Recap: Recognize vs. Decide

For given initial tape contents, a Turing machine could
- halt in the accept state
- halt in the reject state
- loop forever (not halt)

**Recognize**: accept all and only the strings in the language
(reject or loop otherwise)

**Decide**: TM never loops (either accepts or rejects)
Deciding is stronger than recognizing.
Some Turing-recognizable languages are not Turing-decidable
(see later).

From DFAs to Turing Machines

Extra capabilities of TM:
1. can move both ways on tape (revisit input)
2. can write on tape
3. can use additional unlimited memory

Adding just (1): two-way automata (recall CS 250)
Result (surprising?): same as normal DFA.

Add (1) and (2): linear bounded automata
context-sensitive languages

Do Details Matter?

A Turing Machine is a 7-tuple . . .
What can change in the definition while keeping the essence?

- Automata: DFA / NFA / ε-transitions: same
- Pushdown automata:
  - Normalizing moves (either push or pop): same expressiveness
  - Nondeterminism mattered!
“Robustness” of definition

First Change: Stay Put

Our looping constructions so far often “overshoot” by one:
“find first symbol of certain kind”
when found, must do something: move left or right
but perhaps we want to stay / start a sweep there

Change transition function to \( \delta : Q \times \Gamma \rightarrow Q \times \Gamma \times \{L,R,S\} \)
\( S = \text{stay in place} \)
Does this change anything?
Clearly we can model “stay put” by inserting move right, then left.
why not the other way around?

Multitape Turing Machines

- Each tape has its own read/write head
- Input is initially on tape 1, other tapes blank
- Heads move/read/write simultaneously
\( \delta : Q \times \Gamma^k \rightarrow Q \times \Gamma^k \times \{L,R,S\}^k \)
One transition:
\( \delta(q_i,a_1,\ldots,a_k) = (q_j,b_1,\ldots,b_k,R,L,\ldots,R) \)
Does this add expressive power?
Proving Equivalence: Simulation

Show that multitape TM $M$ can be simulated by single-tape TM $S$.

Nondeterministic Turing Machines

New transition function allows choice:
$$\delta : Q \times \Gamma \rightarrow \mathcal{P}(Q \times \Gamma \times \{L, R\})$$

⇒ computation tree, accept if some branch accepts

Theorem: Every nondeterministic TM has an equivalent deterministic TM

Will prove by constructing a three-tape TM that simulates all nondeterministic choices of $N$.

Encoding Computation Branches

Assume max. degree (nondeterminism) = 3. Current path: 1 2 2.
Next paths: 1 2 3, 1 3 1, 1 3 2, etc.

- Copy original input (tape 1) to tape 2
- Simulate run of $N$ on computation string from tape 3
- Update tape 3 to next string

Enumerators

Turing-recognizable languages are also called recursively enumerable

Think Turing machine with attached printer
prints all strings in the language
may print infinite list, or halt if language is finite
repetitions are allowed (don’t matter)

Theorem: A language is Turing-recognizable iff some enumerator enumerates it.