COMPSCI/SFWRENG 2FA3

Discrete Mathematics with Applications II

Winter 2020

6 Turing Machines and Computability

William M. Farmer

Department of Computing and Software
McMaster University

March 14, 2020

Effective Methods

An effective method is a method for producing an output from a set of inputs such that:

1. The method consists of a series of steps in which each step results from executing a precise instruction.
2. Each instruction is expressed by a finite number of symbols.
3. The number of possible instructions is finite.
4. The method succeeds after a finite number of steps.
5. The method can be performed mechanically by a human or a machine.

Models of Computation

- A model of computation is a model that describes how an output is computed from a set of inputs.
- Examples:
  - Lambda calculus (Alonzo Church, 1936).
  - Turing machines (Alan Turing, 1937).
  - General recursive functions (Kurt Gödel 1931, Stephen Kleene, 1936).
  - Combinatory logic (Moses Schönfinkel, 1924, Haskell Curry, 1930).
  - Post systems (Emil Post, 1936).
  - Unlimited register machines (John Shepherdson, Howard Sturgis, 1963).
- All of these models are computationally equivalent!

Church-Turing Thesis

- The Church-Turing thesis states that any model of computation equivalent to those listed above captures exactly our intuition of what an effective method is.
- Church and Turing developed this thesis before there were modern computers!
- The Church-Turing thesis implies that any effective method can be implemented in any model of computation equivalent to those listed above.

The Great Limitation Theorems

- First Incompleteness Theorem (Kurt Gödel, 1931). No consistent, sufficiently strong, recursively axiomatizable proof system can prove all the truths of natural number arithmetic.
- Second Incompleteness Theorem (Kurt Gödel, 1931). No consistent, sufficiently strong, recursively axiomatizable proof system can prove its own consistency.
- Undefinability of Truth (Alfred Tarski, 1936). Truth cannot be defined in any sufficiently strong theory.
- Undecidability of First-Order Logic (Alonzo Church, 1936). Validity is undecidable in first-order logic.

Great Limitation Theorems (iClicker)

The great limitation theorems state that there are limits on what can be

A. Proved.
B. Computed.
C. Defined.
D. All of the above.

Robinson Arithmetic [1/2]

- Let $\Sigma = (\{\mathbb{N}\}, \{0\}, \{S, +, *\}, \emptyset, \tau)$ where $\tau(0) = \mathbb{N}$, $\tau(S) = \mathbb{N} \rightarrow \mathbb{N}$, and $\tau(+) = \tau(*) = \mathbb{N} \times \mathbb{N} \rightarrow \mathbb{N}$.
- Let Γ be the following set of Σ-formulas:
  1. $\forall x : \mathbb{N} \,.\, 0 \neq S(x)$.
  2. $\forall x, y : \mathbb{N} \,.\, (S(x) = S(y) \Rightarrow x = y)$.
  3. $\forall x : \mathbb{N} \,.\, x + 0 = x$.
  4. $\forall x, y : \mathbb{N} \,.\, x + S(y) = S(x + y)$.
  5. $\forall x : \mathbb{N} \,.\, x * 0 = 0$.
  6. $\forall x, y : \mathbb{N} \,.\, x * S(y) = (x * y) + x$.
  7. $\forall x : \mathbb{N} \,.\, x = 0 \vee \exists\, y : \mathbb{N} \,.\, S(y) = x$.
- Then $\mathrm{Q} = (\Sigma, \Gamma)$ is a theory of MSFOL called Robinson arithmetic.
  - Q is a subtheory of first-order Peano arithmetic.
  - Axiom 7 is used in place of the induction schema.

Robinson Arithmetic [2/2]

- Since Q is
  1. very likely to be consistent,
  2. "sufficiently strong", and
  3. recursively axiomatizable,

  the first three great limitation theorems apply to Q. The fourth also applies to Q.
- Metatheorems about Q:
  1. Q is finitely axiomatizable.
  2. Q is incomplete and every consistent recursively axiomatizable extension of Q is incomplete (i.e., Q is essentially incomplete).
  3. Consistency is not provable in Q nor in any consistent recursively axiomatizable extension of Q.
  4. Truth is not definable in Q.
  5. Q is undecidable and every consistent extension of Q is undecidable (i.e., Q is essentially undecidable).

