**Multiple tapes**

\[ \delta : Q \times \Gamma^k \rightarrow Q \times \Gamma^k \times \{L,R\}^k \]

**Lemma:** If \( L \) is accepted by a multitape TM, then \( L \) is accepted by a single tape TM.

**Proof idea:** Use tracks, with special markers for each tape head.

\[
\begin{array}{c|c|c}
w & x & y \\
\hline
\hat{a} & b \\
\hline
m & n & \omega
\end{array}
\]

\( M' \) will simulate \( M \) by making multiple passes over tape, 1 for each track. First, it needs to determine all the symbols being read, in order to determine the next move. So \( k \) symbols must be stored in state. Then, once the next move is determined, it must update each track with new symbol and new tape head position.

---

**Nondeterministic TM's**

\[ \delta : Q \times \Gamma \rightarrow P( Q \times \Gamma \times \{L,R\} ) \]

**Lemma:** If \( L \) is accepted by a nondet. TM \( N \), then \( L \) is accepted by some deterministic TM \( D \).

**Proof idea:** \( D \) will simulate \( N \) by trying all possible branches of \( N \)'s computation on input \( w \). If \( D \) finds an accept state on some branch, it accepts. Otherwise, \( D \) does not accept.

**QU:** How should \( D \) search the computation tree??

**Answer:** How represent a branch?

```
  \[ \text{Input} \rightarrow \text{Simulate} \rightarrow \text{Branch Choice} \]
```
Robustness of TM's

Is this surprising??

TM's --> "Idealized" programming languages

Programming languages, if sufficiently general, can do the same computations

Ex. C, Lisp, Java

TM's make formal our informal notion of algorithm

Church-Turing Thesis: The Turing machine model (and any reasonable variant of it) embodies our informal notion of algorithm.

Can't be proved!

From now on, TM --> Algorithms

Decidable Problems

Represent problem using language

Problem: Is w accepted by DFA B?

\[ A_{DFA} = \{ <B, w> \mid B \text{ is a DFA that accepts } w \} \]

<> denotes encoding as string

Language decidable --> Problem decidable

Th. \[ A_{DFA} \] is a decidable language.

Proof idea: Decider M will simulate D on input w.

If D would accept w, then M accepts; D rejects, M rejects.
### Decidable Problems

\[
A_{\text{NFA}} = \{\langle B, w \rangle \mid B \text{ is an NFA, } B \text{ accepts } w\}
\]

**Th.** \(A_{\text{NFA}}\) is decidable.

*Proof idea:* Define \(N\) to use \(TM\ M\) (last \(th\) as subroutine.

1. Convert NFA \(B\) to DFA \(D\).
2. Run \(TM\ M\) on input \(\langle D, w \rangle\)
3. If \(M\) accepts, then \(N\) accepts; \(M\) rejects, \(N\) rejects.

\[q_0 q_1 q_2 q_3 \# 01 \# 01 X Y \# \langle \text{nd. table} \rangle \# \ldots \]
\[\{q_0\} \{q_0, q_1\} \ldots \# \ldots \# \langle \text{det. table} \rangle \#\]

---

### Decidable Problems

\[
E_{\text{DFA}} = \{\langle B \rangle \mid B \text{ is a DFA, } L(B) \text{ is nonempty}\}
\]

**Th.** \(E_{\text{DFA}}\) is decidable.

*Lemma:* For DFA \(B\), \(L(B)\) is nonempty iff

\(B\) accepts a string of length at most \(|Q|\).

*Trivial.*

Suppose \(B\) accepts some word. Let \(w\) be a shortest word accepted by \(B\).

If \(|w| < |Q|\), then we are done. So suppose \(|w| > |Q|\). Consider the sequence of states of \(B\) on \(w\).
### Decidable Problems

<table>
<thead>
<tr>
<th>DECIDABLE PROBLEMS</th>
</tr>
</thead>
<tbody>
<tr>
<td>$w = w_1 w_2 w_3 \ldots w_m$, $m &gt;</td>
</tr>
<tr>
<td>$q_0 \rightarrow q_1 \leftarrow q_2 \ldots \leftarrow q_m$</td>
</tr>
</tbody>
</table>

Since there are more than $|Q|$ states in this sequence, there must be a repetition of states, by the pigeonhole principle. But then we can take a snippet $z$ out of $w$, and get a shorter string that is accepted by $B$.

