PHIL 222
Philosophical Foundations of Computer Science
Week 5, Thursday

Sept. 26, 2024

The Church-Turing Thesis: Turing's Arguments

So how is the thesis justified, by Church and Turing, or by others?

So how is the thesis justified, by Church and Turing, or by others?

Computer-science textbooks usually give

- The "argument from confluence":

  the various characterizations of computability, while differing in their approaches and formal details, turn out to encompass the very same class of computable functions.

So how is the thesis justified, by Church and Turing, or by others?

Computer-science textbooks usually give

- The "argument from confluence":

  the various characterizations of computability, while differing in their approaches and formal details, turn out to encompass the very same class of computable functions.

- The "argument from non-refutation":

  the thesis has never been refuted, despite sustained (and ongoing) attempts to find a counterexample.

E.g., R. Harper, *Practical Foundations for Programming Languages*, p. 188:

> Church's Law states that any conceivable notion of computable function on the natural numbers is equivalent to the λ-calculus. This assertion is true for all known means of defining computable functions on the natural numbers. The force of Church's Law is that it postulates that all future notions of computation will be equivalent in expressive power (measured by definability of functions on the natural numbers) to the λ-calculus. Church's Law is therefore a scientific law in the same sense as, say, Newton's Law of Universal Gravitation, which predicts the outcome of all future measurements of the acceleration in a gravitational field.

Turing's argument "I", one that he thinks is "[a] direct appeal to intuition" (quotes are from Turing 1936, pp. 249–251):

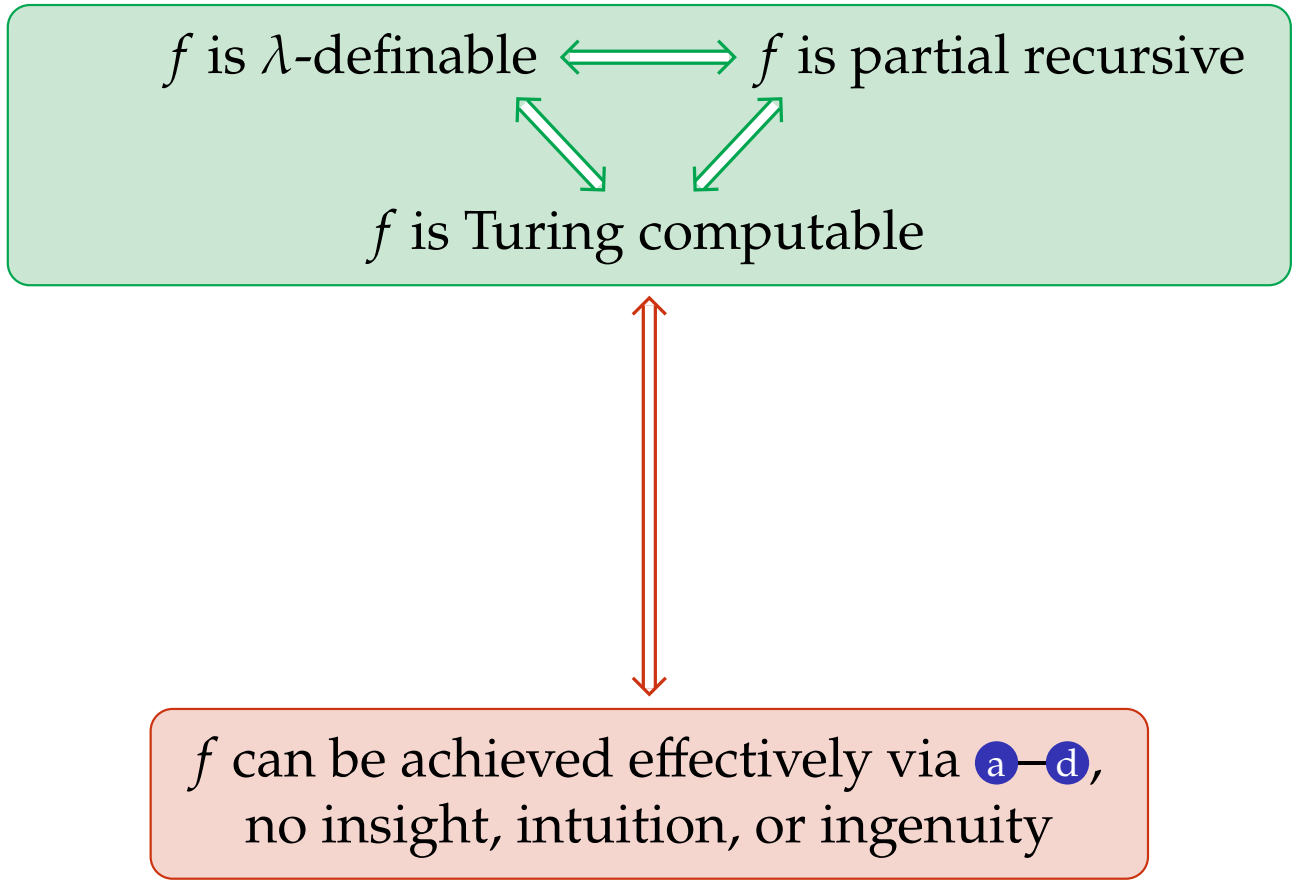
Turing's argument "I", one that he thinks is "[a] direct appeal to intuition" (quotes are from Turing 1936, pp. 249–251): Let's assume

