COMPUTATION: DAY 4

BURTON ROSENBERG
UNIVERSITY OF MIAMI

Contents

1. Review of the third day

We have discussed regular and context free languages, and two ways to describe each. The regular expressions and context free grammars describe the languages through a process of *generation*, and the finite automata and push-down automata describe the languages through *recognition*. We noted an important difference between the two classes, that oracles providing hints are only a convenience for regular languages, but for context free languages, only a more superior computational model can recognize what the grammar can and cannot generate.

2. Turing Machines

Friday, 3 March 2023

The Turing Machine was introduced by Alan Turing in a paper concerning the Halting Problem. In it, he proved that the problem of whether a finitely described mathematical procedure can always be made to come to a decision. For instance, given a notion of true and false statements, whether a proof system can be made to always prove the true statements and disprove the false statements.

And the answer is no. Proof is not entirely useless because the true statements can be proven. However, the false statements are more accurately described as those statements which cannot be shown to be true. Some false statements can be given finite demonstrations for being false, but in general, this is not possible.

3. Turing Machine Formal Definition

Monday, 6 March 2023

A Turing Machine is a finite automata endowed with a semi-infinite tape, consisting of a sequence of cells, each cell containing one from a finite set of tape symbols. The automata and the tape interact with a head, positioned over a cell, that can read, write and move one step left or one step right, according to the programming of the finite automata. Formally, a Turing Machine is,

$$\langle\, Q, \Sigma, \Gamma, \gamma, q_s, q_a, q_r \,\rangle$$

where,

- Q is a finite set of states,
- Σ is a finite set of symbols comprising the alphabet of the language,
- Γ is a finite set of symbols that can appear on the tape,
  - The tape alphabet includes the language alphabet, $\Sigma \subset \Gamma$,
  - There is a special blank symbol $\sqcup \in \Gamma$ but $\sqcup \not\in \Sigma$, that fills the unused portion of the tape.
- γ is the transition function, $\gamma : Q \times \Gamma \rightarrow Q \times \Gamma \times \{L, R\}$,
- $q_s, q_a, q_r \in Q$ are the start, accept and reject states, and $q_a \neq q_r$.

A *computation* by a Turing Machine M is as follows.

- A string $s \in \Sigma^*$ is written on the tape, with no blanks to the left of s.
- The Turing Machine M runs starting from state q_s and the tape head on the leftmost cell.
- Machine M can halt by entering state q_a, in which case the machine *accepts* (denoted T)
- or by entering state q_r, in which case the machine *rejects* (denoted F),
- or by continuing to run without ever stoping, in which case the machine does not halt, denoted $\perp$.

$$M(s) = \begin{cases} T & \text{accept} \\ F & \text{reject} \\ \perp & \text{does not halt} \end{cases}$$

Definition 3.1. A language $\mathcal{L} \subseteq \Sigma^*$ is *recursively enumerable* if there exists a Turing Machine M such that,

$$\mathcal{L} = \{\, s \in \Sigma^* \,|\, M(s) \,\}$$

The language is furthermore *recursive* if $\{\, s \in \Sigma^* \,|\, M(s) = \perp \,\}$ is empty. Recursively enumerable is also called *recognizable* and recursive is also called *decidable*.

Wednesday, 8 March 2023

Some example programs. The language,

$$E = \{\, w\#w \,|\, w \in \Sigma^* \,\}$$

shows the technique of marching back and forth over the tape, matching symbols, and leaving marks to find the location of the symbols to match. This show how an possibly unbounded task, checking *all* of the symbols in w can be made finite, except for updates to the tape contents. The language is decided, that is, the algorithm always halts.

The multi-tape Turing machine was introduced, and it was shown how this is not only a convenient model, but can reduce the time taken to decide the language E from $O(n^2)$ to $O(n)$, where n is the length of the input. However, it is of equal power. The technique of the *simulation argument* was demonstrated, where the equivalent power of two modes of Turing machines is shown by establishing a clear "cross-compiler" that could derive one model from another, while deciding or recognizing the same language.

N.B.: This poses an important boundary. If a one-tape turning machine recognizes a language in $o(n \log n)$, then the language is regular. The language E is not regular, hence the $O(n^2)$ is necessary. Regular languages can be recognized in time $O(n)$, so it is significant that a two-tape machine matches this bound, for a non-regular language.

N.B. Not presented in class, but the book presents an algorithm to recognize the set

$$\{\, 0^{2^k} \,|\, k \in \mathbb{N} \,\}$$

in time $O(n \log n)$. This show that the $o(n \log n)$ bound is exact.