Turing Machine: Informal Description

- A deterministic, one-tape Turing machine (TM) has the following components:
  1. Tape:

$\vdash$	a_0	a_1	a_2	$\cdots$	a_n	$\sqcup$	$\sqcup$	$\sqcup$	$\cdots$

     $\uparrow$
  2. State: $\boxed{\text{q}} \in Q$.
  3. Program: $\boxed{\delta}$, a transition function.
- The tape is two way, read/write, and semi-infinite.
  - Tape is used for input, output, and memory.
  - The input string $a_0 a_1 a_2 \cdots a_n$ is finite.
  - Tape is an unbounded, sequentially accessed memory.
- The program is deterministic.
  - Takes the current tape symbol and state as input.
  - Produces a new tape symbol and state and left/right tape position change as output.

Turing Machines: Formal Definition [1/4]

- A deterministic, one-tape Turing machine (TM) is a tuple $M = (Q, \Sigma, \Gamma, \vdash, \llcorner\!\lrcorner, \delta, s, t, r)$ where:
  1. Q is a finite set of elements called states.
  2. Σ is a finite of symbols called the input alphabet.
  3. $\Gamma \supseteq \Sigma$ is a finite of symbols called the tape alphabet.
  4. $\vdash \in \Gamma \setminus \Sigma$ is the left endmarker.
  5. $\llcorner\!\lrcorner \in \Gamma \setminus \Sigma$ is the blank symbol.
  6. $\delta : Q \times \Gamma \rightarrow Q \times \Gamma \times \{L, R\}$ is the transition function.
  7. $s \in Q$ is the start state.
  8. $t \in Q$ is the accept state.
  9. $r \in Q$ is the reject state.
- For all $p \in Q, b \in \Gamma$, δ must satisfy the conditions:
  1. $\delta(p, \vdash) = (q, \vdash, R)$ for some $q \in Q$
  2. $\delta(t, b) = (t, c, d)$ for some $c \in \Gamma$ and $d \in \{L, R\}$.
  3. $\delta(r, b) = (r, c, d)$ for some $c \in \Gamma$ and $d \in \{L, R\}$.

Example 1

- Let $M = (Q, \Sigma, \Gamma, \vdash, \textvisiblespace, \delta, s, t, r)$ be the TM where:

  $Q = \{s, q, t, r\}$.
  $\Sigma = \{a, b\}$.
  $\Gamma = \Sigma \cup \{\vdash, \textvisiblespace\}$.
  δ is defined by the following table:

	$\vdash$	a	b	$\textvisiblespace$
s	$(s, \vdash, R)$	(s, a, R)	(q, b, R)	$(t, \textvisiblespace, R)$
q	$(q, \vdash, R)$	(r, a, R)	(q, b, R)	$(t, \textvisiblespace, R)$
t	$(t, \vdash, R)$	(t, a, R)	(t, b, R)	$(t, \textvisiblespace, R)$
r	$(r, \vdash, R)$	(r, a, R)	(r, b, R)	$(r, \textvisiblespace, R)$

- M accepts the regular language $\{a^m b^n \mid m, n \geq 0\}$.

Acceptance and Rejection (iClicker)

A Turing machine will either accept or reject an input string. Is this statement true or false?

A. True.

B. False.

Turing Machines: Formal Definition [2/4]

- A configuration of M is a triple $(q, y\sqcup^\omega, n)$ where $q \in Q$, $y \in \Gamma^*$, and $n \geq 0$.
- The configuration (q, z, n) describes M in state q with tape contents z and read/write head at position n.
- Configurations are denoted by $\alpha, \beta, \gamma, \ldots$.
- The start configuration on an input $x \in \Sigma^*$ is the configuration

$$(s, \vdash x\sqcup^\omega, 0).$$

Turing Machines: Formal Definition [3/4]

- The next configuration relation $\alpha \xrightarrow[M]{1} \beta$ is defined by:
  1. $(p, z, n) \xrightarrow[M]{1} (q, \mathrm{sub}_b^n(z), n-1)$ if $\delta(p, z_n) = (q, b, L)$.
  2. $(p, z, n) \xrightarrow[M]{1} (q, \mathrm{sub}_b^n(z), n+1)$ if $\delta(p, z_n) = (q, b, R)$.
- $\alpha \xrightarrow[M]{n} \beta$ and $\alpha \xrightarrow[M]{*} \beta$ are defined by;
  1. $\alpha \xrightarrow[M]{0} \alpha$.
  2. $\alpha \xrightarrow[M]{n+1} \beta$ if there is some γ such that $\alpha \xrightarrow[M]{n} \gamma \xrightarrow[M]{1} \beta$.
  3. $\alpha \xrightarrow[M]{*} \beta$ if there is some $n \geq 0$ such that $\alpha \xrightarrow[M]{n} \beta$.
- $\alpha \xrightarrow[M]{*} \beta$ is the reflexive, transitive closure of $\alpha \xrightarrow[M]{1} \beta$.

