Chomsky Hierarchy, Decidability and Undecidability

Language classes and the limits of computation

Grammar hierarchy • Decision problems • Undecidability proofs

Learning outcomes

Classify

Identify the four grammar types and their equivalent machine models.

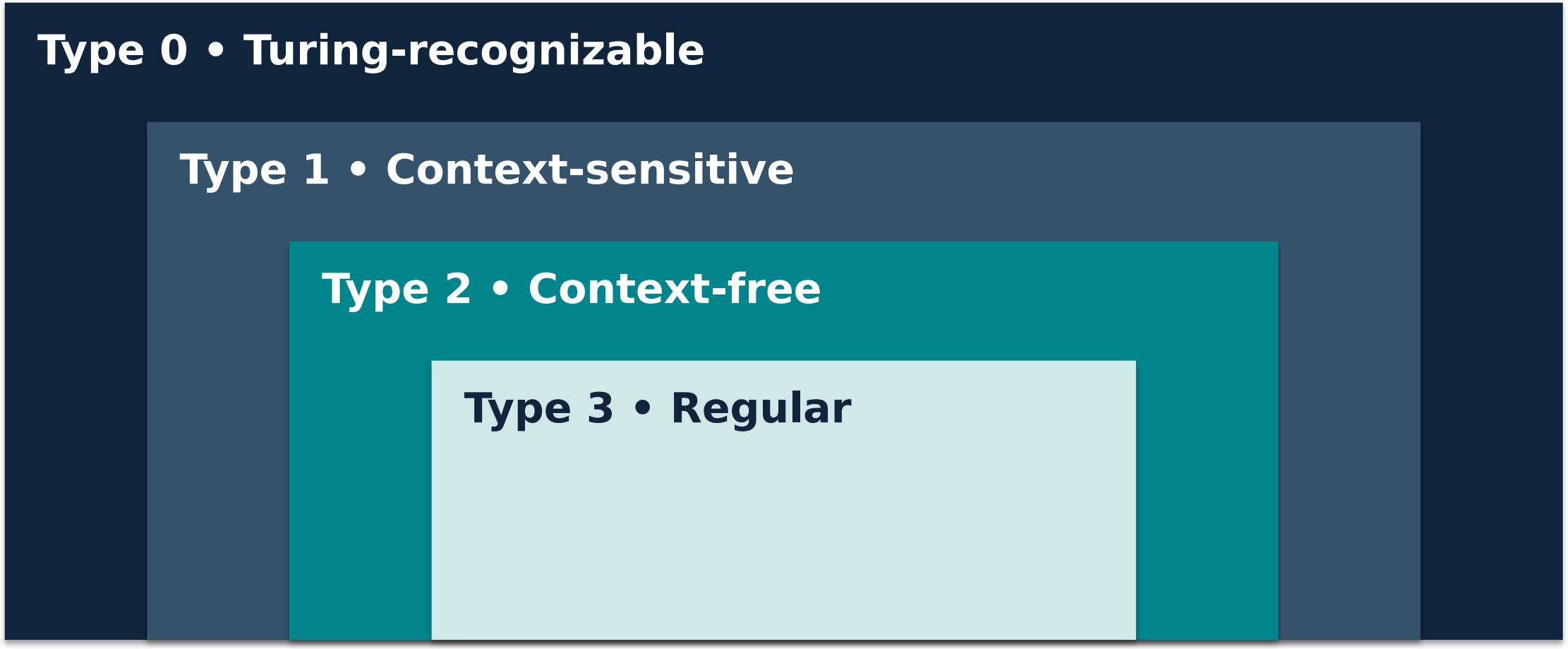
Decide or recognize

Distinguish decidable, recognizable and co-recognizable languages.

Prove limitations

Use diagonalization and mapping reductions to demonstrate undecidability.

The Chomsky hierarchy



Grammar restrictions

Type 3: regular

Right-linear rules $A \to aB$ or $A \to a$ (and ε where allowed). Equivalent to finite automata.

Type 2: context-free

$A \to \alpha$, where A is a single variable. Equivalent to nondeterministic pushdown automata.

Types 1 and 0

Type 1: noncontracting rules $\alpha \to \beta$ with $|\alpha| \leq |\beta|$ (protected start ε exception). Type 0: unrestricted rules with a variable on the left.

Hierarchy: grammar and machine models

Type	Language class	Machine	Example
3	Regular	Finite automaton	a*b*
2	Context-free	NPDA	$\{a^n b^n\}$
1	Context-sensitive	Linear bounded automaton	$\{a^n b^n c^n\}$
0	Turing-recognizable	Turing machine	A_TM

Examples with $n \geq 0$; each listed class strictly contains the class above.

