Lecture 25: Language Hierarchies: Decidable & Semi-Decidable

CSCI 101
Spring, 2019

Kim Bruce

Halting Problem in Java

- Can we determine whether given a Java program P, will P ever halt?
- Suppose there is a solution:
  - a class with method halts that takes as inputs a string with the name of the file, and then returns true or false depending on whether or not the file contains a program that halts on the empty string.

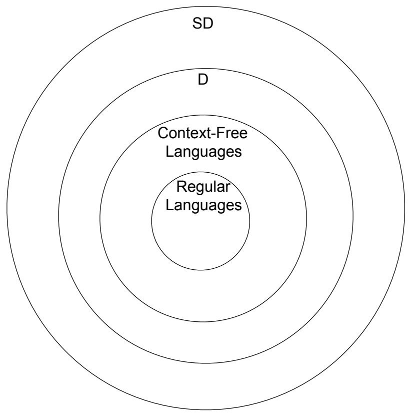
/ Charlatan class contains method halts, which takes a filename as input returning true iff the program in the file is legal and halts on empty input. */*

```
public class Debunker {

   public void what(String fileName) {
      Charlatan charlatan = new Charlatan();
      if (charlatan.halts(fileName)) {
         while (true){}   // run forever!
      }
   }     // else halt

   public static void main(String[] args) {
      Debunker debunk = new Debunker();
      debunk.what("Debunker.java");}
   }
}
```

The Charlatan lies!

- If charlatan.halts("Debunker.java") returns true then method `what` enters while loop
  - runs forever -> doesn't halt!
- If charlatan.halts("Debunker.java") is false then procedure completes
  - halts!
- What do we know about charlatan.halts?
  - Can't work as promised!

Decidable & Semidecidable



The Hierarchy

- Theorem: The set of context-free languages is a proper subset of D.
  - Proof: Every context-free language is decidable, so the context-free languages are a subset of D.
  - There is at least one language, $A^nB^nC^n$, that is decidable but not context-free.

Distinguishing D and SD

- Most obvious languages in SD also in D
  - $A^nB^nC^n = \{a^nb^nc^n \mid n \geq 0\}$
  - $\{wcw \mid w \in \{a, b\}^*\}$
  - $\{ww \mid w \in \{a, b\}^*\}$
  - {w of form x∗y=z: $x,y,z \in \{0, 1\}^*$ and, when x, y, and z are viewed as binary numbers, $x*y = z$}
- But already found some in gap, e.g. H_{TM}

Outside of SD

- Uncountably many languages outside of SD
- *Example:* Complement of H_{TM}

Closure Properties

- Theorem: D is closed under complement
  - Proof: Let L ε D. Build TM deciding L
  - ...
  - Proof depends on TM deterministic and always halts.
- What about SD?
  - Not true for H_{TM}

Equivalences to SD

- A TM M *enumerates* the language L iff, for some fixed state p of M,
  $$L = \{w : (s, \varepsilon) \vdash_M^* (p, w)\}.$$
  - *Potentially infinite computation.*
- A language is *Turing-enumerable* iff there is a Turing machine that enumerates it.

SD & Turing Enumerable

- Theorem: A language is SD iff it is Turing enumerable.
  - Proof: Spose L is Turing enumerable. Show L is SD.
    - Let w be input. Start enumerating L. Every time enter state p, check to see if contents of tape is w. If yes then halt and stop. Otherwise keep going.
  - "Obvious" proof in other direction not work!

SD & Turing Enumerable

- Theorem: A language is in SD iff it is Turing enumerable.
  - Proof (*cont*): Spose L is in SD. Show L can be enumerated.
    - Enumerate all $w \in \Sigma^*$ lexicographically: ε, a, b, aa, ab, ba, bb, ...
    - As each w_i is enumerated, start a copy of M to check with w_i as input.
    - Execute one step of each M with w_i started, excluding those that have halted
    - Whenever an M accepts a w_i, output w_i.
- Called dove-tailing

Lexicographically Enumerable

- M *lexicographically enumerates* L iff M enumerates the elements of L in lexicographic order.
- A language L is *lexicographically Turing-enumerable* iff there is a Turing machine that lexicographically enumerates it.