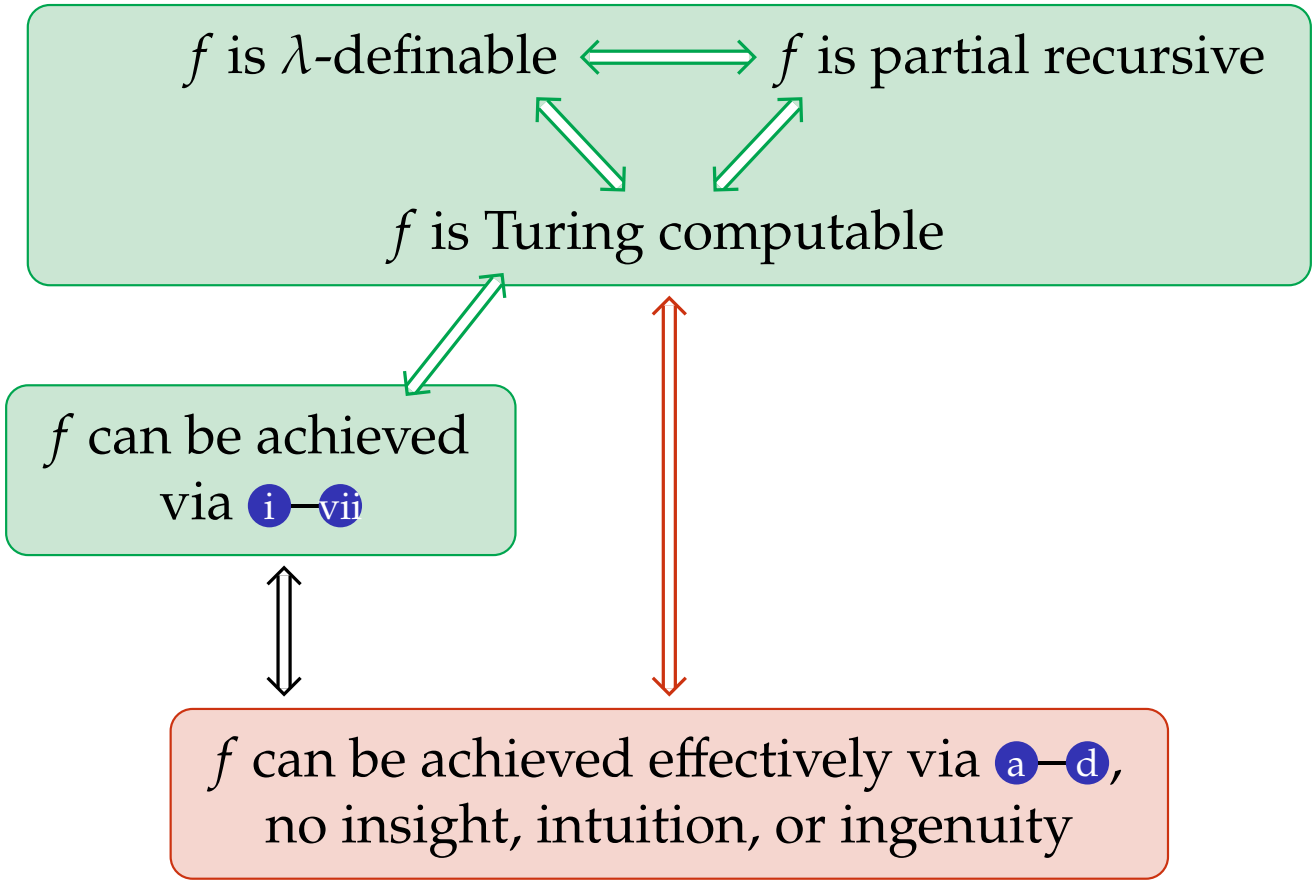
(i) [T]he two-dimensional character of paper is no essential of computation. [. . .] [T]he computation is carried out on one-dimensional paper, i.e. on a tape divided into squares. [. . .] (ii) [T]he number of symbols which may be printed is finite. [. . .] (iii) The behavior of the computer at any moment is determined by the symbols which he is observing, and his "state of mind" at that moment. [. . .] (iv) [T]here is a bound B to the number of symbols or squares which the computer can observe at one moment. [. . .] (v) [T]he number of states of mind which need be taken into account is finite. [. . .] (vi) [I]n a simple operation not more than one symbol is altered. [. . .] (vii) [E]ach of the new observed squares is within L squares of an immediately previously observed square.

Turing's argument "I", one that he thinks is "[a] direct appeal to intuition" (quotes are from Turing 1936, pp. 249–251): Let's assume

> (i) [T]he two-dimensional character of paper is no essential of computation. [. . .] [T]he computation is carried out on one-dimensional paper, i.e. on a tape divided into squares. [. . .] (ii) [T]he number of symbols which may be printed is finite. [. . .] (iii) The behavior of the computer at any moment is determined by the symbols which he is observing, and his "state of mind" at that moment. [. . .] (iv) [T]here is a bound B to the number of symbols or squares which the computer can observe at one moment. [. . .] (v) [T]he number of states of mind which need be taken into account is finite. [. . .] (vi) [I]n a simple operation not more than one symbol is altered. [. . .] (vii) [E]ach of the new observed squares is within L squares of an immediately previously observed square.

Then "We may now construct a [Turing] machine to do the work of this [human] computer" who is subject to (i)–(vii).


f is λ-definable
f is partial recursive
f is Turing computable
f can be achieved effectively via a–d,
no insight, intuition, or ingenuity


f is λ-definable
f is partial recursive
f is Turing computable
f can be achieved via i–vii
f can be achieved effectively via a–d, no insight, intuition, or ingenuity

Turing's argument "II", in a nutshell:

To compute $f(n) \longrightarrow m$ is to provide a step-by-step proof that $f(n) = m$.

