We know $L = \{0^n1^n0^n \mid n \geq 0\}$ is not a CFL (pumping lemma)

Can we show $L$ is decidable?

- Construct a decider $M$ such that $L(M) = L$
- A **decider** is a TM that always halts (in $q_{\text{acc}}$ or $q_{\text{rej}}$) and is guaranteed not to go into an infinite loop for any input

Input: 000001111100000

Idea: Mark off matching 0s, 1s, and 0s with Xs (left end marked with blank)

<table>
<thead>
<tr>
<th>Input</th>
<th>000001111100000</th>
</tr>
</thead>
<tbody>
<tr>
<td>Marked</td>
<td>000001111100000</td>
</tr>
<tr>
<td>0s, 1s, Xs</td>
<td>_00001111100000</td>
</tr>
<tr>
<td>0s</td>
<td>_0000X111100000</td>
</tr>
<tr>
<td>Xs</td>
<td>_0000X1111X0000</td>
</tr>
<tr>
<td>Blank</td>
<td>_X000X1111X0000</td>
</tr>
</tbody>
</table>

....
Idea for a Decider for \( \{0^n1^n0^n \mid n \geq 0\} \)

- **General Idea**: Match each 0 with a 1 and a 0 following the 1.
- **Implementation Level Description** of a Decider for L:
  
  On input w:
  1. If first symbol = blank, ACCEPT
  2. If first symbol = 1, REJECT
  3. If first symbol = 0, Write a blank to mark left end of tape
     a. If current symbol is 0 or X, skip until it is 1. REJECT if blank.
     b. Write X over 1. Skip 1’s/X’s until you see 0. REJECT if blank.
     c. Write X over 0. Move back to left end of tape.
  4. At left end: Skip X’s until:
     a. You see 0: Write X over 0 and GOTO 3a
     b. You see 1: REJECT
     c. You see a blank space: ACCEPT
State Diagram

Try running the decider on:

⇒ 010, 001100, … ⇒ ACCEPT
⇒ 0, 000, 0100, … ⇒ REJECT
⇒ What about 010010?

Note: Some transitions to $q_{REJ}$ (e.g., from $q_{skip0}$) are not shown to avoid clutter

R. Rao, CSE 322
Houston, we have a problem…with our Turing machine.
What’s the problem?

The decider accepts incorrect strings:

- 010010, 010001100 \(\rightarrow\) ACCEPT!!!
- Accepts \((0^n1^n0^n)^*\)

Need to fix it…How??
A Simple Fix (to the Decider)

Scan initially to make sure string is of the form 0*1*0*

On input w:
1. If first symbol = blank, ACCEPT
2. If first symbol = 1, REJECT
3. If first symbol = 0: if w is not in 00*11*00*, REJECT; else,
   Write a blank to mark left end of tape
   a. If current symbol is 0 or X, skip until it is 1. REJECT if blank.
   b. Write X over 1. Skip 1’s/X’s until you see 0. REJECT if blank.
   c. Write X over 0. Move back to left end of tape.
4. At left end: Skip X’s until:
   a. You see 0: Write X over 0 and GOTO 3a
   b. You see 1: REJECT
   c. You see a blank space: ACCEPT

Add this
The Decider TM for L in all its glory

New part tests for 00*11*00*
Can we augment the power of Turing machines with various accessories?
Varieties of TMs

- What if we allow nondeterminism?
- What if we allow multiple tapes?
- What if my date doesn’t show up tonight?
Various Types of TMs

- **Multi-Tape TMs**: TM with $k$ tapes and $k$ heads
  \[ \delta: Q \times \Gamma^k \rightarrow Q \times \Gamma^k \times \{L,R\}^k \]
  \[ \delta(q_i, a_1, \ldots, a_k) = (q_j, b_1, \ldots, b_k, L, R, \ldots, L) \]

- **Nondeterministic TMs (NTMs)**
  \[ \delta: Q \times \Gamma \rightarrow \text{Pow}(Q \times \Gamma \times \{L,R\}) \]
  \[ \delta(q_i, a) = \{(q_1, b, R), (q_2, c, L), \ldots, (q_m, d, R)\} \]

- **Enumerator TM for $L$**: Prints all strings in $L$ (in any order, possibly with repetitions) and only the strings in $L$

- **Other types**: TM with Two-way infinite tape, TM with multiple heads on a single tape, 2D infinite tape TM, Random Access Memory (RAM) TM, etc.
Surprise!
All TMs are born equal…

- Each of the preceding TMs is equivalent to the standard TM
  - They recognize the same set of languages (the Turing-recognizable languages)

- Proof idea: Simulate the “deviant” TM using a standard TM

- **Example 1: Multi-tape TM on a standard TM**
  - Represent k tapes sequentially on 1 tape using separators #
  - Use new symbol $a$ to denote a head currently on symbol $a$

```
0 1 ............
```
```
b a h ............
```
```
3 2 2 ............
```

≡
```
# 0 1 # b a h # 3 2 2 # ........
```

(See text for details)
Example 2: Simulating Nondeterminism

Any nondeterministic TM $N$ can be simulated by a deterministic TM $M$

- NTMs: $\delta: Q \times \Gamma \rightarrow \text{Pow}(Q \times \Gamma \times \{L,R\})$
- No $\varepsilon$ transitions but can simulate them by reading and writing same symbol
- $N$ accepts $w$ iff there is at least 1 path in $N$’s tree for $w$ ending in $q_{\text{ACC}}$

