Automata and Formal Languages - CM0081
The Church-Turing Thesis

Andrés Sicard-Ramírez

EAFIT University

Semester 2015-2
Turing Machine Computable Functions

Number-theoretical functions

\[ \{ f \mid f : \mathbb{N}^k \rightarrow \mathbb{N} \}. \]
Turing Machine Computable Functions

Number-theoretical functions

\[ \{ f \mid f : \mathbb{N}^k \rightarrow \mathbb{N} \}. \]

Examples

The following functions are Turing machine computable:

- \( z(x) = 0 \) (zero function)
- \( s(x) = x + 1 \) (successor function)
- \( \Pi_k(x_1, \ldots, x_n) = x_k \) (\( n \)-ary projection functions)
- \( i(x) = x \) (identity function)
- \( f_k(x) = k \) (\( k \)-constant function)
- \( x + y, xy, x^y \) (addition, multiplication and exponentiation)
Turing Machine Computable Functions

Examples

The following functions are Turing machine computable:

\[ x \div y = \begin{cases} x - y & \text{if } x \geq y, \\ 0 & \text{otherwise}, \end{cases} \]

\[ |x - y| = \begin{cases} x \div y & \text{if } x \geq y, \\ y \div x & \text{otherwise}, \end{cases} \]

\[ \text{sg}(x) = \begin{cases} 1 & \text{if } x = 0, \\ 0 & \text{otherwise}, \end{cases} \]

\[ p(x) = \begin{cases} 1 & \text{if } x \text{ is even}, \\ 0 & \text{otherwise}. \end{cases} \]
Common Versions of the Church-Turing Thesis

“A function is \textit{computable (effectively calculable)} if and only if there is a \textit{Turing machine which computes it}.”\footnote{Galton, Antony (2006). The Church-Turing Thesis: Still Valid after All These Years?, p. 94.}

\footnote{Hopcroft, John E., Motwani, Rajeev and Ullman, Jefferey D. (2007). Introduction to Automata theory, Languages, and Computation, p. 236.}
Common Versions of the Church-Turing Thesis

“A function is computable (effectively calculable) if and only if there is a Turing machine which computes it.”¹

“The unprovable assumption that any general way to compute will allow us compute only the partial-recursive functions (or equivalently, what Turing machines or modern-day computers can compute) is know as Church’s hypothesis or the Church-Turing thesis.”²


“We now define the notion, already discussed, of an effectively calculable function of positive integers by identifying it with the notion of a recursive function of positive integers (or of a $\lambda$-definable function of positive integers). This definition is thought to be justified by considerations which follow, so far as positive justification can ever be obtained for the selection of a formal definition to correspond to intuitive notion.”

---

Alan Turing: A Definition

“The ‘computable’ numbers\textsuperscript{4} include all numbers which would naturally be regarded as computable.”\textsuperscript{5}

\textsuperscript{4}The number whose decimal representation can be generating progressively by a Turing machine.

\textsuperscript{5}Turing, Alan M. (1936). On Computable Numbers, with an Application to the Entscheidungsproblem, p. 249.
Stephen Kleene: Church’s thesis and Turing’s thesis

“The thesis of Church and Turing were not even called ‘thesis’ at all until Kleene (1943, p. 60)\textsuperscript{6} referred to Church’s ‘definition’ as ‘Thesis I’ and the 1952 Kleene\textsuperscript{7} referred to ‘Church’s Thesis’ and ‘Turing’s Thesis’.”\textsuperscript{8}

\textsuperscript{6}Kleene, S. C. (1943). Recursive Predicates and Quantifiers.
\textsuperscript{7}Kleene, Stephen C. (1952). Introduction to Metamathematics.
Stephen Kleene: The Church-Turing Thesis

“The term ‘Church-Turing thesis’ seems to have been first introduce by Kleene, with a small flourish of bias in favor of Church.”\(^9\)

“So Turing’s and Church’s thesis are equivalent. We shall usually refer to them both as Church’s thesis, or in connection with that one of its…version which deal with ‘Turing machines’ as the Church-Turing thesis.”\(^{10}\)

Intensional-Extensional Meaning

“Here we also use the phrase ‘Church-Turing thesis’ to refer to the amalgamation of the two theses (these and others) where we identify all informal concepts of Definition 1.1\textsuperscript{11} with one another we identify all the formal concepts of Definition 1.2\textsuperscript{12}, and their mathematical equivalents, with one another and suppress their intensional meanings.”\textsuperscript{13}

\textsuperscript{11}Definition 1.1: A function is ‘computable’ (also called ‘effectively calculable’ or simply ‘calculable’) if it can be calculated by a finite mechanical procedure.

\textsuperscript{12}Definition 1.2: (i) A function is ‘Turing computable’ if it is definable by a Turing machine, as defined by Turing 1936.