Turing's argument "II", in a nutshell:

To compute $f(n) \longrightarrow m$ *is to provide a step-by-step proof that* $f(n) = m$.

- Find a formula $\varphi(x_1, \ldots, x_k, y)$ in the language of arithmetic that describes the function $f(x_1, \ldots, x_k) = y$.

Turing's argument "II", in a nutshell:

To compute $f(n) \longrightarrow m$ *is to provide a step-by-step proof that* $f(n) = m$.

- Find a formula $\varphi(x_1, \ldots, x_k, y)$ in the language of arithmetic that describes the function $f(x_1, \ldots, x_k) = y$.
- When $f(n_1, \ldots, n_k) = m$, we may be able to find a proof
  1. Axioms of arithmetic: $x + sy = s(x + y)$, $x \times 0 = 0$, etc.

     $\vdots$

  ℓ. m is the one and only number satisfying $\varphi(n_1, \ldots, n_k, m)$.

Turing's argument "II", in a nutshell:

To compute $f(n) \to m$ *is to provide a step-by-step proof that* $f(n) = m$.

- Find a formula $\varphi(x_1, \ldots, x_k, y)$ in the language of arithmetic that describes the function $f(x_1, \ldots, x_k) = y$.
- When $f(n_1, \ldots, n_k) = m$, we may be able to find a proof
  1. Axioms of arithmetic: $x + sy = s(x + y)$, $x \times 0 = 0$, etc.

     $\vdots$

  ℓ. m is the one and only number satisfying $\varphi(n_1, \ldots, n_k, m)$.

Definition. We say that φ "represents" f to mean that

$$\text{such a proof as above exists} \iff f(n_1, \ldots, n_k) = m.$$

Turing's argument "II", in a nutshell:

To compute $f(n) \to m$ *is to provide a step-by-step proof that* $f(n) = m$.

- Find a formula $\varphi(x_1, \ldots, x_k, y)$ in the language of arithmetic that describes the function $f(x_1, \ldots, x_k) = y$.
- When $f(n_1, \ldots, n_k) = m$, we may be able to find a proof
  1. Axioms of arithmetic: $x + sy = s(x + y)$, $x \times 0 = 0$, etc.

     $\vdots$

  ℓ. m is the one and only number satisfying $\varphi(n_1, \ldots, n_k, m)$.

Definition. We say that φ "represents" f to mean that

$$\text{such a proof as above exists} \iff f(n_1, \ldots, n_k) = m.$$

Theorem (Gödel)**.** Given any function f,

$$\text{some formula } \varphi \text{ represents } f \iff f \text{ is partial recursive}$$

Turing's argument "II", in a nutshell:

To compute $f(n) \to m$ *is to provide a step-by-step proof that* $f(n) = m$.

- Find a formula $\varphi(x_1, \ldots, x_k, y)$ in the language of arithmetic that describes the function $f(x_1, \ldots, x_k) = y$.
- When $f(n_1, \ldots, n_k) = m$, we may be able to find a proof
  1. Axioms of arithmetic: $x + sy = s(x + y)$, $x \times 0 = 0$, etc.

     $\vdots$

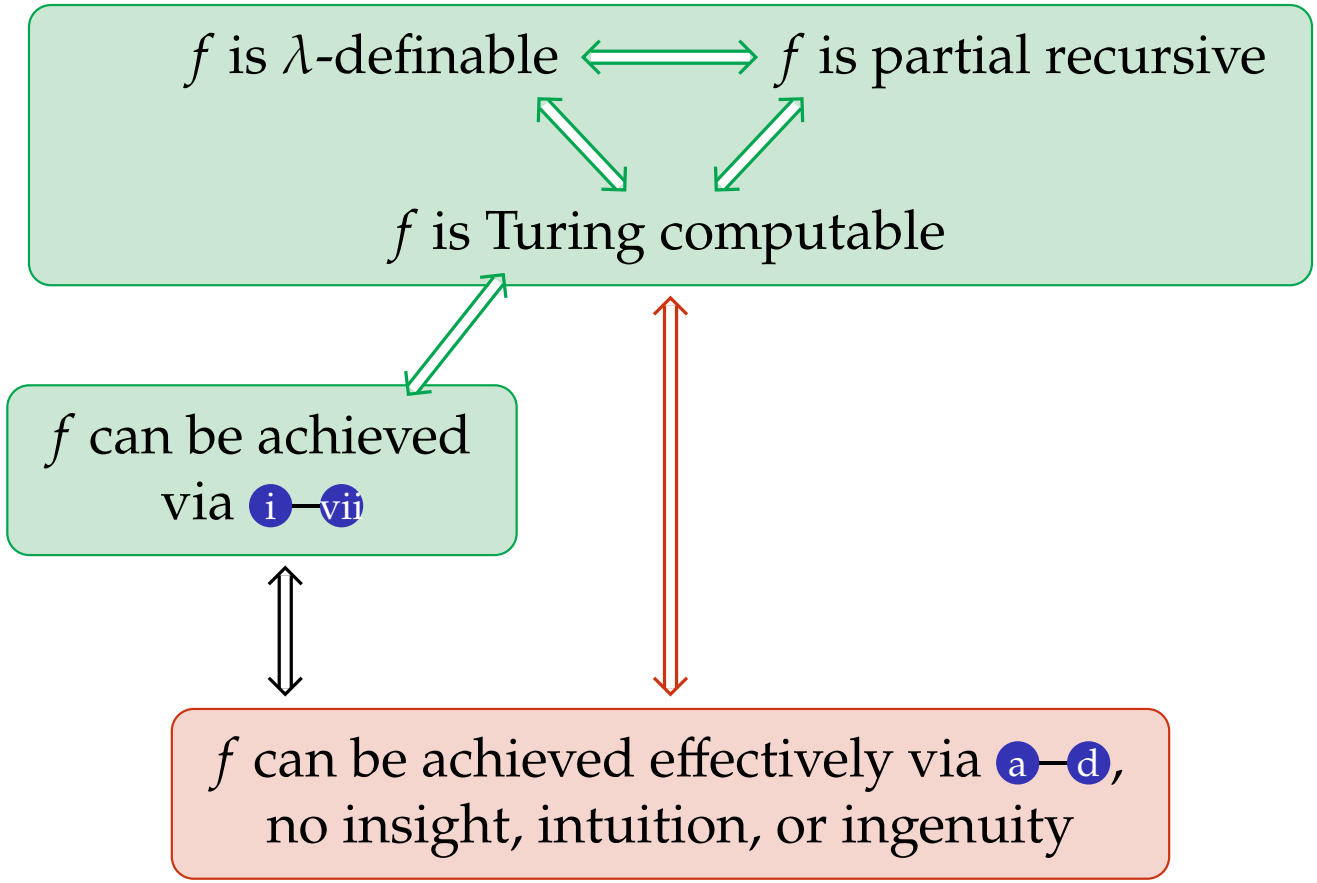
  ℓ. m is the one and only number satisfying $\varphi(n_1, \ldots, n_k, m)$.

