

CS_236 Language and Computation

Course Notes

Part IV: Limits of Computation

Chapt. 4: The Church-Turing Thesis

Anton Setzer

[http://www.cs.swan.ac.uk/~csetzer/lectures/
languageComputation/09/index.html](http://www.cs.swan.ac.uk/~csetzer/lectures/languageComputation/09/index.html)

December 12, 2009

The Church-Turing Thesis

We have introduced two models of computations:

- ▶ The URM-computable functions.
- ▶ The Turing-computable functions.

Further we have indicated why the two models of computation compute the same partial functions.

The Church-Turing Thesis

Lots of other models of computation have been studied:

- ▶ The partial recursive functions.
- ▶ The while programs.
- ▶ Symbol manipulation systems by Post and by Markov.
- ▶ Equational calculi by Kleene and by Gödel.
- ▶ The λ -definable functions.
- ▶ Any of the programming languages Pascal, C, C++, Java, Prolog, Haskell, ML (and many more).
- ▶ Lots of other models of computation.

The Church-Turing Thesis

- ▶ One can show that the partial functions computable in these models of computation are again exactly the Turing computable functions.
- ▶ So all these attempts to define a complete model of computation result in the same set of partial recursive functions.
- ▶ Therefore we arrive at the Church Turing Thesis.

The Church-Turing Thesis

Church-Turing Thesis:

The (in an intuitive sense) computable partial functions are exactly the Turing-computable functions

*(or equivalently the URM-computable functions
or equivalently the functions computable in any other known
Turing-complete model of computation).*

Philosophical Thesis

- ▶ This thesis is **not a mathematical theorem**.
- ▶ It is a **philosophical thesis**.
- ▶ Therefore the Church-Turing thesis **cannot be proven**.
- ▶ We can only provide **philosophical evidence** for it.
- ▶ This evidence comes from the following **considerations and empirical facts**:

Empirical Facts

- ▶ All complete models of computation suggested by researchers define the same set of partial functions.
- ▶ Many of these models were carefully designed in order to capture intuitive notions of computability:
 - ▶ The Turing machine model captures the intuitive notion of **computation on a piece of paper** in a general sense.
 - ▶ The URM machine model captures the general notion of **computability by a computer**.
 - ▶ Symbolic manipulation systems capture the general notion of computability by **manipulation of symbolic strings**.

Empirical Facts

- ▶ No intuitively computable partial function, which is not partial recursive, has been found, despite lots of researchers trying it.
- ▶ A strong intuition has been developed that in principal programs in any programming language can be simulated by Turing machines and URM's.

Because of this, only few researchers doubt the correctness of the Church-Turing thesis.

Decidable Sets

- ▶ A predicate A is URM-/Turing-decidable iff χ_A is URM-/Turing-computable.
- ▶ A predicate A is decidable iff χ_A is computable.
- ▶ By Church's thesis to be computable is the same as to be URM-computable or to be Turing-computable.
- ▶ So the decidable predicates are exactly the URM-decidable and exactly the Turing-decidable predicates.

Halting Problem

- ▶ Because of the equivalence of the models of computation, the halting problem for any of the above mentioned models of computation is undecidable.
- ▶ Especially it is undecidable, whether a program in one of the programming languages mentioned terminates:
 - ▶ Assume we had a decision procedure for deciding whether or not say a Java program terminates for given input.
 - ▶ Then we could, using a translation of URMs into Java programs, decide the halting problem for URMs, which is impossible.