Lexicographically Enumerable

- Theorem: A language is in D iff it is lexicographically enumerable.
  - Proof: Suppose L is in D. Show L can be enumerated lexicographically.
    - Enumerate all $w \in \Sigma^*$ lexicographically: ε, a, b, aa, ab, ba, bb, ...
    - As each w_i is enumerated, run M deciding L on w_i as input.
    - If M accepts a w_i, output w_i. Otherwise go to next.
    - Easier here because M always halts

Lexicographically Enumerable

- Theorem: A language is in D iff it is lexicographically enumerable.
  - Proof (*cont)*: Spose L can be enumerated lexicographically. Show L is in D.
  - To determine if w in L:
    - Start enumerating all elts of L lexicographically
    - If w is enumerated, then accept.
    - If go past w in lexicographic order, then reject
    - If halts before getting to w, then also reject.

Oops!

- Second part of proof has a hole.
  - Suppose M lexicographically enumerates L = {a,ba,aba} by enumerating three elements and then continuing forever without ever enumerating another element. If w comes after last element, then won't know if in or not!
  - Can only happen if L is finite.
  - But all finite languages are decidable.
  - Fixes proof
- But not decidable whether L is finite!!

Summary

- Church-Turing Thesis: all models of computation give same computable functions
- Halting problem is undecidable.
  - Can prove lots of others undecidable by using them to solve halting problem.
- Semi-decidable sets equivalent to Turing enumerable.
  - Decidable if Turing lexicographically-enumerable.

Application of Homework

- Can define the typed lambda calculus.
- Can prove that every program in typed lambda calculus is total.
- Therefore does not include all total computable functions.
- Type systems are either conservative or incorrect.

Rice's Theorem

- No non-trivial property of the SD languages is decidable
  or equivalently
- Any language that can be described as
  {<M> | P(L(M)) = true}
  for any non-trivial property P, is not in D
- A property is *trivial* if it is either true for all languages or false for all languages.

Applying Rice

- Must specify property P
- Show domain of P is SD languages, e.g., languages accepted by TM's.
- Show P is non-trivial
  - *true of at least one , false of at least one.*

Proof of Rice

- Proof: Let P be a non-trivial property of SD languages. Show can reduce Halting to $L = \{<M> \mid P(L(M)) = \text{true}\}$
- $\varnothing$ is an SD language. Suppose $P(\varnothing)$ = true
  - Similar proof if it is false (use not-P instead).
- Since P is non-trivial, $\exists L'$ in SD s.t. $P(L')$ = false. Let M' semi-decide L'

Proof of Rice *(cont.)*

- Build new TM from M' that will decide H_{TM}.
- Given $<M,w>$, build M_w' s.t., when started w/ x:
  - Copy x onto another work tape
  - Erase and write w on input tape
  - Run M on w
  - Copy x back on tape and run M' on x
- Recall M' semi-decides L' and $P(L')$ = false

Proof of Rice *(cont.)*

- Recall M' semi-decides L' and $P(L')$ = false
- If $<M,w> \in H_{TM}$ then M_w' accepts L'
- If $<M,w> \notin H_{TM}$ then M_w' accepts $\varnothing$.
- Thus $<M,w> \in H_{TM}$ iff $P(L(M_w'))$ = false
  iff not($<M_w'> \in L$)
- Therefore $H_{TM} \leq_M L$ and L is undecidable!

More Undecidable

- Is L(M) regular?
- Is L(M) context-free?
- Can we automatically check if your program is correct (e.g., matches the solution)?
- Does M ever halt with an error?

Decision Problems for Regular Languages

- For FSMs/Regular languages, most things decidable:
  - $A_{FSM} = \{\langle M,w \rangle \mid M \text{ is an FSM and } w \in L(M)\}$
  - $E_{FSM} = \{M \mid M \text{ is a FSM and } L(M)=\emptyset\}$
  - $TOTAL_{FSM} = \{M \mid M \text{ is a FSM and } L(M)=\Sigma^*\}$
  - $EQUAL_{FSM} = \{<M,N> \mid M,N \text{ are FSMs and } L(M)=L(N)\}$

Decision Problems for CFGs

- Not for PDAs/CFGs:
  - $A_{CFG} = \{\langle G,w \rangle \mid G \text{ is a CFG and } w \in L(G)\}$
  - $E_{CFG} = \{G \mid G \text{ is a CFG and } L(G)=\emptyset\}$
  - $Finite_{CFG} = \{G \mid G \text{ is a CFG and } L(G) \text{ is finite}\}$
  - $TOTAL_{CFG} = \{G \mid G \text{ is a CFG and } L(G)=\Sigma^*\}$
  - $EQUAL_{CFG} = \{< G, G' > \mid G, G' \text{ are CFGs and } L(G)=L(G')\}$
  - ....

Not Decidable!