General proof idea: Simulate each branch sequentially

- Proof idea 1: Use depth first search?
  No, might go deep into an infinite branch and never explore other branches!
- Proof idea 2: Use breadth first search
  Explore all branches at depth $n$ before $n+1$
Simulating Nondeterminism: Details, Details

- Use a 3-tape DTM M for breadth-first traversal of N’s tree on w:
  - Tape 1 keeps the input string w
  - Tape 2 stores N’s tape during simulation along 1 path (given by tape 3) up to a particular depth, starting with w
  - Tape 3 stores current path number
    E.g. $\varepsilon = \text{root node } q_0$
    $213 = \text{path made up of 3rd child of 1st child of 2nd child of root}$

- See text for more details
What about other types of computing machines?

- Enumerator TMs (or Printer Machines)
- TMs with 2-Way Infinite Tape
- TMs with Multiple Read/Write Heads
- TMs with 2-Dimensional Tape
- TMs with Random Access Memory (RAM)
The Church-Turing Thesis

- Various definitions of “algorithms” were shown to be equivalent in the 1930s
- **Church-Turing Thesis**: “The intuitive notion of algorithms equals Turing machine algorithms”
  - Turing machines serve as a precise formal model for the intuitive notion of an algorithm
- “Any computation on a digital computer is equivalent to computation in a Turing machine”

Dude, that’s pretty deep…
Recap: Recognizable versus Decidable Languages

- A language \( L \) is called **Turing-Recognizable** if there exists a TM \( M \) such that \( L(M) = L \)
  - Note: \( M \) need not halt on all inputs but it should halt and accept all and only those strings that are in \( L \); it can reject strings by either going to \( q_{\text{rej}} \) or by looping forever

- A TM is a **decider** if it halts on all inputs

- A language \( L \) is **decidable** if there exists a decider \( D \) such that \( L(D) = L \)
Closure Properties of Decidable Languages

✨ Decidable languages are closed under ∪, °, *, ∩, and complement

✨ Example: Closure under ∪

✨ Need to show that union of 2 decidable L’s is also decidable
  Let M1 be a decider for L1 and M2 a decider for L2
  A decider M for L1 ∪ L2:
    On input w:
    1. Simulate M1 on w. If M1 accepts, then ACCEPT w. Otherwise, go to step 2 (because M1 has halted and rejected w)
    2. Simulate M2 on w. If M2 accepts, ACCEPT w else REJECT w.
   M accepts w iff M1 accepts w OR M2 accepts w
   i.e. L(M) = L1 ∪ L2
Closure Properties

- Consider the proof for closure under $\cup$
  A decider $M$ for $L_1 \cup L_2$:
    On input $w$:
    1. Simulate $M_1$ on $w$. If $M_1$ accepts, then ACCEPT $w$. Otherwise, go to step 2 (because $M_1$ has halted and rejected $w$)
    2. Simulate $M_2$ on $w$. If $M_2$ accepts, ACCEPT $w$ else REJECT $w$.
    M accepts $w$ iff $M_1$ accepts $w$ OR $M_2$ accepts $w$
    i.e. $L(M) = L_1 \cup L_2$

Will the same proof work for showing Turing-recognizable languages are closed under $\cup$? Why/Why not?

Uh…I dunno. Wait, will $M_1$ always halt?!

M1 may never halt but $w$ may be in $L_2$
Closure Properties of Recognizable Languages

- Turing recognizable languages are closed under ∪
  
  A TM M for L1 ∪ L2:
  
  On input w:
  
  Simulate M1 and M2 *alternatively* on w *step by step*.
  
  If either accepts, then ACCEPT w.
  
  If both halt and reject w, then REJECT w.

L(M) = L1 ∪ L2

If either M1 or M2 accepts, then M accepts w (even if one of them loops, M will accept and halt when the other accepts and halts because M alternates between M1 and M2).

Otherwise, M rejects w by halting or by looping forever.
Closure for Recognizable Languages

- Turing-Recognizable languages are closed under $\cup$, $\circ$, $*$, and $\cap$ (but not complement! We will see this later)

- Example: **Closure under $\cap$**
  Let $M_1$ be a TM for $L_1$ and $M_2$ a TM for $L_2$ (both may loop)
  A TM $M$ for $L_1 \cap L_2$:
  - On input $w$:
    1. Simulate $M_1$ on $w$. If $M_1$ halts and accepts $w$, go to step 2. If $M_1$ halts and rejects $w$, then REJECT $w$. (If $M_1$ loops, then $M$ will also loop and thus reject $w$)
    2. Simulate $M_2$ on $w$. If $M_2$ halts and accepts, ACCEPT $w$. If $M_2$ halts and rejects, then REJECT $w$. (If $M_2$ loops, then $M$ will also loop and thus reject $w$)
  - $M$ accepts $w$ iff $M_1$ accepts $w$ AND $M_2$ accepts $w$ i.e. $L(M) = L_1 \cap L_2$