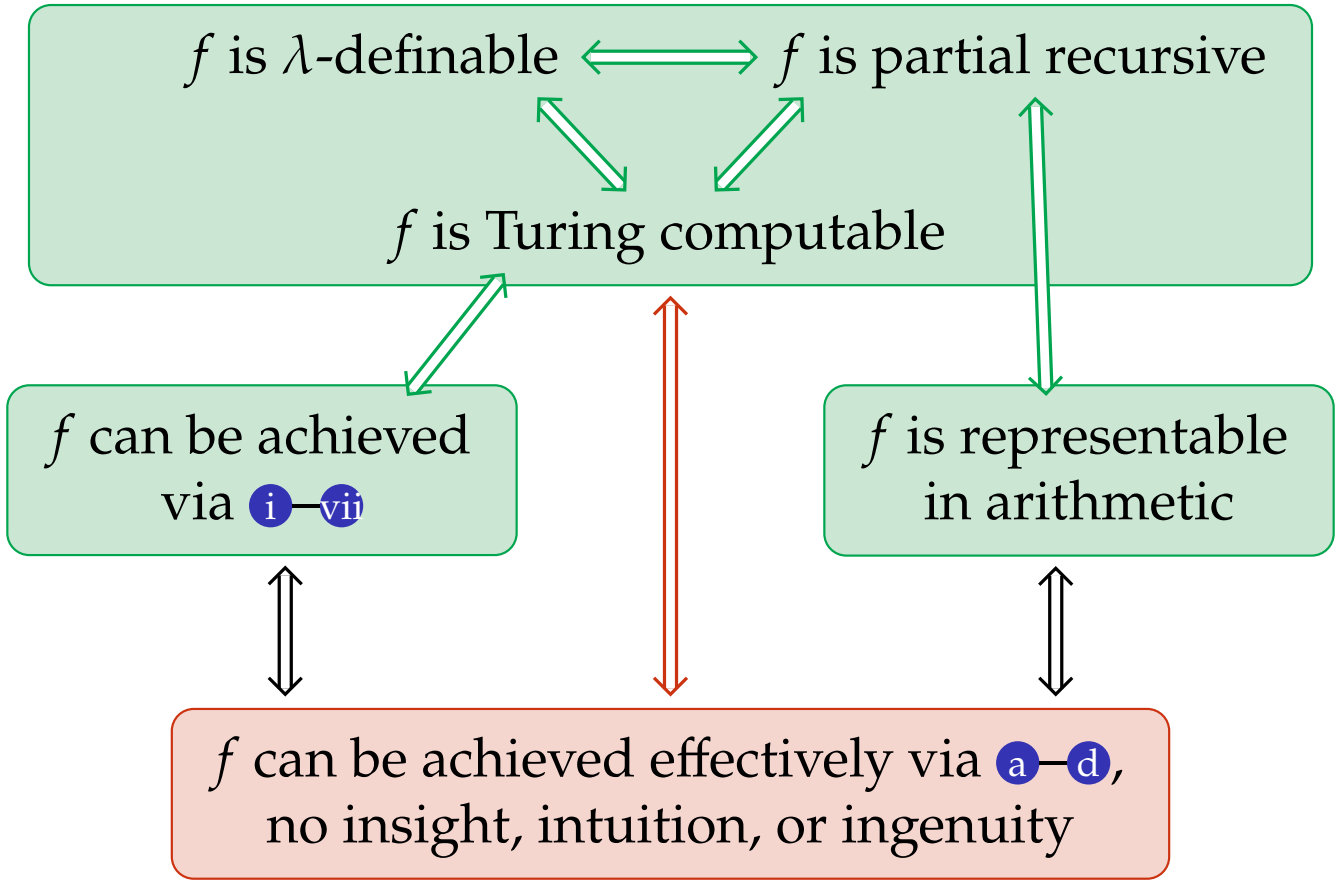
Definition. We say that φ "represents" f to mean that

$$\text{such a proof as above exists} \iff f(n_1, \ldots, n_k) = m.$$

Theorem (Gödel)**.** Given any function f,

$$\begin{aligned}\text{some formula } \varphi \text{ represents } f &\iff f \text{ is partial recursive} \\ &\iff f \text{ is Turing computable.}\end{aligned}$$


f is λ-definable
f is partial recursive
f is Turing computable
f can be achieved via i–vii
f can be achieved effectively via a–d, no insight, intuition, or ingenuity


f is λ-definable
f is partial recursive
f is Turing computable
f can be achieved via i–vii
f is representable in arithmetic
f can be achieved effectively via a–d, no insight, intuition, or ingenuity

What is the nature of Turing's arguments I and II?
Are they mathematical proofs of the Church-Turing thesis?