4. Nondeterministic Turing Machines

Friday, 10 March 2023

The nondeterministic Turing machine was introduced and shown to be of equal power to a deterministic Turing machine. The non-deterministic counterpart of the Turing machine differs in that,

(1) The transition map δ maps $Q\times\Gamma$ is a set of actions, a subset of $Q\times\Gamma\times\{\,L, R\,\}$. It is possible that $\delta(q,\sigma)$ is the empty set, in which case the machine halts.
(2) There are no reject states — there are accept states and non-accepting halting states. Accepting a string means there is an computation that accepts. Rejecting a string means that all computation paths do not halt or halt without accepting.

It is sometimes useful to summarize the non-determinism of a non-deterministic Turing machine in a second tape. This tape is a Merlin, who answers for the computing Arthur, when $\delta(q,\sigma)$ is not empty or a singleton, which of the possible transitions to use. The language of machine M is defined as,

$$\mathcal{L}(M) = \{\, s \in \Sigma^* \,|\, \exists \pi \in \Pi^*, M(s,\pi) = T \,\}$$

where π can be thought of as the "proof" that s is in the language.

For a non-deterministic machine to be a deciding machine, the machine must always halt on all of Merlin's advice. It takes a little care to get this right, as the advice must be finite. So the restriction is,

$$\forall s \notin \mathcal{L}(M),\ \exists k \in \mathbb{N},\ \forall \pi \in \Pi^k,\ M(s,\pi) = F.$$

The *simulation argument* was presented, which had a non-Merlin enumerate all of Π^* in some exhaustive fashion (e.g. lexicographically, all shortest strings first) looking for an accept. This is a sufficient argument for recognizers. For deciders, the simulation must also note when, for the k large enough, all computations halt.

The complete the notion of Merlin, in the case that the string is in the language, Merlin is both wise and honest, and provides a proof π. If the string is not in the language, Merlin can only provide a special symbol $\perp$, to denote that it proof is non-terminating, consisting of all π and the quality that none will provide a proof.

5. Decidability of Regular Languages

It is decidable for a fixed Regular Language, whether a string is in that language. It is sufficient to write the DFA into a Turing machine, which will then always halt, accepting exactly those strings in the language.

5.1. **Acceptance is decidable for Regular Languages.** The *acceptance problem for Regular Languages* is a broader question. It treats the language as part of the input, and we consider the language of all pairs,

$$A_{FA} = \{\, (i,j) \in \mathbb{N} \times \mathbb{N} \,|\, M_i(s_j) \,\}$$

where M_i is the i-th DFA is an enumeration of DFA's, and s_j is the j-th word in the enumeration of the words in M_i's alphabet.

The approach in the book is different, and looks at the set of pairs $(\langle M \rangle, s)$ where $\langle M \rangle$ is a description of an DFA, in some universal description language, and s is an input, in some universal input language. However, I find it good to keep in mind that both DFA's and inputs can be enumerated, and the representation of both is left to the construction of the enumerator. In which case the the set A_{FA} is a more familiar object, a subset of $\mathbb{N} \times \mathbb{N}$.

Returning to the book's description, the language is decidable because we can create a Turing machine that takes the plan for a DFA and an input string as input, and using a universal DFA simulator, calculate out what the DFA would do on the given input string. In this sense A_{FA} is decideable.

5.2. **Emptiness is decidable for Regular Languages.** Many other sets concerning DFA' are decidable. The set,

$$E_{FA} = \{\, i \in \mathbb{N} \,|\, \forall j \in \mathbb{N} \text{ s.t. } M_i(s_j) \text{ rejects } \,\}$$

is the *emptiness for Regular languages* problem. In this case, the we cannot argue that we take a description of M_i and run it on every s_j, since this is not decider, as it does not terminate.

Instead, the machine description is examined to see if there as any accepting path. This can be done with a graph search algorithm. Hence E_{FA} is a decidable set.

5.3. **Equality is decidable for Regular Languages.** Finally the language of Regular Language equality is decidable,

$$EQ_{FA} = \{\, (i,j) \in \mathbb{N} \times \mathbb{N} \,|\, \mathcal{L}(M_i) = \mathcal{L}(M_j) \,\}$$

This cannot be done by checking the machines at all inputs, since this is not a finite procedure. However we have techniques for creating complement and intersection machines and then assert,

$$\mathcal{L}(M_i) = \mathcal{L}(M_j) \Leftrightarrow \mathcal{L}(M_i \oplus M_j) = \emptyset$$

where $M_i \oplus M_j$ is a machine, constructed from M_i and M_j, that accepts those strings accepted by exactly one of M_i or M_j. This machine can be described, and that description is put into the algorithm that determines emptiness of a language, from a machine description.

