-semi-decidable}\end{aligned}$$

However,

Proposition

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

$$\begin{aligned}\text{read-}\textit{semi-decidable}^A &\not\Rightarrow \text{(run,K)-}\textit{semi-decidable}^A \quad \textit{(when } A \textit{ computes Halt)}\\ &\not\Rightarrow \text{run-}\textit{semi-decidable}^A \quad \textit{(when } A \textit{ computes Tot)}\end{aligned}$$

Relativization

Properties of **total** functions.

Theorem

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

$$\text{read-}\textit{semi-decidable}^A \iff \text{(run,K)-}\textit{semi-decidable}^A$$

There are two cases, whether A computes Halt 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 is read-computable $\iff$ f is (run,K)-computable.

What about non-countably-based spaces?

Theorem

Equivalence is broken for some Y.

Computable functions

Reminder

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

$$f \text{ is read-computable} \iff f \text{ is (run,K)-computable.}$$

What about non-countably-based spaces?

Theorem

Equivalence is broken for some Y*.*

Open question

What about X?

Future work

- What are the read-semi-decidable properties of total functions?
- Precise limits of the equivalence read≡(run,K)?
- The objects always lived in effective topological spaces. What about other spaces?