Why the inclusions are strict

Regular ⊊ context-free

$\{a^n b^n : n \geq 0\}$ is context-free but not regular (Lecture 31).

Context-free ⊊ context-sensitive

$\{a^n b^n c^n : n \geq 0\}$ is not context-free (Lecture 39); bounded tape suffices to match triples.

Context-sensitive ⊊ recognizable

LBA acceptance can be decided by finite configuration reachability. A_{TM} is recognizable but undecidable.

Decision problems and encodings

Decision problem

A yes/no question is encoded as a language of yes-instances.

Example: DFA membership

$A_{DFA} = \{\langle D, w\rangle : D \text{ is a DFA and accepts } w\}$. Simulate D on w and inspect the final state.

Encoding convention

Machine descriptions and input strings are finite words. Reject malformed encodings before running the decision algorithm.

Deciders vs. recognizers

DECIDER	RECOGNIZER
Member → accept	Member → accept
Nonmember → reject	Nonmember → reject or loop
Always halts	May not halt

Every decidable language is recognizable; the converse is false.

Decidable problems for FA and CFG

Finite automata

Membership: simulate. Emptiness: search reachable states. Equivalence: test emptiness of the symmetric difference.

Context-free grammars

Membership: CYK after CNF conversion (handle ε separately). Emptiness: mark generating variables until a fixed point.

Boundary

Equivalence of arbitrary CFGs is undecidable, unlike equivalence of DFAs.

The Turing machine acceptance problem

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

A universal TM can simulate M on w. If M accepts, accept; if M rejects, reject; if M loops, the simulation loops.

Recognizable

The universal simulation recognizes A_{TM}.

Undecidable

There is no algorithm that always halts and correctly decides whether an arbitrary TM accepts its input. Proof follows.

Diagonalization: assume a decider exists

Assumption for contradiction

Suppose H decides A_TM. On ⟨M,w⟩, it always halts and answers accept or reject correctly.

Construct a new machine D

On input ⟨M⟩, run H on ⟨M,⟨M⟩⟩. If H accepts, D rejects. If H rejects, D accepts.

Key observation

D itself is a valid TM and always halts because the assumed H always halts.

Diagonalization: the contradiction

Run D on its own encoding

Evaluate D(⟨D⟩). By construction it accepts precisely when H says D does not accept ⟨D⟩.

Two impossible cases

If D accepts, H must accept, causing D to reject. If D rejects, H must reject, causing D to accept.

Conclusion

The assumed decider H cannot exist. A_TM is recognizable but undecidable.

The halting problem

HALT_TM = {⟨M,w⟩ : M halts on w}

Halting includes both accepting and rejecting. Acceptance and halting are different questions.

Recognizer

Simulate M on w; accept as soon as it halts, regardless of its decision.

Undecidability

A hypothetical halting decider could decide A_TM: if M will not halt, reject; otherwise simulate until it halts and return its decision.

Mapping reductions

Definition

$A \leq_m B$ means a total computable function f satisfies $x \in A$ if and only if $f(x) \in B$.

Transfer of decidability

If B is decidable, decide A by computing f(x) and then running a decider for B.

Proof strategy

To prove B undecidable, choose a known undecidable A and show $A \leq_m B$. The direction matters.

Worked reduction: acceptance to halting

Input

Given $\langle M,w\rangle$, construct a machine N that ignores its own input and simulates M on w.

Behavior of N

If M accepts w, N halts. If M rejects w, N loops forever. If M loops, N also loops.

Reduction

$f(\langle M,w\rangle) = \langle N,\varepsilon\rangle$. Then M accepts w iff N halts on ε. Map malformed inputs to a fixed looping machine on ε.

Recognizability and complements

Co-recognizable

L is co-recognizable if its complement is recognizable.

Theorem

L is decidable iff both L and its complement are recognizable. Run both recognizers in alternating steps; one must eventually accept.

Application

Since A_TM is recognizable but undecidable, its complement cannot be recognizable.

Classroom exercises

Exercise 1

Classify a^*b^*, $\{a^n b^n\}$, $\{a^n b^n c^n\}$ by their smallest Chomsky type.

Exercise 2

Is every undecidable language recognizable? Explain using the complement of A_TM.

Exercise 3

Why does a reduction from an undecidable language to B establish undecidability of B, but not the reverse direction?

Solutions and recap

Exercise 1

Types 3, 2 and 1 respectively; use strict hierarchy witnesses.

Exercise 2

No. The complement of A_TM is not recognizable and is undecidable (otherwise A_TM would be decidable).

Exercise 3

A decider for B would decide the known undecidable source A. A reduction in the reverse direction provides no such contradiction.