[Shawed "Collatz 3n+1 Conjecture" Turing Machine.]

We believe the language is $(01)^*$ but don't even know whether it's decidable! We can add $\varepsilon$ to make the language equal $(01)^*$. Then if we could decide $ALL_m$ for this 8-state machine, we could solve the conjecture.

**Theorem:** $ALL_m$ is not even recognizable.

Note: Earlier we showed $ALL_m \leq_m ALL_{tm}$. Here is $K_{tm} \leq_m ALL_{tm}

\[
M \leftarrow M' \quad \text{down input } w
\]

"All or Nothing Switch":

\[
\begin{align*}
&M' \in K \quad \Rightarrow \quad L(M') = \Sigma^* \quad \underbrace{\text{Accept } w.}_{\text{OK}} \\
&M' \notin K \quad \Rightarrow \quad L(M') = \emptyset \\
&f(\langle M \rangle) = \langle M' \rangle \in ALL_{tm}.
\end{align*}
\]

Now show $B_{tm} \leq_m ALL_{tm}$ via the "Delay Switch." To reduce we need:

\[
\begin{align*}
&M \in B_{tm} \quad \Rightarrow \quad L(M'') = \Sigma^* \\
&M' \in ALL_{tm} \quad \Rightarrow \quad L(M'') = \emptyset
\end{align*}
\]

So:

\[
\begin{align*}
&M \in B_{tm} \quad \Rightarrow \quad (\forall w) \text{ } M \text{ does not accept } \langle M \rangle \text{ in } 1W1 \text{ steps (or at all)} \\
&M' \in ALL_{tm} \quad \Rightarrow \quad (\forall w) \text{ } M' \text{ does not accept } \langle M' \rangle \text{ in } 1W1 \text{ steps (or at all)}
\end{align*}
\]

\[
\begin{align*}
&M \in B_{tm} \quad \Rightarrow \quad L(M'') = \Sigma^* \\
&M' \in ALL_{tm} \quad \Rightarrow \quad L(M'') = \emptyset
\end{align*}
\]
Defn: A language $B$ is hard for a class $C$ of languages under $\leq_m$ if:

For all $A \in C$, $A \leq_m B$.

$B$ is complete for $C$ (under $\leq_m$) if also $B \in C$.

Theorem: $A_{TM}$ is complete for $REB$ under $\leq_m$.

Proof: First, $A_{TM} \in REB$. Let any $A \in REB$ be given.

Let $M$ be a deterministic Turing machine such that $A = L(M)$.

$M$ is fixed, so the function $f(x) = \langle M, x \rangle$ is computable.

And $x \in A \iff M$ accepts $x \iff \langle M, x \rangle \in A_{TM}$.

So $A \leq_m A_{TM}$, and since $A \in REB$ is arbitrary, $A_{TM}$ is complete.

Since $A_{TM}$ is not even recognizable, then if $A_{TM} \leq_m B$,

the $B$ is not recognizable either, let alone decidable.

Observe: $\langle M \rangle \in E_{TM} \iff L(M) = \emptyset \iff M$ has no accepting computation.

The language $L(M) = \{ w \mid w \text{ is a valid accepting computation of } M \}$.

Now, for any fixed $M$, $L(M)$ is decidable — in fact, decidable in linear time.

The length $N = |w_0(w)| = \max \{ \ell(T) \mid w = w_0(T); \ell(T) < \ell(w) \}$

because $|w_0(T)| = n + 1$. In $t$ steps, $T$ can grow by at most $t$ states.
Moreover, the decider just needs to check \((A) : i \in \text{st}) \overline{I}_{j-1} \overline{m} I_j\)
and \(I_0 = \infty \) for some \(x\) and \(I_j = \text{acc} \) by "good housekeeping" for \(M\).

It can do this without using any tape besides the input chain of \(I\).

Hence the decider can be a \textbf{Linear Bounded Automaton (LBA)}.

\[
E_{\text{LM}} \equiv \text{LBA} \quad \text{via the conversion from } M \text{ to an LBA for } V_M.
\]

The simplest LBA I know emulates having 2 tape heads that
check \(I_0 \overline{m} I_1 \overline{m} I_2 \overline{m} I_3 \overline{m} I_4 \overline{m} \ldots I_{k-1} \overline{m} I_k\)
in \textit{tag-team} fashion. Hence emptiness \(E_{\text{LM}}\) is undecidable. What other \textit{tag-team} machines can check computations?

One DPDA cannot: Text shows how we can help it by writing
every odd ID in reverse: \(I_0 \# I_1 \overline{m} \# I_2 \overline{m} \# I_3 \overline{m} \# I_4 \overline{m} \ldots I_{k-1} \overline{m} \).

