L11- Decidability

The Formal Model of Computation

Sarah Sorahi

Position of This Lecture

- Finite Automata $\rightarrow$ Decidable problems
- Context-Free Grammars $\rightarrow$ Mostly decidable
- Turing Machines $\rightarrow$ General computation
- Decidability: limits of algorithmic solvability

Decision Problems as Languages

Key Idea

Every decision problem can be expressed as a language.

$$A_{DFA} = \{\langle B, w\rangle \mid B \text{ is a DFA and } B \text{ accepts } w\}$$

- Input: encoded object
- Output: YES / NO
- YES instances form a language

Decidable Languages

Definition

A language L is **decidable** if there exists a Turing machine M such that:

- M halts on *all* inputs
- M accepts exactly the strings in L

Such a machine is called a **decider**.

Recognizable vs Decidable

Definitions

- **Recognizable (RE)**: TM halts and accepts strings in L
- **Decidable**: TM halts on all inputs

Key Relationship

$$\text{Decidable} \subsetneq \text{Recognizable}$$

co-RE Languages

- $L \in co\text{-}RE$ if $\overline{L}$ is recognizable
- L is decidable iff:

$$L \in RE \text{ and } L \in co\text{-}RE$$

Interpretation

YES and NO answers can both be detected algorithmically.

Decidable Example: DFA Acceptance

$$A_{DFA} = \{\langle B, w\rangle \mid B \text{ accepts } w\}$$

Decider

1. Simulate B on input w
2. If final state is accepting $\rightarrow$ accept
3. Otherwise reject

- Simulation halts in $|w|$ steps
- Therefore A_{DFA} is decidable

Decidable Example: DFA Emptiness

$$E_{DFA} = \{\langle B \rangle \mid L(B) = \emptyset\}$$

Algorithm

- View DFA as directed graph
- Perform reachability from start state
- Check if any accepting state is reachable

$$\text{Reachable accepting state} \iff L(B) \neq \emptyset$$

CFG Membership is Decidable

$$A_{CFG} = \{\langle G, w \rangle \mid G \text{ generates } w\}$$

- Convert grammar to Chomsky Normal Form
- Use CYK algorithm
- Finite dynamic programming table

Key Point

Termination is guaranteed by finite input length.

CFG Emptiness is Decidable

$$E_{CFG} = \{\langle G \rangle \mid L(G) = \emptyset\}$$

- Mark productive variables
- Mark reachable variables
- Check if start symbol is both

The Shift to Turing Machines

- Infinite tape
- Unbounded runtime
- Machines can simulate machines

Consequence

Behavioral questions become extremely hard.

A_{TM} is Recognizable

$$A_{TM} = \{\langle M, w \rangle \mid M \text{ accepts } w\}$$

- Simulate M on w
- If M accepts $\rightarrow$ accept
- If M loops $\rightarrow$ simulation loops

$$A_{TM} \in RE$$

The Halting Problem

$$HALT_{TM} = \{\langle M, w\rangle \mid M \text{ halts on } w\}$$

- Recognizable: simulate and wait
- Central undecidable problem

Undecidability of the Halting Problem

Assumption

Assume a decider H for $HALT_{TM}$ exists.

Construction

Define TM D:

- On input $\langle M \rangle$, run $H(\langle M, M \rangle)$
- If H accepts, loop forever
- If H rejects, halt and accept

Diagonalization Contradiction

Consider $D(\langle D \rangle)$:

- If H says “halts” $\rightarrow$ D loops
- If H says “loops” $\rightarrow$ D halts

Conclusion

Contradiction $\Rightarrow$ $HALT_{TM}$ is undecidable.