5.4. **Encoding and enumeration.** The languages above can be posed universally in terms of recognizing a subset of $\mathbb{N}$ or $\mathbb{N} \times \mathbb{N}$. This subset itself isn't natural, but depends on the details of the encoding and enumeration of the thought object — in this case, of DFA's or input strings. What is being said is that any such subset is decidable, as it represents regular language and their membership functions.

As an example of an encoding and enumeration. Once a language for DFA's has been fixed, for instance, our Python dictionary objects, strings over this language can be enumerated lexicographically, and one by one the string checked for syntactic correctness. With each found syntactically correct string, it is assigned the next index, so to get an a map (albeit rather slow) of counting integers to DFA's, $i \mapsto M_i$, and therefore a surjection of counting integers to regular languages.

6. Decidability of Context Free Languages

As for context free, all of the above are decidable except that last. It is undecidable whether two context free grammars, whether they describe the same language.

It is in fact recognizable if the grammars do not give the same language, as Merlin can provide us with a string in one language but not the other. However, the other side of the question is not so easily determined, and it is in fact beyond the ability of a Turing machine, or any computational device, to come to that determination.

7. Enumeration of Turing Machines

Friday, 24 March 2023

We need to speak of "all" Turing Machines, and in fact go so far as to make a list of them. That will have the entirety of Turing Machines in a list,

$$M_1, M_2, M_3, \ldots$$

that is, an enumeration function $\psi : \mathbb{N} \to \{\,\text{Turing Machines}\,\}$. There is considerable variability in this definition, in that the Turing Machine can be presented in various ways. It is sufficient that there is a programming language that in a sequence of symbols, say over Σ^*, if properly arranged (the symbol string is a compilable program) provide a Turning machine, and all Turing Machines are among the strings in this language. Besides that the description become a string over some fixed alphabet, that what is an isn't a proper machine description is decidable. In fact, the syntax of a viable Turing machine can be as simple as a regular language.

That said then, the enumeration proceeds as follows,

(1) A generator writes down one by one all elements on Σ^*, say in lexicographic order.

(2) The generated string is checked for syntax, by some simple Turing machine.
(3) If the string is a proper description it is given its number, n, and written onto the output tape.

hence the output tape has the sequence we desire. The inverse of this, for the map ψ is,

(1) Given n, use the described machine to check for proper TM descriptions,
(2) Do not write the correct descriptions to the output tape until the n-th description is reacher.
(3) Then write this description to the tape and halt.

We also need that the inputs to the machine be enumerated. There is a slight complication that the TM can announce the input alphabet, but once this alphabet is known, it is a simple and usual technique to enumerate it. Hence a run of M_i on the j input would involve a complicated denotation $M_i(\psi'_i(j))$, where ψ'_i is the input enumerator for machine i.

In what follows we will employ an abuse of notation and simple think of $(i,j) \in \mathbb{N} \times \mathbb{N}$ as the run of the i-th Turing machine on the j-th input, written $M_i(j)$.

8. Undecidability of A_{TM}

Theorem 8.1. The set,

$$A_{TM} = \{\, (i,j) \in \mathbb{N} \times \mathbb{N} \,|\, M_i(j) \,\}$$

is undecidable.

That is there is no Turing machine decider for A_{TM}. As a reminder, the assertion $M_i(j)$ means that M_i accepts j, and therefore halts. The set does not contain those inputs that are rejected, or input on which the machine does not halt.

Hence there is a function $\mathcal{H} : \mathbb{N} \times \mathbb{N} \to \{\, T, F \,\}$ that we can suppose is decidable, and look for the contradiction. From the machine H that decides $\mathcal{H}$ we create a new machine D defined by,

$$D(i) = \neg H(i,i)$$

The machine D runs H on the input (i,i) and if H accepts, D rejects; but if H rejects, D accepts. So D is also a decider. And a Turing machine in our list, so there is a j such that,

$$D(i) = M_j(i)$$

Then we have,

$$D(j) = M_j(j) = H(j,j) = \neg D(j)$$

a contradiction. Hence the set $\mathbb{H}$ is not decidable.

The actual set A_{TM} as a subset of $\mathbb{N} \times \mathbb{N}$ depends on ϕ, the encoding and enumerating function. That function addresses the questions of representation. However, this argument applies for any such ϕ, hence any such set so derived. It is helpful to know that undecidable sets are very numerous, and all of the results from various ways of defining ϕ are undecidable sets.

9. The uncountability of the reals

In class, Cantor's diagonalization argument for the uncountable of the reals was presented. There was a discussion of how it is relevant to Turing undecidability proof.

First that the techniques are similar. Second, that a broader argument can be made when the size of infinities is recognized. Turing machines are a countable infinity. However functions are an uncountable infinity. So there are not enough Turing machines to go around, if the goal is to compute all functions.

Computations are like the rationals, while computations are like the reals.