And NPDA can do \(A\;\text{NPDA}\) can push on \(I_0\), pop-compare on \(I_2\), but then has nothing left on the stack by which to test \(I_1 \overline{m} I_2\).

However, \(V_M\) does equal \(L(P_1) \cap L(P_2)\) for two DPDAs that
\textit{tag-team} checking the odd and even parity of \(I\).

Moreover, an \textit{NPDA} can recognize when \(I_0 \ldots I_k\) is invalid by guessing which pair fails.

Thus these two problems \(\text{a) RNSP: CFGs } G_1 \text{ and } G_2\)
\(\text{are both undecidable: } E_{\text{LM}} \text{ reduces to both of them.}\)

\(\text{b) ALL CFGs }\)
\(L(G_1) \cap L(G_2) = \emptyset ?\)
Nevertheless, **ALL DFA** is decidable, indeed in time $O(n^2) \in O(1|\Sigma|^2)$ ignoring log factors, by *breadth-first search*

\[O(nm) = O(m(n \log m))\]  

**Defn:** A language $L$ belongs to the class $P$ of problems decidable in deterministic polynomial time if there is a polynomial $p(n)$ and a det. TM $M$ s.t. for all $x \in \Sigma$,

$M$ has an accepting config of length $\leq p(|x|)$ and $L = L(M)$.

**Defn:** $L \in \text{NP}$ if we allow a nondet. TM $N$ in place of $M$.

$x \in L \iff N$ has an acc config of length $\leq p(|x|)$.

$P$-computable for $f$ is defined similarly, and $A \leq^P_\text{co} B$ if there is an $f$ (computable in poly time s.t. $\forall x : x \in A \iff f(x) \in B$.

**Defn:** $B$ is $\text{NP}$ hard $\equiv$ for all $A \in \text{NP}$, $A \leq^P_\text{co} B$.

and $B$ is $\text{NP}$ complete $\equiv$ $B$ is NP-hard and $B$ is NP-complete.

*All* NFA is NP-hard.  (Added). The reason converting the given NFA $N$ into an equivalent DFA $M$ is no good for $P$ is that it runs in $\sim 2^{1|\Sigma|}$ time in worst case. The easier complementary problem, "Is there some string of length $\leq 1|\Sigma|$ that $N$ fails to accept?" does belong in NP (your HW) and is in fact NP-complete, as Thursday will finish by showing.
**Footnotes:** The "Sipser Naming Scheme" extends to other kinds of concepts subscripts by other Types of Machine or Formal Object labeled "T<sub>mac</sub>". Let "T<sub>mac</sub>" stand for a "T<sub>mac</sub>" could be TM or FA or CFL or EFG etc., or "a tandem or completion of a T<sub>mac</sub>(s), etc.

**A<sub>T<sub>mac</sub></sub>:** Given a T<sub>mac</sub> and a x e Σ<sup>*</sup>, is x e L(T<sub>mac</sub>)?<n>\text{just for all FA in §5.1}

**E<sub>T<sub>mac</sub></sub>:** Given a T<sub>mac</sub>, is L(T<sub>mac</sub>) = ∅?<n>

**ALL<sub>T<sub>mac</sub></sub>:** Given a T<sub>mac</sub>, is L(T<sub>mac</sub>) = Σ<sup>*</sup>?

**EQ<sub>T<sub>mac</sub></sub>:** Given T<sub>1</sub> and T<sub>2</sub>, is L(T<sub>1</sub>) = L(T<sub>2</sub>)?

**HALT<sub>T<sub>mac</sub></sub>:** Given T<sub>mac</sub> and x, does T<sub>mac</sub> accept x halt?

**PT<sub>T<sub>mac</sub></sub>:** aka TO<sub>T<sub>mac</sub></sub>: Given a T<sub>mac</sub>, is T<sub>mac</sub> total, i.e. Vx T<sub>x</sub> exists?

**DIST<sub>T<sub>mac</sub></sub>:** Given T<sub>1</sub> and T<sub>2</sub>, is L(T<sub>1</sub>) n L(T<sub>2</sub>) = ∅?

---

<table>
<thead>
<tr>
<th>T&lt;sub&gt;mac&lt;/sub&gt;</th>
<th>DFA</th>
<th>NFA</th>
<th>RegP&lt;sup&gt;o&lt;/sup&gt;</th>
<th>DFOA</th>
<th>NPD&lt;sup&gt;L&lt;/sup&gt;</th>
<th>NLB&lt;sup&gt;G&lt;/sup&gt;</th>
<th>Ptime TMs</th>
<th>DTM&lt;sup&gt;NF&lt;/sup&gt;</th>
<th>Boolean Formulas φ</th>
</tr>
</thead>
<tbody>
<tr>
<td>HALT</td>
<td>Always</td>
<td>Always</td>
<td>Always</td>
<td>Always</td>
<td>Always</td>
<td>Always</td>
<td>Always</td>
<td>Always</td>
<td>Always</td>
</tr>
</tbody>
</table>

