CSE 396 Lecture Tue May 1 Spr 2017 (1)

Dist Speaker Barbara Ryder Thu. SU Theater 3:30pm

Scientista Wed 3pm Davis 113A (until 3:30)

Next Week Cynthia Rudin Wed 2-3pm SU 330 (UB '96!)

Consider Java programs that take ASCII input from System.in (or from a file), (choose and fix one) → and that accept by either {executing System.exit(0); printing "1" and halting. Then we can define:

$$D_{Java} = \{\overbrace{\langle P \rangle}^{\text{string}} : P \text{ is a } \overset{\text{it compiles}}{\text{legal}} \text{ Java program that does not accept } \langle P \rangle\}$$

? for "Quixotic"

Theorem: There does not exist a Java program Q s.t. $D_{Java} = L(Q)$.

Proof: Suppose Q exists. Then for one thing, the string $x = \langle Q \rangle$ compiles and yields Q.class.

Q accepts $\overset{x}{\langle Q \rangle} \iff \langle Q \rangle \in D_{Java} \overset{\text{by } L(Q)}{=} D_{Java}$

$\iff Q$ does not accept $\langle Q \rangle$ by defn of D_{Java}.

There is no way to escape this contradiction. ∴ Q does not exist. ☒

I.e. D_{Java} is not "Java-recognizable." By the interconvertibility of Java and Turing machines, D_{Java} is not Turing-recognizable either. Similarly, D_{TM} is not Java-recognizable either. Hence neither D_{TM} nor D_{Java} is {Turing acceptable; computably enumerable; recursively enumerable; in RE.

However, $\sim D_{Java} = \{x :$ x does not compile in Java, or x compiles to P such that P does accept x $\}$ is in RE.

Notable Fact: The language of strings that compile in Java is decidable: javac always halts. Not true for C++!

Modulo not compiling, the "moral" complement is

standard, useful text

$$\underline{\underline{K}}_{Java} = \{x : x \text{ compiles to } P \text{ such that } P \text{ accepts } x\}.$$

$$\underline{\underline{K}}_{TM} = \{x : x \text{ is a legal Turing Machine that accepts } x\}.$$

∴ K is r.e. but not decidable. D is not even r.e./c.e.

One other standard notation uses a fixed "Enumeration" $M_0, M_1, M_2, M_3, \ldots$ of Turing Machines, or a aka "Gödel Numbering" $P_0, P_1, P_2, P_3, \ldots$ of Java programs. Then: (subscript "TM or Java")

$D_{Java} = \{i : P_i$ does not accept $i\}$ — $i \equiv$ number or string.

$D_{TM} = \{e : M_e$ does not accept $e\}$ — $e \equiv$ number $\equiv$ string.

$K_{TM} = \{e : M_e$ does accept $e\}$. Now $K_{TM} = \sim D_{TM}$ literally

$A_{TM} = \{\langle M, x\rangle :$ The TM M accepts x$\}$

Theorem: A_{TM} is r.e. (c.e.) but not decidable.

§4.2

Proof: First, A_{TM} does equal L("Turing Kit") ← a Java program

But if it were decidable by a program R, then

$e \in K \iff R$ accepts $\langle e, e\rangle$ so that

K_{TM} would be decidable too. But we showed that false. ☒

Intuitively, K_{TM} is a "special case" restriction of the A_{TM} problem.

Since K_{TM} is hard, A_{TM} is only harder.

Def^n: Define A and B to be "para-complements" if

$A \cap B = \emptyset$ and $A \cup B$ is decidable.] i.e. $B = U \setminus A$ for some decidable U st. $A \subseteq U$ too.

Call it a regular para-complement if U is regular.

Examples:
- D_{Java} and K_{Java} are literally para-complements $U = \{x : x$ compiles in Java$\}$
- Literal complements are "para" with $U = \Sigma^*$, which is regular.
- $\{a^n b^n c^n : n \geq 0\}$ and $\{a^i b^j c^k : i \neq j$ or $j \neq k\}$ are regular para-complements with $U = a^* b^* c^*$ as on HW 7.

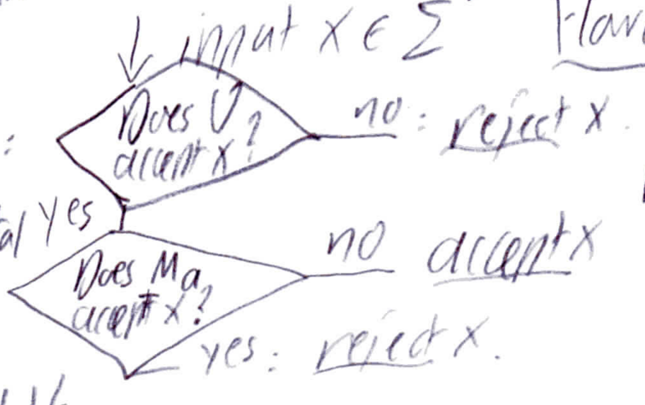
Theorem: If A and B are para-complements (for a decidable U)
Then (a) A is decidable $\iff$ B is decidable.
(b) if A and B are r.e. then both are decidable.

