**cs3102: Theory of Computation**

**Class 15:**

**Church-Turing Thesis**

Spring 2010  
University of Virginia  
David Evans

---

**Turing Machine Recap**

\[(Q, \Sigma, \Gamma, \delta, q_0, q_{accept}, q_{reject})\]

- **Q**: finite set of states
- **\(\Sigma\)**: input alphabet, finite set of symbols (cannot include \(\sqcup\))
- **\(\Gamma\)**: tape alphabet, finite set of symbols (includes \(\Sigma\) and \(\sqcup\))
- **\(\delta\)**: transition function: \(Q \times \Gamma \rightarrow Q \times \Gamma \times \{L, R\}\)
- **q_0**: start state, \(q_0 \in Q\)
- **q_{accept}**: accepting state, \(q_{accept} \in Q\)
- **q_{reject}**: rejecting state, \(q_{reject} \subset Q\), \(q_{reject} \neq q_{accept}\)

---

**Defining \(\delta^* : \Gamma^* \times Q \times \Gamma^* \rightarrow \Gamma^* \times Q \times \Gamma^*\)**

\(\forall u, v \in \Gamma^*, a, b \in \Gamma, q \in Q:\)

if \(q \in \{q_{accept}, q_{reject}\}:\)

\[\delta^*(u, q_F, v) = (u, q_F, v)\]

else:

if \((q, b) = (q_r, c, L)\):

- not at edge: \(\delta^*(ua, q, bv) = \delta^*(u, q_r, acv)\)
- left edge: \(\delta^*(e, q, bv) = \delta^*(e, q_r, cv)\)

if \((q, b) = (q_r, c, R)\):

- \(v \neq e:\) \(\delta^*(u, q, bv) = \delta^*(uc, q_r, v)\)
- right edge: \(\delta^*(u, q, b) = \delta^*(uc, q_r, \sqcup)\)

---

**TM Computing Model**

A Turing Machine

\[M = (Q, \Sigma, \Gamma, \delta, q_0, q_{accept}, q_{reject})\]

accepts a string \(w \in \Sigma^*\) iff

\[\delta^*(e, q_0, w) = (\gamma_L, q_{accept}, \gamma_R)\]

for some \(\gamma_L, \gamma_R \in \Gamma^*\).

---

**Drawing Turing Machines**

- \(\delta(q_0, 0) = (q_1, 1, R)\)
- \(\delta(q_0, 0) = (q_1, 0, L)\)

*Note: one deterministic TM could not include both of these rules!*

---
Example TM

Is 00 in the language accepted by this TM?

Designing TMs

ADD = \{ 1^x 1^y 1^z \mid x, y \in N, z = x + y \}

Is ADD context-free?

MULT = \{ 1^x 1^y 1^z \mid x, y \in N, z = x \times y \}

Is MULT context-free?

Example 2

Is 00 in the language accepted by this TM?

One to try at home...*

Does it accept \( \epsilon \)?

TM Computing Model

If \( \delta^* (\epsilon, q_0, w) = (\gamma_L, q_{accept}, \gamma_R) \) the TM accepts \( w \).

If \( \delta^* (\epsilon, q_0, w) = (\gamma_L, q_{reject}, \gamma_R) \) the TM rejects \( w \).

* If you have a few spare millennia (or a very fast computer...)

Is this enough to know if \( w \) is in the language?
TM Execution Possible Outcomes

**Accept:** Running TM $M$ on input $w$ eventually leads to $q_{\text{accept}}$.

**Reject:** Running TM $M$ on input $w$ eventually leads to $q_{\text{reject}}$.

**No Decision:** Running TM $M$ on input $w$ runs forever, never reaches $q_{\text{accept}}$ or $q_{\text{reject}}$.

**Recognizing vs. Deciding**

**Turing-recognizable:** A language $L$ is “Turing-recognizable” if there exists a TM $M$ such that for all strings $w$:
- If $w \in L$: eventually $M$ enters $q_{\text{accept}}$.
- If $w \notin L$: either $M$ enters $q_{\text{reject}}$ or $M$ never terminates.

**Turing-decidable:** A language $L$ is “Turing-decidable” if there exists a TM $M$ such that for all strings $w$:
- If $w \in L$: eventually $M$ enters $q_{\text{accept}}$.
- If $w \notin L$: eventually $M$ enters $q_{\text{reject}}$.

**Decider vs. Recognizer?**

Deciders *always* terminate.

Recognizers can run forever without deciding.

**Decidable and Recognizable Languages**

Decidable languages are a subset of recognizable languages, which in turn are a subset of all languages.

**Diophantus (~200-~284)**

'Here lies Diophantus,' the wonder behold. Through art algebraic, the stone tells how old: 'God gave him his boyhood one-sixth of his life, One twelfth more as youth while whisksers grew ripe; And then yet one-seventh ere marriage begun; In five years there came a bouncing new son. Alas, the dear child of master and sage After attaining half the measure of his father’s life chill fate took him. After consoling his fate by the science of numbers for four years, he ended his life.'

$$K^Y \alpha \zeta \beta \land \dot{M}_\gamma x^3 + 2x - 3$$
Hilbert’s Tenth Problem (1900)

To devise a process according to which it can be determined in a finite number of operations whether a given Diophantine equation is solvable in rational integers.

David Hilbert (1862 – 1943)
University of Göttingen

No Ignorabimus

However unapproachable these problems may seem to us and however helpless we stand before them, we have, nevertheless, the firm conviction that their solution must follow by a finite number of purely logical processes. ... The conviction of the solvability of every mathematical problem is a powerful incentive to the worker. We hear within us the perpetual call: There is the problem. Seek its solution. You can find it by pure reason, for in mathematics there is no ignorabimus.