---

1. Formally, regular expressions disallow integer powers like (1+1)<sup>3</sup>. If they are allowed, the complexity gets worse.
2. CREC is short for "complete for CREC complex, i.e. = DTM.
3. Not is short for "NP-hard", all complete for CREC.
4. ALL<sub>TM</sub> = e<sub>TM</sub> is a class called "T<sub>mac</sub>".
5. Complete means complete for (NP-cre, i.e., the complements NP<sub>cre</sub> etc. are all NP-cre complete.
6. NF<sub>cre</sub> = { x | (x x) = h[ε] } SAT which is NPC.
7. NF<sub>cre</sub> was only proved decidable a decade ago.
8. The diagonal language D<sub>φ</sub> = { p | machine that don't accept h[c] is decidable but not in p. Try defining this yourself! Hence Ap, ditto.
9. Sipser's text only proves that every CFL is in P. Ap, ditto.

---

**Sipser's notes:** The "Sipser Naming Scheme" extends to other kinds of concepts subscripts by other Types of Machine or Formal Object labeled "T<sub>mac</sub>". Let "T<sub>mac</sub>" stand for a "T<sub>mac</sub>" could be TM or FA or CFL or EFG etc., or "a tandem or completion of a T<sub>mac</sub>(s), etc.

**A<sub>T<sub>mac</sub></sub>:** Given a T<sub>mac</sub> and a x e Σ<sup>*</sup>, is x e L(T<sub>mac</sub>)?

**E<sub>T<sub>mac</sub></sub>:** Given a T<sub>mac</sub>, is L(T<sub>mac</sub>) = ∅?

**ALL<sub>T<sub>mac</sub></sub>:** Given a T<sub>mac</sub>, is L(T<sub>mac</sub>) = Σ<sup>*</sup>?

**EQ<sub>T<sub>mac</sub></sub>:** Given T<sub>1</sub> and T<sub>2</sub>, is L(T<sub>1</sub>) = L(T<sub>2</sub>)?

**HALT<sub>T<sub>mac</sub></sub>:** Given T<sub>mac</sub> and x, does T<sub>mac</sub> accept x halt?

**PT<sub>T<sub>mac</sub></sub>:** aka TO<sub>T<sub>mac</sub></sub>: Given a T<sub>mac</sub>, is T<sub>mac</sub> total, i.e. Vx T<sub>x</sub> exists?

**DIST<sub>T<sub>mac</sub></sub>:** Given T<sub>1</sub> and T<sub>2</sub>, is L(T<sub>1</sub>) n L(T<sub>2</sub>) = ∅?

---

**Table:**

<table>
<thead>
<tr>
<th>T&lt;sub&gt;mac&lt;/sub&gt;</th>
<th>DFA</th>
<th>NFA</th>
<th>RegP&lt;sup&gt;o&lt;/sup&gt;</th>
<th>DFOA</th>
<th>NPD&lt;sup&gt;L&lt;/sup&gt;</th>
<th>NLB&lt;sup&gt;G&lt;/sup&gt;</th>
<th>Ptime TMs</th>
<th>DTM&lt;sup&gt;NF&lt;/sup&gt;</th>
<th>Boolean Formulas φ</th>
</tr>
</thead>
<tbody>
<tr>
<td>HALT</td>
<td>Always</td>
<td>Always</td>
<td>Always</td>
<td>Always</td>
<td>Always</td>
<td>Always</td>
<td>Always</td>
<td>Always</td>
<td>Always</td>
</tr>
</tbody>
</table>

---

1. Formally, regular expressions disallow integer powers like (1+1)<sup>3</sup>. If they are allowed, the complexity gets worse.
2. CREC is short for "complete for CREC complex, i.e. = DTM.
3. Not is short for "NP-hard", all complete for CREC.
4. ALL<sub>TM</sub> = e<sub>TM</sub> is a class called "T<sub>mac</sub>".
5. Complete means complete for (NP-cre, i.e., the complements NP<sub>cre</sub> etc. are all NP-cre complete.
6. NF<sub>cre</sub> = { x | (x x) = h[ε] } SAT which is NPC.
7. NF<sub>cre</sub> was only proved decidable a decade ago.
8. The diagonal language D<sub>φ</sub> = { p | machine that don't accept h[c] is decidable but not in p. Try defining this yourself! Hence Ap, ditto.
9. Sipser's text only proves that every CFL is in P. Ap, ditto.