Proof: Take a total TM M_U st. $L(M_U) = U$ i.e. st M_U halts for all inputs.
(a) Suppose A is decidable. Then we have a total TM M_a st. $L(M_a) = A$.

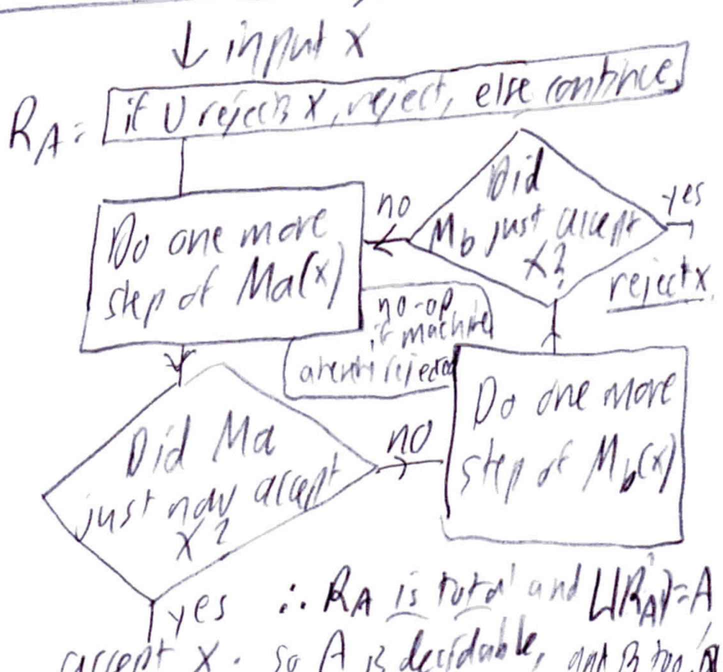
Flowchart convention: Total routines go in solid boxes or ◇s.

$\boxed{B = U \setminus A}$

Define M_b as:



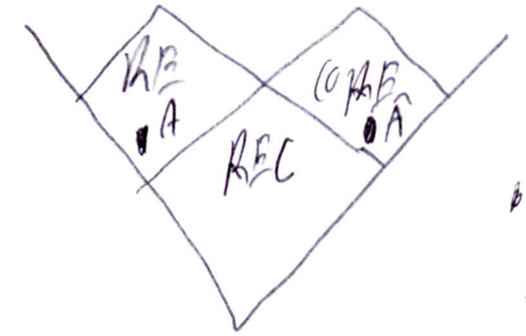
Then M_b is total and $L(M_b) = B$, so B is decidable.



$\therefore$ R_A is total and $L(R_A) = A$. So A is decidable, and B too. ∎

Part (b) We can take TMs M_a and M_b st. $L(M_a) = A$ and $L(M_b) = B$ but they need not be total. However, the act of stepping each one step ahead in a computation does terminate, so can go in solid box.

When $U = \Sigma^*$ this says: • A language A is decidable $\iff$ $\tilde{A}$ is decidable



REC is closed under complements.

• If A and $\tilde{A}$ are r.e., then both are decidable.

$\Rightarrow$ RE $\cap$ co-RE = REC (REC aka DEC)

The Halting Problem: — not in text but standard

INST: A TM M and an input x to M.

QUES: Does $M(x)\downarrow$? ($\downarrow \equiv$ halts, $\uparrow \equiv$ does not halt)

The language of this problem can be called HP_{TM}.

Thm: HP_{TM} is $\begin{cases} \text{r.e.} \\ \text{c.e.} \end{cases}$ but undecidable. fixed?

Proof: HP_{TM} would be L("Turing Kit") if Turing Kit accepted "when M halts with string not accepted" too.

If we had a decider R for HP_{TM}, then we would get one for A_{TM} by modifying a given M to M' that has a loop q_{rej} $(c/c, S)$ at its rejecting state

$M'(x)\downarrow \iff M(x)$ accepts, so R would decide A_{TM} too.

Added: What we have actually done is reduce A_{TM} to HP_{TM}. We can reduce the other way too: make the new TM accept if and when the original TM halts (by redirecting arcs to q_{rej} to go to q_{acc} again.) Reductions are the topic of Thursday's lecture.