What is the nature of Turing's arguments I and II?
Are they mathematical proofs of the Church-Turing thesis?

- Argument II hinges on a mathematical theorem.

  f is representable in arithmetic $f \iff f$ is Turing computable.

- The part of argument I showing

  f can be computed by a method satisfying i–vii

  $\implies f$ can be computed by a Turing machine

  can also be mathematical.

What is the nature of Turing's arguments I and II?
Are they mathematical proofs of the Church-Turing thesis?

- Argument II hinges on a mathematical theorem.

  f is representable in arithmetic $f \iff f$ is Turing computable.

- The part of argument I showing

  f can be computed by a method satisfying i–vii

  $\implies f$ can be computed by a Turing machine

  can also be mathematical.

Do these mean that arguments I and II are mathematical proofs of the Church-Turing thesis?

What is the nature of Turing's arguments I and II?
Are they mathematical proofs of the Church-Turing thesis?

- Argument II hinges on a mathematical theorem.

  f is representable in arithmetic $f \iff f$ is Turing computable.

- The part of argument I showing

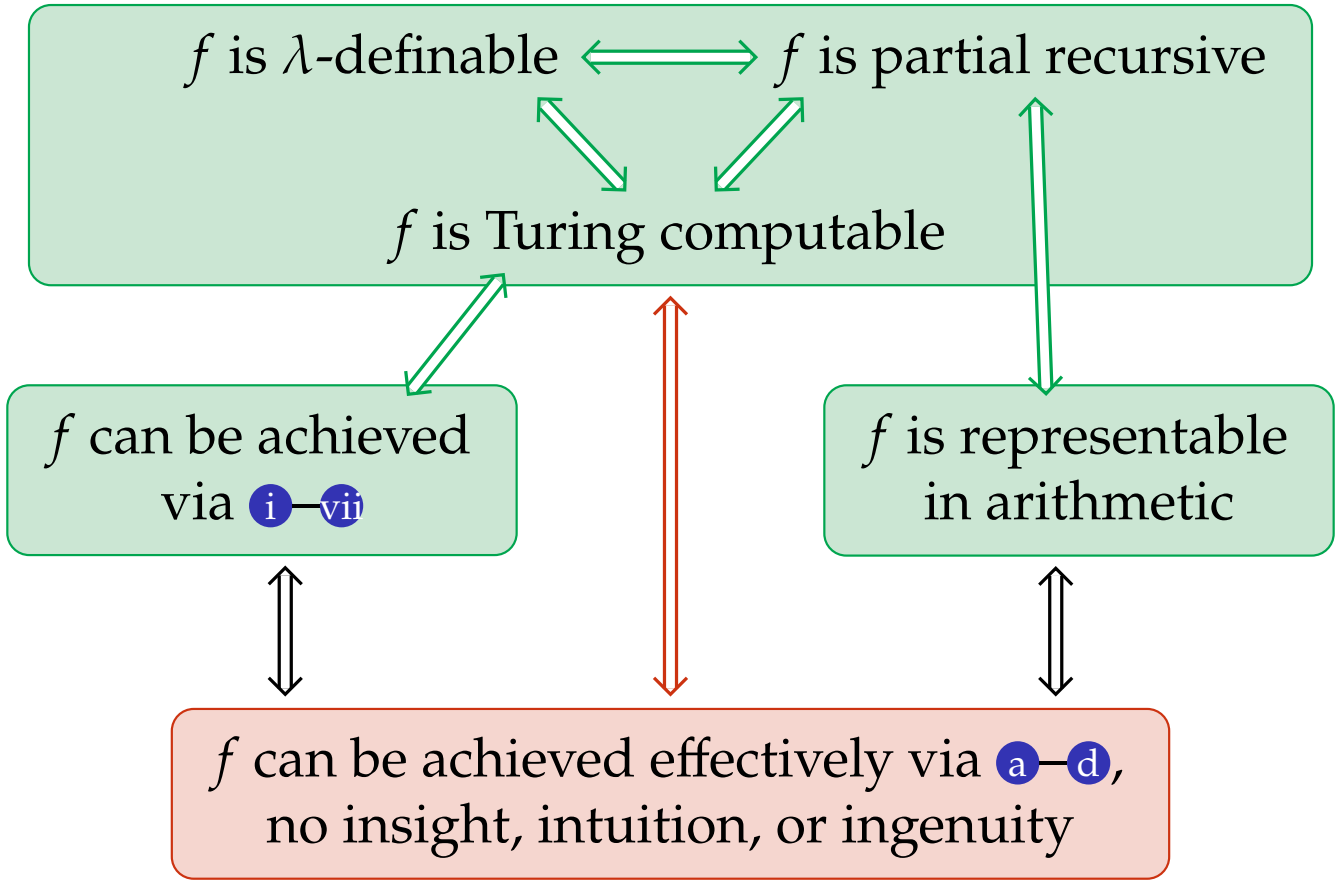
  f can be computed by a method satisfying i–vii

