Computable Languages

- recursively enumerable languages are languages that can be defined by computation
- so far, every computational method that has ever been developed for specifying languages produces only recursively enumerable languages
- yet, most languages are not recursively enumerable
  - there are uncountably many languages over a particular alphabet
  - there are only countably many recursively enumerable languages over the same alphabet

  - *recursively enumerable* is a synonym for *Turing-acceptable*
  - *recursive* is a synonym for *Turing-decidable*

Uncomputable Languages

- most languages are not recursively enumerable
- what do these languages look like?
  - whatever property defines whether *w* is in *L* can't be computable
    - there is no Turing machine (or computer program) that tests whether *w* has the property

Symbolic Representation of Turing Machines

- consider a Turing machine *M*
- we can assume, without loss of generality –
  - *q* is the start state, *h* is the halt state, and the other states are named *q'*, *q''*, *q'''*, …
  - the symbols are 0, 1, *a,* # (blank) with auxiliary symbols *a'*, *a''*, *a'''*, …
- call such a Turing machine a *standard Turing machine*
- *M* can be represented with a string of symbols from the alphabet { *h*, *q*, *L*, *R*, #, 0, 1, *a*, ', $ }
  - the transition rule δ(*q*'',0) = (*a*''',*L*,*q*) is encoded as `q''0a'''Lq`
  - encode a complete machine by listing the transition rules, separated by $

without loss of generality (*w.l.o.g.)* – this assumption does not limit what we can consider because states and symbols can be renamed without changing the machine's function

A Turing Machine Generator

δ(*q*'',0) = (*a*''',*L*,*q*)
encoded as `q''0a'''Lq`

- not every string involving the alphabet { *h*, *q*, *L*, *R*, #, 0, 1, *a*, ', $ } is an encoded standard Turing machine
  - but whether or not *w* is an encoded Turing machine can be checked
- a list of all strings encoding standard Turing machines can be generated –
  - generate all strings over { *h*, *q*, *L*, *R*, #, 0, 1, *a*, ', $ }
  - for each string *w*, check if it encodes a Turing machine
  - if so, add *w* to the output list
- the symbolic representation of standard Turing machines is a recursively enumerable set
  - let T_i be the machine encoded by the *i*th string produced
  - given $n \in \mathbb{N}$, the symbolic representation for T_n can be found by repeating the generate-and-check process until *n*+1 TMs are found
- thus we can build a Turing machine *G* which, when run with input a^n, halts with the encoding of T_n

Universal Turing Machine

- the universal Turing machine *U* simulates the computation of any standard Turing machine *T* on any input *w*
  - (the symbolic representation of) *T* and *w* are written on *U*'s tape
  - *U* keeps track of *T*'s state and position
    - *T*'s state is written after *w* on the tape
    - use a special symbol @ to the left of the current symbol in *w* to denote the current position
  - *U*'s operation
    - write @ at the beginning of *w* and *q* after *w*
    - for each step in the computation of *T*
      - determine the current state and symbol for *T*
      - locate a transition rule that applies in this case
      - update the representation of *T*'s state, position, and tape to reflect applying the transition
      - if the new state is *h*, halt
  - observation: *U* halts if and only if *T* halts on input *w*

Theorem 5.4. *Let $T_0, T_1, T_2, \ldots,$ be the standard Turing machines, as described above. Let K be the language over the alphabet $\{a\}$ defined by*

$$K = \{a^n \mid T_n \text{ halts when run with input } a^n\}.$$

Then K is a recursively enumerable language, but K is not recursive. The complement

$$\overline{K} = \{a^n \mid T_n \text{ does not halt when run with input } a^n\}.$$

is a language that is not recursively enumerable.

- *recursively enumerable* is a synonym for *Turing-acceptable*
- *recursive* is a synonym for *Turing-decidable*

$$K = \{a^n \mid T_n \text{ halts when run with input } a^n\}.$$

- *K* is recursively enumerable because there's a Turing machine *M* which accepts it –
  - let *M* be a Turing machine which, given input a^n –
    - copies the input and runs *G* on the first copy of a^n, producing a symbolic description of the Turing machine T_n
    - runs *U* to simulate the computation of T_n on (the second copy of) input a^n
  - this simulation ends if and only if T_n halts when run with input a^n i.e. $a^n \in K$

$$K = \{a^n \mid T_n \text{ halts when run with input } a^n\}.$$

- *K* is not recursive because there's not a Turing machine *M* which decides it –
  - let *H* be any Turing machine
  - let *M* be a Turing machine which does the same thing as *H* until *H* halts (if *H* halts)
    - if *H* halts with output 1, *M* goes into an infinite loop
    - otherwise (any other output) *M* halts
  - assume, without loss of generality, that *M* is a standard Turing machine i.e. $M = T_n$ for some $n \in \mathbb{N}$
  - run $M = T_n$ on input a^n –
    - if *H* halts with output 1 on input a^n, T_n doesn't halt on input a^n
    - if *H* halts with output 0 on input a^n, T_n halts on input a^n
    - (what happens with other output or if *H* doesn't halt doesn't matter)
  - we need to show that *H* doesn't decide *K*
    - if *H* decides *K*, running *H* on a^n means that it should halt with output 1 if $a^n \in K$, that is, T_n halts when run with input a^n, and output 0 if $a^n \notin K$, that is, T_n doesn't halt when run with input a^n
    - but that's not what happens – *H* doesn't give the right answer, and so a Turing machine *H* that decides *K* doesn't exist

Theorem 5.4. *Let $T_0, T_1, T_2, \ldots$, be the standard Turing machines, as described above. Let K be the language over the alphabet $\{a\}$ defined by*

$$K = \{a^n \mid T_n \text{ halts when run with input } a^n\}.$$

Then K is a recursively enumerable language, but K is not recursive. The complement

$$\overline{K} = \{a^n \mid T_n \text{ does not halt when run with input } a^n\}.$$

is a language that is not recursively enumerable.

- $\overline{K}$ is not recursively enumerable because if both K and $\overline{K}$ were, K would be recursive

Theorem 5.3. *Let Σ be an alphabet and let L be a language over Σ. Then L is recursive if and only if both L and its complement, $\Sigma^* \setminus L$, are recursively enumerable.*

The Halting Problem

$$K = \{a^n \mid T_n \text{ halts when run with input } a^n\}.$$

- deciding *K* is known as the *halting problem*
- no Turing machine – and thus no computer program – can solve this problem
  - it is *computationally unsolvable*
  - note that this doesn't mean that no instances of the problem can be solved, just that no Turing machine (or program) can produce the correct answer in all cases
- the halting problem is not the only computationally unsolvable problem
  - e.g. does a particular Turing machine halt for all possible inputs?
  - e.g. does a program halt with a particular input?
  - e.g. are two Turing machines (or programs) equivalent, that is, do they produce the same output for each possible input?
  - e.g. will a particular Turing machine halt if started with a blank tape?