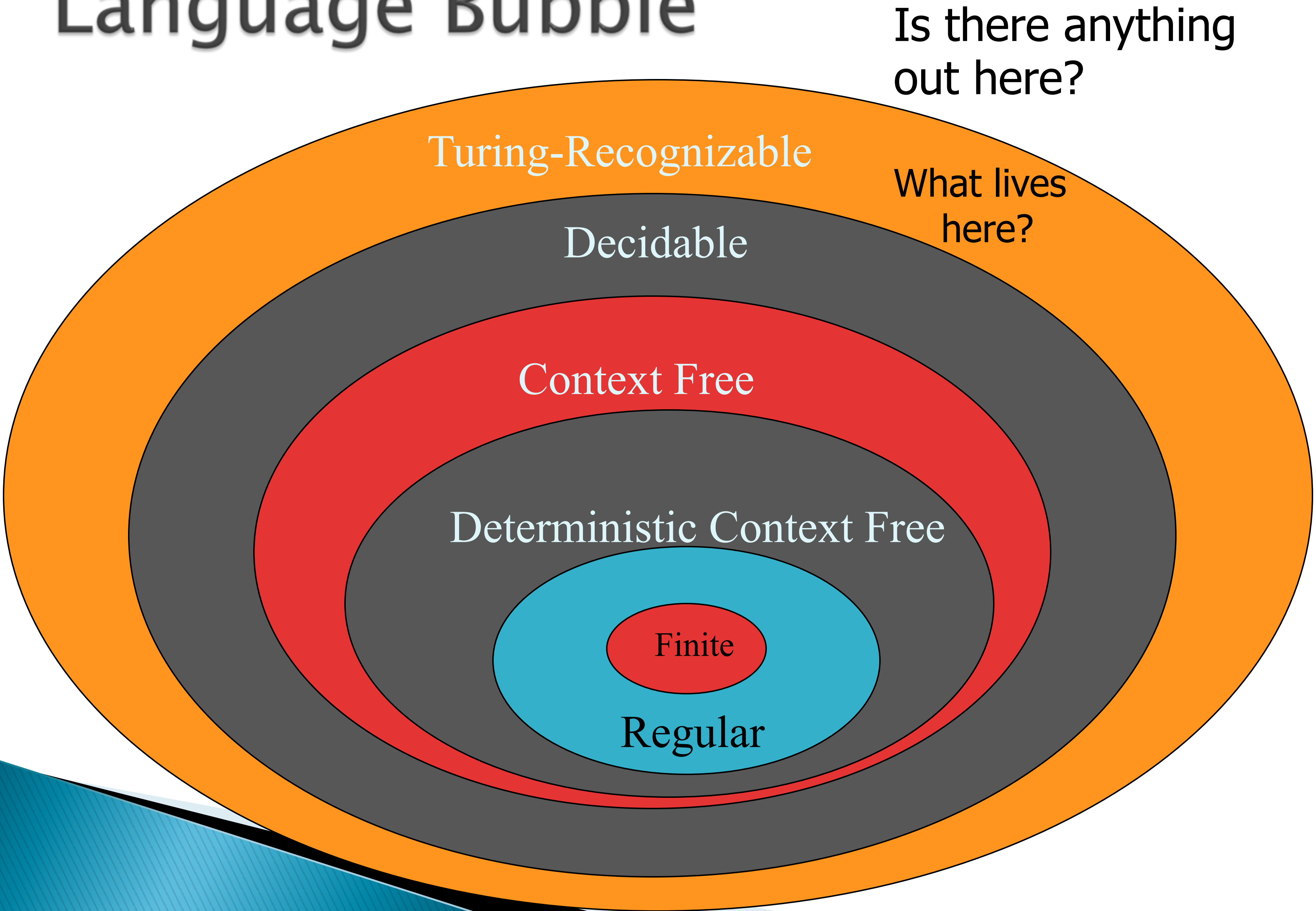
Turing Machines

and their languages

Summary

- Some languages are Turing-decidable
  - A Turing Machine will halt on all inputs (either accepting or rejecting). No infinite loops.
- Some languages are Turing-recognizable, but not decidable.
  - A Turing Machine recognizes the language, but it will loop infinitely on some inputs
- Some languages are not Turing-recognizable
  - There is no Turing Machine that can recognize the language

Language Bubble



Turing Machines and Languages

- Game plan
  1. Show that there exists a language that is Turing-recognizable but not decidable.

  2. Show that there exists a language that is not Turing-recognizable.

Note: it's not enough to just design a TM that loops on some input. As we've seen, this may just be a poor approach, and a better (TM decider) approach may exist. Instead, we have to prove there is no way to build such a decider.

Not Just a Theoretical Exercise

- Variety of Unsolvable problems
  - Hilbert's 10^{th} problem: determine if a polynomial has an integral root
  - Software Verification: determine if software is performing as it is intended to
  - Ambiguity: determine if an arbitrary context-free grammar is ambiguous
  - Acceptance test for Turing Machines: determine if an arbitrary Turing Machine accepts an arbitrary string
  - Halting problem for Turing Machines: determine if an arbitrary Turing Machine halts on a given input string

Algorithms and Turing Machines

- Running algorithms on objects:
  - TMs take strings as input.
  - For running "algorithms" on other objects,
    - Must encode the object as a string.
    - Any decent encoding will do.
    - When running TMs on objects, it is assumed that decoding gets performed by the TM and that input is valid.

  - Notation:
    - If O is an object to be input to a TM, <O> is the encoded object.
    - If $O_1, O_2, \ldots, O_n$ are multiple objects to be used as input to a TM, $<O_1, O_2, \ldots, O_n>$ is the encoded list of objects.