Possible Refutation

Idea: Turing machine computability ⇏ effective calculability

“A function is considered effectively computable if its value can be computed in an effective way in a finite number of steps, but there is no bound on the number of steps required for any given computation. Thus, the fact that there are effectively computable functions which may not be humanly computable has nothing to do with Church’s thesis.”¹⁴

Possible Refutation

Idea: Effective calculability $\iff$ Turing machine computability (the interesting one!)

From a Church’s letter to Pepis (June 8, 1937):

“Therefore to discover a function which was effectively calculable but no general recursive would imply discovery of an utterly new principle of logic, not only never before formulated, but never before actually used in a mathematical proof...Moreover this new principle of logic must be of so strange, and presumably complicated,...I should be inclined to scrutinize the alleged effective applicability of the principle with considerable care.”\textsuperscript{15}

\textsuperscript{15}Sieg, Wilfred (1997). Step by Recursive Step: Church’s Analysis of Effective Calculability, pp. 175–176.
A Precise Version of the Church-Turing Thesis

Turing’ analysis: Features of computations performed by human computers

- “States of mind” \(\Rightarrow\) Finite number of states
A Precise Version of the Church-Turing Thesis

Turing’ analysis: Features of computations performed by human computers

- “States of mind” $\Rightarrow$ Finite number of states
- A human cannot reliably discriminate infinitely many symbols $\Rightarrow$ Finite alphabet
A Precise Version of the Church-Turing Thesis

Turing’ analysis: Features of computations performed by human computers

- “States of mind” $\Rightarrow$ Finite number of states
- A human cannot reliably discriminate infinitely many symbols $\Rightarrow$ Finite alphabet
- Unlimited sheets of paper $\Rightarrow$ Unbounded tape
A Precise Version of the Church-Turing Thesis

Turing' analysis: Features of computations performed by human computers

- “States of mind” $\Rightarrow$ Finite number of states
- A human cannot reliably discriminate infinitely many symbols $\Rightarrow$ Finite alphabet
- Unlimited sheets of paper $\Rightarrow$ Unbounded tape
- The human read/write symbols on the paper $\Rightarrow$ Read/Write head
A Precise Version of the Church-Turing Thesis

Turing’ analysis: Features of computations performed by human computers

- “States of mind” ⇒ Finite number of states
- A human cannot reliably discriminate infinitely many symbols ⇒ Finite alphabet
- Unlimited sheets of paper ⇒ Unbounded tape
- The human read/write symbols on the paper ⇒ Read/Write head
- Human’s shift of attention form one part of the paper to another ⇒ Displacement of the read/write head
A Precise Version of the Church-Turing Thesis

Precise Church-Turing thesis

“Any procedure than can be carried out by an idealised human clerk working mechanically with paper and pencil can also be carried out by a Turing machine.”\textsuperscript{16}

Stronger Theses

Gandy’s theses

“Thesis P. The computable functions are all and only those computable by a discrete deterministic mechanical device.”

\[10\] Gandy, Robin (1980). Church’s Thesis and Principles for Mechanisms.
Stronger Theses

Gandy’s theses\textsuperscript{17}

\textit{“Thesis P. The computable functions are all and only those computable by a discrete deterministic mechanical device.”}

\textit{“Thesis M. What can be calculated by a machine is Turing machine computable.”}

\textsuperscript{17}Gandy, Robin (1980). Church’s Thesis and Principles for Mechanisms.
Stronger Theses

Physical Church-Turing thesis

“A function is computable by means of a physically possible computing device if and only if there is a Turing machine which computes it.”\textsuperscript{18}

\textsuperscript{18}Galton, Antony (2006). The Church-Turing Thesis: Still Valid after All These Years?, p. 95.
Stronger Theses

Current situation

- At the moment, it does not exists a refutation to the precise Church-Turing thesis.
Stronger Theses

Current situation

- At the moment, it does not exist a refutation to the precise Church-Turing thesis.
- The hypercomputation models refute the theoretical version of the thesis M.
Stronger Theses

Current situation

- At the moment, it does not exists a refutation to the precise Church-Turing thesis.
- The hypercomputation models refute the theoretical version of the thesis M.
- Open problem: the refutation of the realizable version of the thesis M (i.e. the physical Church-Turing thesis).
Example

Let $f$ and $\bot$ be a terminating and a non-terminating function from $a$ to $a$, respectively. The parallel or function

$$\text{por} :: (a \to a) \to (a \to a) \to a \to a$$

$$\text{por } f \bot = f$$

$$\text{por } \bot f = f$$

$$\text{por } \bot \bot = \bot$$

is an effectively calculable function which is not $\lambda$-definable.\(^{19}\)

\(^{19}\)Turner, David (2007). Church’s Thesis and Functional Programming.
Solution: To add extra primitives to $\lambda$-calculus (interleaving, concurrency, etc.).

---

Bonus Slides: Higher Type Computability

Solution: To add extra primitives to $\lambda$-calculus (interleaving, concurrency, etc.).

General problem: A definition of higher type computability.\(^{20}\)

\(^{20}\)Longley, John R. (2003). Notions of Computability at Higher Types I.
References


References