Mapping Reductions

Definition

$A \leq_m B$ if a computable function f exists such that:

$$x \in A \iff f(x) \in B$$

- Used to prove undecidability
- "Solving B would solve A"

A_{TM} is Undecidable

- Reduce $HALT_{TM}$ to A_{TM}
- Encode halting as acceptance
- Contradiction if A_{TM} were decidable

TM Emptiness is Undecidable

$$E_{TM} = \{\langle M \rangle \mid L(M) = \emptyset\}$$

- Reduce from A_{TM}
- Construct machine that accepts iff M accepts w

TM Equivalence is Undecidable

$$EQ_{TM} = \{\langle M_1, M_2 \rangle \mid L(M_1) = L(M_2)\}$$

- Semantic property of languages
- Reduce from E_{TM}

Rice's Theorem

Statement

Any nontrivial semantic property of a TM-recognized language is undecidable.

- Nontrivial: true for some TMs, false for others
- Applies to language properties only

Applications of Rice's Theorem

Undecidable properties:

- $L(M)$ is empty
- $L(M)$ is finite
- $L(M)$ is regular
- M accepts all strings

- Decidable vs Recognizable vs co-RE
- Dovetailing and the "RE $\cap$ coRE $=$ Decidable" theorem
- Decidable examples: EQ_{DFA}, A_{CFG}, E_{CFG}, ALL_{DFA}
- Undecidable core: $HALT_{TM}$, A_{TM}
- Reductions: E_{TM}, EQ_{TM}
- Rice's Theorem $+$ applications ($FINITE_{TM}$, ALL_{TM}, REG_{TM})
- Classic extra: Post Correspondence Problem (PCP)
- One CFL vs TM contrast: CFG ambiguity (undecidable)

Decision Problems as Languages

Encoding Convention

Inputs are strings encoding machines/objects, e.g. M, w, G, w, B.

Example:

$$A_{DFA} = \{B, w \mid B \text{ is a DFA and } B \text{ accepts } w\}$$

- YES-instances form a language
- We study which such languages are decidable/recognizable

Decidable vs Recognizable

Decidable (Recursive)

L is decidable if some TM halts on all inputs and decides membership in L.

Recognizable (RE / Semi-decidable)

L is recognizable if some TM halts and accepts when input is in L (may loop otherwise).

$$\text{Decidable} \subsetneq \text{Recognizable}$$

co-RE and Complements

- $L \in coRE$ iff $\overline{L}$ is recognizable.
- Intuition: there is a procedure that halts on NO-instances (for L).

Key Theorem (Preview)

$$L \text{ decidable} \iff L \in RE \text{ and } L \in coRE.$$

Dovetailing Technique (Parallel Simulation)

Dovetailing

Given two machines R_1 and R_2, run them in interleaving steps:

$$1 \text{ step of } R_1,\ 1 \text{ step of } R_2,\ 2 \text{ steps of } R_1,\ 2 \text{ steps of } R_2,\ \ldots$$

- Ensures: if either machine halts, we eventually discover it.
- Used to combine recognizers into a decider (when both sides are recognizable).

Theorem: $RE \cap coRE = Decidable$

Theorem

If L and $\overline{L}$ are recognizable, then L is decidable.

Proof (Constructive)

Let R recognize L and S recognize $\overline{L}$. On input w:

1. Dovetail simulations of $R(w)$ and $S(w)$.
2. If R accepts, accept.
3. If S accepts, reject.

Exactly one of $w \in L$ or $w \in \overline{L}$ holds, so one recognizer must accept; dovetailing guarantees halting.

Warm-up Decidable: A_{DFA}

$$A_{DFA} = \{B, w \mid B \text{ accepts } w\}$$

Decider

Simulate B on w for exactly $|w|$ steps of input consumption; halt with accept/reject by final state.

Decidable: E_{DFA} (Emptiness)

$$E_{DFA} = \{B \mid L(B) = \emptyset\}$$

Decider (Graph Reachability)

Compute all states reachable from start via BFS/DFS. If no accepting state is reachable, accept (empty); otherwise reject.