Turing Machines: Formal Definition [4/4]

- M accepts $x \in \Sigma^*$ if, for some y and n,
  $$(s, \vdash x \sqcup^\omega, 0) \xrightarrow[M]{*} (t, y \sqcup^\omega, n).$$
- M rejects $x \in \Sigma^*$ if, for some y and n,
  $$(s, \vdash x \sqcup^\omega, 0) \xrightarrow[M]{*} (r, y \sqcup^\omega, n).$$
- M halts on $x \in \Sigma^*$ if it accepts or rejects x.
- M loops on $x \in \Sigma^*$ if it neither accepts nor rejects x.
- M is total if it halts on all inputs.
- $L(M)$ is the set of strings accepted by M.
- A language is recursively enumerable (r.e.) if it is $L(M)$ for some TM M.
- A language is recursive if it is $L(M)$ for some total TM M.

Example 2

- Let $M = (Q, \Sigma, \Gamma, \vdash, \sqcup, \delta, s, t, r)$ be the TM where:

  $Q = \{s, q_1, \ldots, q_{10}, t, r\}$.
  $\Sigma = \{a, b, c\}$.
  $\Gamma = \Sigma \cup \{\vdash, \sqcup, \dashv\}$.
  δ is defined on p. 212 of D. Kozen, Automata and Computability.

- M accepts the non-context-free language $\{a^n b^n c^n \mid n \geq 0\}$.

- M does the following:
  1. Checks to see if the input has the form $a^* b^* c^*$.
  2. Overwrites the first $\sqcup$ with a $\dashv$.
  3. Scans left and right between $\vdash$ and $\dashv$ erasing one a, one b, and one c on each pass.
  4. Continues until there are no occurrences of a, b, or c
  5. Accepts or rejects appropriately.

Recursive and Recursively Enumerable Sets

- Let $A \subseteq \Sigma^*$.
- Proposition 1. If A is recursive, then $\sim A$ is also recursive.
- Proposition 2. If A and $\sim A$ are r.e., then A is recursive.
- The decision problem "$x \in A$?" is:
  - Decidable iff A is recursive.
  - Semidecidable iff A is r.e.
  - Undecidable iff A is nonrecursive.
- Theorem 1. If A is r.e., then there is a Turing machine that will enumerate the members of A.

Closure under Complement (iClicker)

Which of the following statements is true?

A. If A is a regular language, then so is $\sim A$.

B. If A is a context-free language, then so is $\sim A$.

C. If A is a recursive language, then so is $\sim A$.

D. If A is an r.e. language, then so is $\sim A$.

Different Kinds and Uses of Turing Machines

- There are many equivalent definitions of a Turing machine:
  1. With two-way tapes.
  2. With multiple tapes.
  3. With two-dimensional tapes.
  4. With multiple heads.
  5. Nondeterministic Turing machines.
- Turing machines can be used in different ways to:
  1. Decide a decision problem.
  2. Semidecide a decision problem.
  3. Compute a total function.
  4. Compute a partial function.
  5. Enumerate a set of values.

Modern Computers (iClicker)

What is the main characteristic of a modern computer?

A. Can store and manipulate massive amounts of data with great speed and accuracy.

B. Can access or control a large variety of peripheral devices.

C. Can store and run programs.

D. Implemented using electronic technology.

A Universal Turing Machine

- Fix an encoding scheme over $\{0, 1\}$ for Turing machines (over any alphabet) such that:
  1. Each TM M is represented by a string $s_M \in \{0, 1\}^*$.
  2. Each input string x to M is represented by a string $s_x \in \{0, 1\}^*$.
- It is then possible to construct a universal Turing machine U such that:
  1. U accepts $s_M\#s_x$ if M accepts x.
  2. U rejects $s_M\#s_x$ if M rejects x.
  3. U loops on $s_M\#s_x$ if M loops on x.
  4. U rejects $y\#z$ if it is not the case that y is a valid encoding of a TM M and z is a valid encoding of a string over M's alphabet.
- A universal Turing machine can thus simulate any Turing machine!