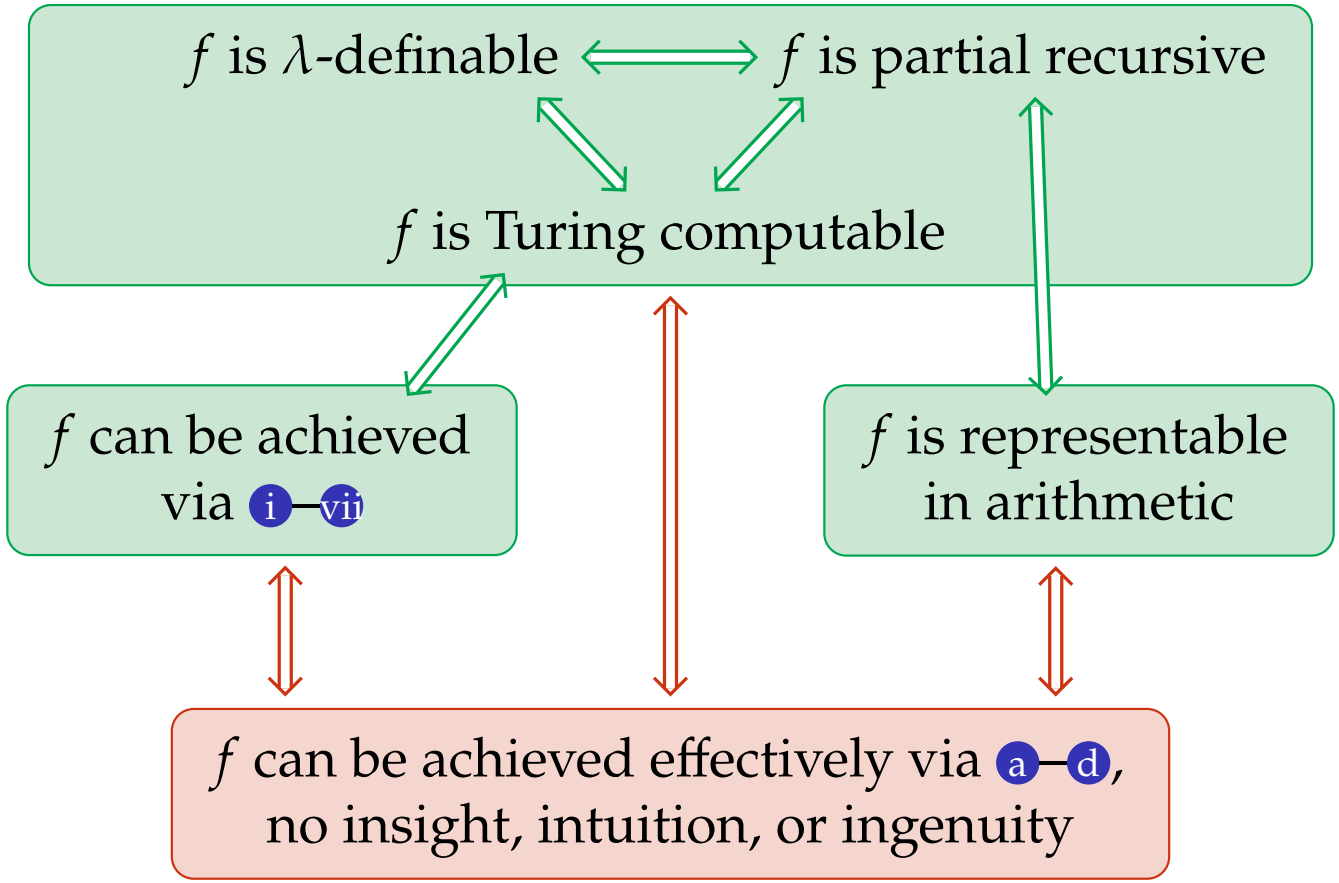
  $\implies f$ can be computed by a Turing machine