Examples

- Examples of decidable languages in which the accepted strings are encodings of objects
  - A = {<G>| G is a connected undirected graph }
  - A_{DFA} = {<B,w> | B is a DFA that accepts input string w}
  - EQ_{DFA} = {<A,B> | A and B are DFAs and L(A) = L(B)}
  - A_{CFG} = {<G,w> | G is a CFG that generates string w}
  - E_{CFG} = {<G> | G is a CFG and L(G) = Ø}

What About Encoding a Turing Machine?

- Consider the following
  - $A_{TM} = \{ <M, w> \mid M \text{ is a TM and } M \text{ accepts } w\}$
    - This is the acceptance problem for Turing Machines
      - A_{TM} is the language corresponding to whether a string is accepted by a given Turing Machine
    - Can we build a TM that takes a TM as input…
    - Yes, but this is where we will lose decidability

Acceptance for Turing Machines

- Note that A_{TM} is recognizable.
- Define TM U as follows:
  - U = "on input <M, w>, where M is a TM, and w is a string:
    - Simulate M on input w
    - If M accepts, U accepts. If M rejects, U rejects."
- U recognizes A_{TM}
- U is sometimes called the Universal Turing Machine. (it can simulate any other TM)

Proof That A_{TM} is Undecidable

- Proof by contradiction
  - Assume A_{TM} = {<M, w> | M is a TM and M accepts w} is decidable
  - Let H be a decider for A_{TM}
    - H is a TM that halts on all inputs

  - Next we construct a TM, D, that takes as input the encoding of a TM, M.
    - It uses H as a subroutine to determine what M does with a string w that is actually the encoding of the machine M itself. That is, it simulates H on input <M, <M>>.
    - Whatever H does – D does the opposite

Proof That A_{TM} is Undecidable

- Proof by contradiction
  - D = “On input <M> where M is a TM:
    - Run H on input <M, <M>>
    - Output the opposite of what H outputs.

  - D accepts <M> if H rejects <M, <M>>
    - D accepts <M> if M does not accept <M>
  - D rejects <M> if H accepts <M, <M>>
    - D rejects <M> if M accepts <M>

Proof That A_{TM} is Undecidable

- Proof by contradiction
  - D accepts <M> if M does not accept <M>
  - D rejects <M> if M accepts <M>
- What happens if D is run with its own description?
  - D accepts <D> if D does not accept <D>
  - D rejects <D> if D accepts <D>
    - CONTRADICTION
- D and H can not exist: A_{TM} is not decidable

Unrecognizable Languages

- Turing-unrecognizable languages exist
  - Consider any arbitrary Turing Machine
    - $M = (Q, \Sigma, \Gamma, \delta, q_0, q_{accept}, q_{reject})$
  - We can define a small alphabet to encode M
    - E.g. $\Sigma = \{0, 1, p\}$
      - 0,1 are used to indicate numerical values
        - How many states there are
        - Unicode/ASCII encoding of alphabet symbols
        - Details of transition function
      - p is a punctuation symbol that is used to demarcate different pieces of the string
- Any TM can be encoded using just these symbols!

Unrecognizable Languages

- Turing-unrecognizable languages exist
  - Consider finite alphabet Σ
  - How many strings over Σ ?
    - Countably infinite – we can list all strings of length 0, 1, 2, …
  - How many TM can be encoded using Σ ?
    - Countably infinite (some subset of all strings over Σ)
    - Thus there at most are countably infinite different TM, and each recognizes only 1 language

Unrecognizable Languages

- Turing-unrecognizable languages exist
  - How many languages are there over Σ ?
    - Power set of all strings over Σ
    - All subsets of an infinite collection
    - Uncountably infinite
      - SOME (MOST) OF THESE LANGUAGES HAVE NO TURING MACHINE TO RECOGNIZE THEM

A Turing-Unrecognizable Language

- The complement of A_{TM} is not Turing-recognizable
- This will follow from the following theorem:
  - A language is decidable if and only if it is Turing-recognizable and co-Turing-recognizable
    - (A language is co-Turing-recognizable if its complement is Turing-recognizable)

Proof

- A language is decidable if and only if it is Turing-recognizable and co-Turing-recognizable
  - Assume language A is decidable.
    - Decidable languages are closed under complement.
    - (Since TM halts on all input, we can have a different TM that flip-flops reject/accept. It also halts on all input, and accepts the complement of A.)
    - So both A and A' are decidable
    - Decidable languages are also recognizable.

Proof

- A language is decidable if and only if it Turing-recognizable and co-Turing-recognizable
  - Assume A and A' are both Turing-recognizable.
  - Let M_1 be a TM for A
  - Let M_2 be a TM for A'
    - M_1 and M_2 may loop on some inputs, but they halt on all accepted strings.
    - Have TM M run both M_1 and M_2 in parallel (take turns running each machine for an increasing number of steps)
      - if M_1 accepts, then M accepts
      - If M_2 accepts, then M rejects
      - Since M_1 and M_2 both halt on accepting inputs, M will halt on all inputs

Corollary

- The complement of A_{TM} is not Turing-recognizable
  - We know A_{TM} is Turing-recognizable but not decidable.
  - If the complement of A_{TM} were Turing-recognizable, then by the previous theorem, they would both also be decidable.

Closure Properties

- The class of decidable languages is closed under:
  - Union
  - Concatenation
  - Kleene Star
  - Complementation
  - Intersection
  - Difference

  - See Problem 3.15 for details

Closure Properties

- The class of Turing-recognizable languages is closed under:
  - Union
  - Concatenation
  - Kleene Star
  - Intersection

- But not closed under complementation (we just showed that using A_{TM})
  - And therefore not closed under difference either