Diagonalization

- Theorem 2 (Georg Cantor, 1891). There does not exist a bijection $f : \mathbb{N} \to \mathcal{P}(\mathbb{N})$ (i.e., there is an uncountable set).

  Proof. The proof is by diagonalization. Assume f exists. Then f can be displayed as an infinite two-dimensional matrix:

$$
\begin{array}{c|ccccccc}
 & 0 & 1 & 2 & 3 & 4 & 5 & \dots \\
\hline
f(0) & 1 & 0 & 0 & 1 & 0 & 0 & \\
f(1) & 1 & 1 & 0 & 1 & 1 & 0 & \\
f(2) & 0 & 0 & 0 & 1 & 1 & 1 & \dots \\
f(3) & 0 & 1 & 1 & 1 & 1 & 1 & \\
f(4) & 1 & 0 & 0 & 1 & 0 & 0 & \\
f(5) & 0 & 1 & 1 & 1 & 0 & 1 & \\
\vdots & & & \vdots & & & & \ddots
\end{array}
$$

  Let D be the set represented by the sequence $b_0, b_1, b_2, \dots$ where

$$b_m = \begin{cases} 0 & \text{if } m \in f(m) \\ 1 & \text{if } m \notin f(m) \end{cases}$$

  Then $f(n) \neq D$ for all $n \in \mathbb{N}$ — which is a contradiction!

Halting Problem [1/3]

- The halting problem is the problem of deciding whether a given Turing machine will halt on a given input.
- Proposition 3. The halting problem is semidecidable.

  Proof. Any universal Turing machine semidecides the halting problem.
- Theorem 3 (Turing, 1937). The halting problem is undecidable.

  Proof. The proof is by diagonalization.

  Let $x \in \{0,1\}^*$. If x is the encoding of a Turing machine whose input alphabet is $\{0,1\}$, then let M_x be that machine; otherwise, let M_x be some trivial Turing machine that immediately halts on all strings in $\{0,1\}^*$.

Halting Problem [2/3]

Assume that K is a total Turing machine such that:

1. K accepts $x\#y$ if M_x halts on y.
2. K rejects $x\#y$ if M_x loops on y.

Now consider the following infinite two-dimensional matrix:

	ϵ	0	1	00	01	10	$\ldots$
M_ϵ	L	H	H	L	H	H	
M_0	L	L	H	L	L	H	
M_1	H	H	H	L	L	L	$\ldots$
M_{00}	H	L	L	L	L	L	
M_{01}	L	H	H	L	H	H	
M_{10}	H	L	L	L	H	L	
$\vdots$			$\vdots$				$\ddots$

H means M_x halts on y, while L means M_x loops on y.

Halting Problem [3/3]

Let D be the Turing machine that on an input $x \in \{0,1\}^*$, runs K on $x\#x$, accepting if K rejects and loops if K accepts.

Then $D = M_x$ for some $x \in \{0,1\}^*$. Therefore,

$$\begin{aligned} D \text{ halts on } x \quad &\text{iff} \quad K \text{ rejects } x\#x \\ &\text{iff} \quad M_x \text{ loops on } x \\ &\text{iff} \quad D \text{ loops on } x \end{aligned}$$

which is a contradiction!

Reduction

- Let $A \subseteq \Sigma^*$ and $B \subseteq \Pi^*$.
- A (many-one) reduction of A to B is a total computable function

  $$\sigma : \Sigma^* \to \Pi^*$$

  such that, for all $x \in \Sigma^*$,

  $$x \in A \text{ iff } \sigma(x) \in B.$$

- A is reducible to B, written $A \leq_m B$, if there is a reduction of A to B.
- Theorem 4.
  1. If $A \leq_m B$ and B is r.e., then A is r.e.
  2. If $A \leq_m B$ and B is recursive, then A is recursive.

Undecidable Problems

- There are many undecidable problems.
- The main techniques for showing that problems are undecidable are:
  1. Diagonalization.
  2. Reduction.
- If "$x \in A$?" is an undecidable problem, then we can show that "$x \in B$?" is an undecidable problem by reducing A to B.
  - The reduction establishes that, if B is recursive, then A must be recursive by Thm 4.2, which is a contradiction.