David Hilbert, 1900

The Entscheidungsproblem

Decision procedure: given a well-formed statement, determine whether it can be proven.

It is of fundamental importance for the character of this problem that only mechanical calculations according to given instructions, without any thought activity in the stricter sense, are admitted as tools for the proof. ... Perhaps one could later let the procedure be carried out by a machine.

Heinrich Behmann, 1921

Closely related, but stronger goal:

Truth procedure: given a well-formed statement, determine whether it is true.

Gödel proved this is impossible (1930)

Spring 1935: Max Newman’s class (ends with Gödel)

April 1936: On Computable Numbers, with an Application to the Entscheidungsproblem

Breaking Enigma

1946: ACE, first stored-program computer design (operational in 1950)
1950: Computing Machinery and Intelligence
1952: Convicted of “gross indecency”
1954: Committed suicide by eating cyanide apple

Alan Turing (1912-1954)
Turing was a quite brilliant mathematician, most famous for his work on breaking the German Enigma codes. It is no exaggeration to say that, without his outstanding contribution, the history of World War Two could well have been very different. ... The debt of gratitude he is owed makes it all the more horrifying, therefore, that he was treated so inhumanely. ... It is difficult to believe that in living memory, people could become so consumed by hate - by anti-Semitism, by homophobia, by xenophobia and other murderous prejudices ...

It is thanks to men and women who were totally committed to fighting fascism, people like Alan Turing, that the horrors of the Holocaust and of total war are part of Europe’s history and not Europe’s present. So on behalf of the British government, and all those who live freely thanks to Alan’s work I am very proud to say: we’re sorry, you deserved so much better.

Prime Minister’s Apology, Gordon Brown, September 2009

Turing’s Model

Turing needed a precise model of a mechanical procedure to prove what numbers can be computed.

“Computing is normally done by writing certain symbols on paper. We may suppose this paper is divided into squares like a child’s arithmetic book.”

“For the present I shall only say that the justification lies in the fact that the human memory is necessarily limited.”

-Alan Turing, On computable numbers, with an application to the Entscheidungsproblem, 1936

Turing’s Paper

“I give some arguments with the intention of showing that the computable numbers include all numbers which would naturally be regarded as computable. In particular, I show that certain large classes of numbers are computable...”

Argue that the model corresponds well to computation.

“The computable numbers do not, however, include all definable numbers, and an example is given of a definable number which is not computable. Although the class of computable numbers is so great, and in many ways similar to the class of real numbers, it is nevertheless enumerable. I examine certain arguments which would seem to prove the contrary. By the correct application of one of these arguments, conclusions are reached which are superficially similar to those of Gödel. These results have valuable applications. In particular, it is shown that the Hilbertian Entscheidungsproblem can have no solution.”

Show some numbers cannot be computed by the model.

We’ll look at Turing’s proof next week.

Resolving the Entscheidungsproblem

March 1936: it is unsolvable  May 1936: it is unsolvable

-Alonzo Church (1903-1995)
-Alan Turing (1912-1954)

Turing studied with Church, 1936-1938 at Princeton

Alonzo Church’s “Less Successful” PhD Students

Hartley Rogers
Raymond Smullyan
Michael Rabin
John Kemeny
Stephen Kleene
Dana Scott
Martin Davis

See http://www.genealogy.ams.org/id.php?id=8011 for full list
Church-Turing Thesis

As stated by Kleene:

Every effectively calculable function (effectively decidable predicate) is general recursive.

“Since a precise mathematical definition of the term effectively calculable (effectively decidable) has been wanting, we can take this thesis ... as a definition of it...”

Yes, this is circular: everything calculable can be computed by a TM, and we define what is calculable as what can be computed by a TM.

Church-Turing Thesis

• We can model any mechanical computer with a TM.
• Any mechanical computation can be performed by a Turing Machine.
• There is some TM corresponding to every computable problem.
• The set of languages that can be decided by a TM is identical to the set of languages that can be decided by any mechanical computing machine.
• If there is no TM that decides problem $P$, there is no algorithm that solves problem $P$.

All of these statements are implied by the Church-Turing thesis.

Examples

[Last class and PS4] TM and 2-stack deterministic PDA
[PS4] Making the tape infinite in both directions adds no power
[Soon] Adding a second tape adds no power
[Church] Lambda Calculus is equivalent to TM
[Chomsky] Unrestricted replacement grammars are equivalent to TM
[Takahara and Yokomori] DNA is as powerful as a TM

[Hotly Debated] Is the human brain equivalent to a TM?

The Most Bogus Sentence

“A Turning machine can do everything a real computer can do.”

On the first page!

Allison Light (1:07pm)
Nathan Case (1:13pm)
Kevin Leach
Things Real Computers Can Do

- Generate Heat
- Stop a Door
- Provide an adequate habitat for fish

Computational Thing Most Real Computers Can Do (that Turing Machines can’t)

- Generate randomness

Something More Powerful than TM

- With a pencil and compass, we can compute π exactly! No discrete computer can do this!

Charge

- PS4 Due Tuesday
- Next week:
  - Turing’s Proof
  - What languages cannot be decided by a TM?
  - What languages cannot be recognized by a TM?
- Read Chapter 4: Decidability
  - I don’t think it has any extremely bogus sentences, but if you find one send it to me…