Decidable: ALL_{DFA} (Universality)

$$ALL_{DFA} = \{B \mid L(B) = \Sigma^*\}$$

Decider

Construct complement DFA $\overline{B}$ (swap accept/nonaccept). Then $L(B) = \Sigma^*$ iff $L(\overline{B}) = \emptyset$. So decide ALL_{DFA} by reducing to E_{DFA}.

Decidable: EQ_{DFA} (Equivalence)

$$EQ_{DFA} = \{B_1, B_2 \mid L(B_1) = L(B_2)\}$$

Decider

Compute symmetric difference:

$$L(B_1) \triangle L(B_2) = (L(B_1) \setminus L(B_2)) \cup (L(B_2) \setminus L(B_1)).$$

Build DFA for $L(B_1) \triangle L(B_2)$ using product construction. Then $L(B_1) = L(B_2)$ iff $L(B_1) \triangle L(B_2) = \emptyset$. Reduce to E_{DFA}.

Decidable: A_{CFG} (Membership)

$$A_{CFG} = \{G, w \mid G \text{ generates } w\}$$

- Convert G to CNF.
- Use CYK-style dynamic programming (finite table over substrings of w).
- Halts because table has $O(|w|^2)$ cells.

Decidable: E_{CFG} (Emptiness)

$$E_{CFG} = \{G \mid L(G) = \emptyset\}$$

Decider Sketch

Mark variables that can derive terminal strings (productive). Mark variables reachable from start. If start is productive and reachable $\Rightarrow$ nonempty; otherwise empty.

Shift: Turing Machines and the Danger Zone

- Infinite tape, unbounded computation time
- Behavioral questions about arbitrary programs
- Self-reference enables diagonalization

Rule of Thumb

If a question asks about the behavior of *arbitrary* TMs, suspect undecidability.

A_{TM} is Recognizable

$$A_{TM} = \{M, w \mid M \text{ accepts } w\}$$

Recognizer

Simulate M on w. If M accepts, accept. If M rejects or loops, the simulation may not halt.

$$A_{TM} \in RE$$

$HALT_{TM}$ is Recognizable

$$HALT_{TM} = \{M, w \mid M \text{ halts on } w\}$$

Recognizer

Simulate M on w. If M halts (accept or reject), accept.

$$HALT_{TM} \in RE$$

Undecidable: The Halting Problem (Diagonalization)

Assume for contradiction

There exists a decider H for $HALT_{TM}$.

Construct D

On input M:

1. Run $H(M, M)$.
2. If H accepts (halts), then loop forever.
3. If H rejects (does not halt), then halt and accept.

Contradiction

Run $D(D)$: both possibilities contradict H's prediction. Hence $HALT_{TM}$ undecidable.

Mapping Reductions: The Workhorse

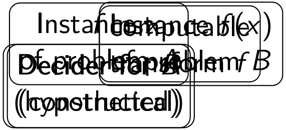
Definition

$A \leq_m B$ if there exists a computable function f such that

$$x \in A \iff f(x) \in B.$$

- If A is undecidable and $A \leq_m B$, then B is undecidable.
- Interpretation: "If we could decide B, we could decide A."

Diagram: How a Mapping Reduction Works



A_{TM} is Undecidable (Reduction from $HALT_{TM}$)

Goal

Show $HALT_{TM} \leq_m A_{TM}$.

Construction of $f(M, w) = M', x$

Given M, w, build M' that on input x:

1. Simulates M on w.
2. If M halts, then *accept* x (e.g., accept immediately when M halts).
3. If M never halts, M' never accepts.

Then $M, w \in HALT_{TM}$ iff $M', x \in A_{TM}$.

Conclusion

If A_{TM} decidable, then $HALT_{TM}$ decidable. Contradiction. So A_{TM} undecidable.

Undecidable: E_{TM} (Emptiness of TM Languages)

$$E_{TM} = \{M \mid L(M) = \emptyset\}$$

Reduction from A_{TM}

Given M, w, build M' such that on input x:

1. Simulate M on w.
2. If M accepts w, accept x.
3. Otherwise (reject/loop), do not accept x.

- If M accepts w, then $L(M') = \Sigma^*$ (nonempty).
- If M does not accept w, then $L(M') = \emptyset$.