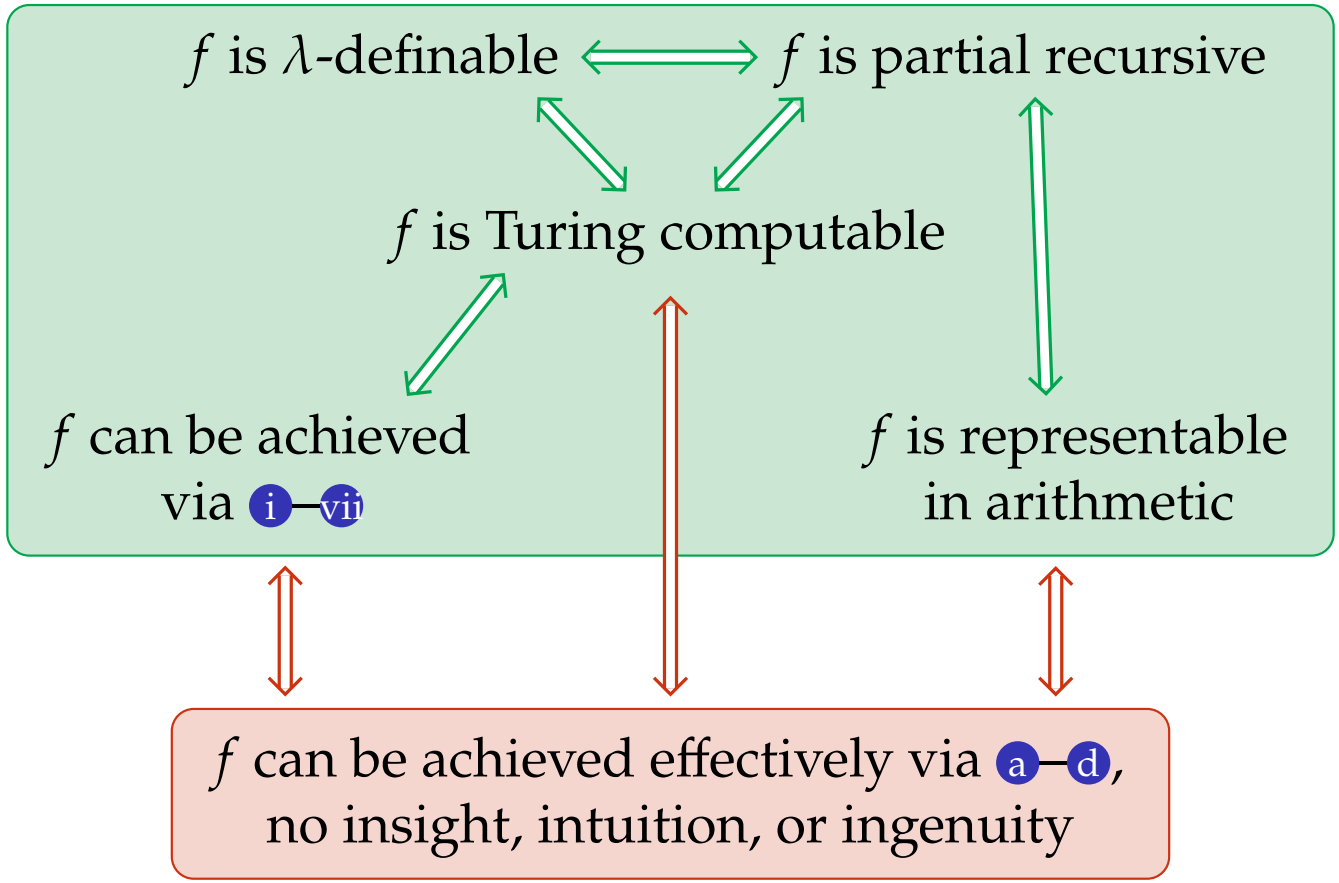
  can also be mathematical.

Do these mean that arguments I and II are mathematical proofs of the Church-Turing thesis?

Some philosophers think so, although Church and Turing did not (neither does Copeland).


f is λ-definable
f is partial recursive
f is Turing computable
f can be achieved via i–vii
f is representable in arithmetic
f can be achieved effectively via a–d, no insight, intuition, or ingenuity


f is λ-definable
f is partial recursive
f is Turing computable
f can be achieved via i–vii
f is representable in arithmetic
f can be achieved effectively via a–d, no insight, intuition, or ingenuity


f is λ-definable
f is partial recursive
f is Turing computable
f can be achieved via i–vii
f is representable in arithmetic
f can be achieved effectively via a–d, no insight, intuition, or ingenuity

The Church-Turing Thesis: Other Versions

"Non-human computers" variants of the Church-Turing thesis.

$$f \text{ "can be computed"} \iff f \text{ is Turing computable}$$

What should we mean by "can be computed"?

"Non-human computers" variants of the Church-Turing thesis.

$$f \text{ "can be computed"} \iff f \text{ is Turing computable}$$

What should we mean by "can be computed"? We may try:

I. "Machines can generate f by taking x in and outputting $f(x)$."

"Non-human computers" variants of the Church-Turing thesis.

$$f \text{ "can be computed"} \iff f \text{ is Turing computable}$$

What should we mean by "can be computed"? We may try:

I. "Machines can generate f by taking x in and outputting $f(x)$."

II. "Algorithms can generate f by taking x in and outputting $f(x)$."

"Non-human computers" variants of the Church-Turing thesis.

$$f \text{ "can be computed"} \iff f \text{ is Turing computable}$$

What should we mean by "can be computed"? We may try:

Ⅰ "Machines can generate f by taking x in and outputting $f(x)$."

Ⅱ "Algorithms can generate f by taking x in and outputting $f(x)$."

But, for Ⅰ, what type of machines to consider?

"Non-human computers" variants of the Church-Turing thesis.

$$f \text{ "can be computed"} \iff f \text{ is Turing computable}$$

What should we mean by "can be computed"? We may try:

- (I) "Machines can generate f by taking x in and outputting $f(x)$."
- (II) "Algorithms can generate f by taking x in and outputting $f(x)$."

But, for (I), what type of machines to consider? We may try:

- (A) "Computing machines", implementing (a non-human version of) an effective method, M, satisfying (a)–(d).
  - (a) M is set out in terms of a finite number of exact instructions, each expressed by means of a finite number of symbols;
  - (c) M can (in practice or in principle) be carried out by a human being unaided by any machinery except paper and pencil;
  - (d) M demands no insight, intuition, or ingenuity, on the part of the human being carrying out the method.

"Non-human computers" variants of the Church-Turing thesis.

$$f \text{ "can be computed"} \iff f \text{ is Turing computable}$$

What should we mean by "can be computed"? We may try:

- (I) "Machines can generate f by taking x in and outputting $f(x)$."
- (II) "Algorithms can generate f by taking x in and outputting $f(x)$."

But, for (I), what type of machines to consider? We may try:

- (A) "Computing machines", implementing (a non-human version of) an effective method, M, satisfying (a)–(d).

"Non-human computers" variants of the Church-Turing thesis.

$$f \text{ "can be computed"} \iff f \text{ is Turing computable}$$

What should we mean by "can be computed"? We may try:

- (I) "Machines can generate f by taking x in and outputting $f(x)$."
- (II) "Algorithms can generate f by taking x in and outputting $f(x)$."