Contradiction. Therefore $w$ is of length $< |Q|$.

**Back to Theorem:**

*Decider $D$ will, on input $<B>$, simulate $B$ on inputs of length 0 to $|Q|$. If $B$ accepts any string of that length, $D$ accepts. If no such string is found, $D$ rejects. By the Lemma, $D$ accepts $E_{DFA}$.***

### Decidable Problems

<table>
<thead>
<tr>
<th>DECIDABLE PROBLEMS</th>
</tr>
</thead>
<tbody>
<tr>
<td>$EQ_{DFA} = {&lt;A,B&gt; \mid A,B \text{ are DFA, and } L(A) = L(B) }$</td>
</tr>
</tbody>
</table>

**Th:** $EQ_{DFA}$ is decidable.

**Proof:** We construct a DFA $C$ with

$L(C) = (L(A) \cap \overline{L(B)}) \cup (\overline{L(A)} \cap L(B))$

$(\text{symmetric difference})$

$L(A)$

$L(B)$

$C$ can be constructed by TM $M$ using algorithms we developed for closure under $\cup$, $\cap$, $\overline{}$. Since $L(C)$ is empty iff. $L(A) = L(B)$, we can use previous theorem for deciding $L(C)$ nonempty.
Decidability Problems for CFG’s

\[ A_{\text{CFG}} = \{ <G, w> \mid G \text{ is a CFG, } G \text{ generates } w \} \]

**Th.** \( A_{\text{CFG}} \) is decidable.

Proof idea: TM could start with start var. \( S \) and try all derivations. If \( w \) is not in \( L(G) \), there will be none!

So trying all derivations to see if \( w \) would not yield decider.

To define a decider, we first put \( G \) in Chomsky normal form.

Then we can bound the number of steps in a derivation TM needs to try to \( 2|w|+1 \). QUI: WHY?

The decider \( D \) will then list all derivations of length 0 to \( 2|w|+1 \). If any derivation generates \( w \), \( D \) accepts; otherwise, reject.

---

Decidability Problems on CFG’s

\[ E_{\text{CFG}} = \{ <G> \mid G \text{ is a CFG, } L(B) \text{ is nonempty} \} \]

**Th.** \( E_{\text{CFG}} \) is decidable.

Proof idea: Can’t try all strings (won’t be decidable).

Consider any parse tree for a string \( w \).

If parse tree has path of length > \( |V| \),

then there is a repetition of variables along that path, and we can produce a shorter parse tree. We can repeat this process so that there is a parse tree with no path of length > \( |V| \). Therefore, if \( G \) generates any string, it generates a string whose parse tree has no path > \( |V| \).
### Decidable Problems on CFG's

**Th.** $E_{\text{CFG}}$ is decidable. *(continued)*

A decider $D$ for this problem will generate a collection $C$ of parse trees on its tape. $C$ will initially contain $S$. $D$ repeatedly adds to $C$ any tree that can be obtained from one already in $C$ by applying a single rule, such that:

1. the new tree is not in $C$
2. the new tree does not have any path of length $> |V|$

Since there are only a finite number of trees of fixed length, the TM $D$ will eventually complete $C$. $L(G)$ is nonempty iff at least one tree in $C$ has only terminals as leaves.

---

### Decidable Problems for CFG's

$E_{\text{CFG}} = \{ <G,H> \mid G,H \text{ are CFG, and } L(G) = L(H) \}$

$E_{\text{CFG}}$ decidable??

Can't use same idea we used for DFA's (symmetric diff.) because CFL’s are NOT closed under complement.

*Turns out $E_{\text{CFG}}$ is NOT decidable. We can’t prove it yet.*

**Th.** Every CFL is decidable. *(HW 3)*

---

[Diagram showing relationship between Regular Languages (Reg.), Context-Free Languages (CFL), Decidable (Decid.), and Turing Recognizable (Rec.).]