Conclusion

Deciding E_{TM} would decide A_{TM}. Hence E_{TM} undecidable.

Undecidable: EQ_{TM} (Equivalence)

$$EQ_{TM} = \{M_1, M_2 \mid L(M_1) = L(M_2)\}$$

Reduction from E_{TM}

Let $M_\emptyset$ be a TM that rejects every input (so $L(M_\emptyset) = \emptyset$). Then

$$M \in E_{TM} \iff M, M_\emptyset \in EQ_{TM}.$$

Conclusion

If EQ_{TM} were decidable, then E_{TM} would be decidable. Contradiction.

Rice's Theorem (Semantic Properties)

Statement (Informal but Standard)

Let P be any nontrivial property of the language recognized by a TM. Then the language

$$L_P = \{M \mid L(M) \text{ has property } P\}$$

is undecidable.

- **Semantic** = depends only on $L(M)$, not on syntax of M.
- **Nontrivial** = true for some TMs and false for others.

Rice Application 1: ALL_{TM} is Undecidable

$$ALL_{TM} = \{M \mid L(M) = \Sigma^*\}$$

- Property: "language is all strings"
- Semantic? Yes (depends only on $L(M)$)
- Nontrivial? Yes (some TMs accept all strings, some accept none)

Conclusion

By Rice's theorem, ALL_{TM} is undecidable.

Rice Application 2: $FINITE_{TM}$ is Undecidable

$$FINITE_{TM} = \{M \mid L(M) \text{ is finite}\}$$

- Property: "recognized language is finite"
- Semantic? Yes
- Nontrivial? Yes

Conclusion

By Rice's theorem, $FINITE_{TM}$ is undecidable.

Rice Application 3: REG_{TM} is Undecidable

$$REG_{TM} = \{M \mid L(M) \text{ is regular}\}$$

- Property: regularity of $L(M)$
- Semantic and nontrivial

Conclusion

By Rice's theorem, REG_{TM} is undecidable.

Classic Extra: Post Correspondence Problem (PCP)

PCP Instance

A finite set of dominoes (pairs of strings) over Σ:

$$\{(u_1, v_1), \ldots, (u_k, v_k)\}$$

Question

Is there a nonempty sequence of indices $i_1, \ldots, i_m$ such that

$$u_{i_1} \cdots u_{i_m} = v_{i_1} \cdots v_{i_m} \text{ ?}$$

- PCP is a classical undecidable problem (via reduction from A_{TM}).
- Useful source of reductions later (e.g., grammars, tiles, rewriting).

PCP: Mini Worked Example (Solvable Instance)

Dominoes:

$$(1): \ (a, ab) \qquad (2): \ (ba, a)$$

Try sequence $(1, 2)$:

$$u_1 u_2 = a \cdot ba = aba, \qquad v_1 v_2 = ab \cdot a = aba.$$

Conclusion

This instance has a solution.

Undecidable for CFGs: Ambiguity

Language

$$AMB_{CFG} = \{G \mid G \text{ is an ambiguous CFG}\}$$

A CFG is **ambiguous** if some string has two distinct parse trees (two distinct leftmost derivations).

- This is a classic example where CFGs still have undecidable properties.
- Proof uses reductions (often via PCP or TM-encoding constructions).

Takeaway

Not all CFG properties are decidable: membership is decidable, ambiguity is not.

Recognizable vs co-Recognizable: A Useful Mindset

- $HALT_{TM}$ is recognizable (simulate and wait).
- Many complements are *not* recognizable (often shown via reductions).
- Heuristic: "If NO requires proving non-existence of a future event, it may be non-RE."

Big Picture

$$\text{Decidable} \subsetneq (RE \cup coRE) \subsetneq \mathcal{P}(\Sigma^*)$$