But, for (I), what type of machines to consider? We may try:

- (A) "Computing machines", implementing (a non-human version of) an effective method, M, satisfying (a)–(d).
- (B) "Computing machines", satisfying (a non-human version of) (i)–(vii).
  - (ii) The number of symbols which may be printed is finite.
  - (iii) The behavior of the computer at any moment is determined by the symbols which he is observing, and his "state of mind" at that moment.

"Non-human computers" variants of the Church-Turing thesis.

$$f \text{ "can be computed"} \iff f \text{ is Turing computable}$$

What should we mean by "can be computed"? We may try:

- (I) "Machines can generate f by taking x in and outputting $f(x)$."
- (II) "Algorithms can generate f by taking x in and outputting $f(x)$."

But, for (I), what type of machines to consider? We may try:

- (A) "Computing machines", implementing (a non-human version of) an effective method, M, satisfying (a)–(d).
- (B) "Computing machines", satisfying (a non-human version of) (i)–(vii).

"Non-human computers" variants of the Church-Turing thesis.

$$f \text{ "can be computed"} \iff f \text{ is Turing computable}$$

What should we mean by "can be computed"? We may try:

- (I) "Machines can generate f by taking x in and outputting $f(x)$."
- (II) "Algorithms can generate f by taking x in and outputting $f(x)$."

But, for (I), what type of machines to consider? We may try:

- (A) "Computing machines", implementing (a non-human version of) an effective method, M, satisfying (a)–(d).
- (B) "Computing machines", satisfying (a non-human version of) (i)–(vii).
- (C) Any machine ...

"Non-human computers" variants of the Church-Turing thesis.

$$f \text{ "can be computed"} \iff f \text{ is Turing computable}$$

What should we mean by "can be computed"? We may try:

- (I) "Machines can generate f by taking x in and outputting $f(x)$."
- (II) "Algorithms can generate f by taking x in and outputting $f(x)$."

But, for (I), what type of machines to consider? We may try:

- (A) "Computing machines", implementing (a non-human version of) an effective method, M, satisfying (a)–(d).
- (B) "Computing machines", satisfying (a non-human version of) (i)–(vii).
- (C) Any machine that is physically possible or implementable.
- (D) Any conceivable machine, regardless of the physical laws.

"Non-human computers" variants of the Church-Turing thesis.

$$f \text{ "can be computed"} \iff f \text{ is Turing computable}$$

What should we mean by "can be computed"? We may try:

Ⓘ "Machines can generate f by taking x in and outputting $f(x)$."

Ⓘⓘ "Algorithms can generate f by taking x in and outputting $f(x)$."

But, for Ⓘ, what type of machines to consider? We may try:

Ⓐ "Computing machines", implementing (a non-human version of) an effective method, M, satisfying ⓐ–ⓓ.

Ⓑ "Computing machines", satisfying (a non-human version of) ⓘ–ⓥⓘ.

Ⓒ Any machine that is physically possible or implementable.

- Does any physical process count?

Ⓓ Any conceivable machine, regardless of the physical laws.

"Non-human computers" variants of the Church-Turing thesis.

$$f \text{ "can be computed"} \iff f \text{ is Turing computable}$$

What should we mean by "can be computed"? We may try:

Ⓘ "Machines can generate f by taking x in and outputting $f(x)$."

Ⓘⓘ "Algorithms can generate f by taking x in and outputting $f(x)$."

But, for Ⓘ, what type of machines to consider? We may try:

Ⓐ "Computing machines", implementing (a non-human version of) an effective method, M, satisfying ⓐ–ⓓ.

Ⓑ "Computing machines", satisfying (a non-human version of) ⓘ–ⓥⓘ.

Ⓒ Any machine that is physically possible or implementable.

- Does any physical process count?
- Is the universe a computer?

Ⓓ Any conceivable machine, regardless of the physical laws.

For ⓫, what exactly is an algorithm?

For II, what exactly is an algorithm? We may try:

E. A process (abstractly speaking) that implements (an indendent-of-human version of) an effective method satisfying a–d.

F. A process (abstractly speaking) that satisfies (an indendent-of-human version of) i–vii.

For II, what exactly is an algorithm? We may try:

- E A process (abstractly speaking) that implements (an indendent-of-human version of) an effective method satisfying a–d.
- F A process (abstractly speaking) that satisfies (an indendent-of-human version of) i–vii.

A/E: *f can be computed by machines / algorithms that implements an effective method satisfying a–d $\iff$ f is Turing computable*

may be equivalent to the original Church-Turing thesis.

For II, what exactly is an algorithm? We may try:

- E A process (abstractly speaking) that implements (an indendent-of-human version of) an effective method satisfying a–d.
- F A process (abstractly speaking) that satisfies (an indendent-of-human version of) i–vii.

A/E: *f can be computed by machines / algorithms that implements an effective method satisfying a–d $\iff$ f is Turing computable*

may be equivalent to the original Church-Turing thesis.

B/F: *f can be computed by machines / algorithms that satisfies i–vii $\iff$ f is Turing computable*

may be mathematically provable.

For (II), what exactly is an algorithm? We may try:

- (E) A process (abstractly speaking) that implements (an indendent-of-human version of) an effective method satisfying (a)–(d).
- (F) A process (abstractly speaking) that satisfies (an indendent-of-human version of) (i)–(vii).

(A)/(E): *f can be computed by machines / algorithms that implements an effective method satisfying (a)–(d)* $\iff$ *f is Turing computable*

may be equivalent to the original Church-Turing thesis.

(B)/(F): *f can be computed by machines / algorithms that satisfies (i)–(vii)* $\iff$ *f is Turing computable*

may be mathematically provable.

Let's investigate whether contemporary computer scientists undertand the Church-Turing thesis as (A)/(E)/(B)/(F).