

Turing Machines

- **Beyond CFGs**
- **What TMs Are**
- **Universal TMs**
- **Strengths**
- **Limits: Uncomputability**

Beyond CFGs

- No CFG generates $\{ a^n b^n c^n \}$
- Unrestricted Grammars (UGs)
- There is a UG for $\{ a^n b^n c^n \}$

No CFG generates $L = \{ a^n b^n c^n \}$

Part 1: Repetition in a Path

- Suppose some G generates L .
- Note that L has arbitrarily long strings,
- so G needs taller and taller trees,
- with longer and longer paths,
- containing some repeated nonterminal, A .

No CFG generates $L = \{ a^n b^n c^n \}$

Part 2: Two-way Balance

- Repetition of A in a path implies
- that for some w and y not both empty,
- $A \Rightarrow^* wAy \Rightarrow^* wwAyy$ and so
- $S \Rightarrow^* vAz \Rightarrow^* v w w x y y z$.

No CFG generates $L = \{ a^n b^n c^n \}$

Part 3: The Contradiction

- $S \Rightarrow^* v w w x y y z$.
- w and y must be homogeneous,
- leading to unequal numbers of a vs b vs c .
- Therefore $\mathcal{L}(G)$ is not L ,
- contradicting the assumption that such a G exists,
- and so there can't be a CFG that generates L .

But there is a UG for $\{ a^n b^n c^n \}$

- $S \rightarrow W D Z \mid \epsilon$ 1. start
- $D \rightarrow A B C D \mid A B C$ 2. n of each
- $C A \rightarrow A C$ 3. sorting
- $C B \rightarrow B C$ is
- $B A \rightarrow A B$ permitted
- $W A \rightarrow a W$ 4. traverse,
- $W B \rightarrow b X$ lowercase,
- $X B \rightarrow b X$ and
- $X C \rightarrow c Y$ check
- $Y C \rightarrow c Y$ the sort
- $Y Z \rightarrow \epsilon$ 5. check step 4

Turing's Question

- His question: how do we compute?
- His answer: read, write, look, control.
 - read: input to □
 - write: output from □
 - look: move (□) tape head
 - control: change state

Turing Machines

- A TM is 5-Tuple: $(Q, \Sigma, \Gamma, q_0, \delta)$
... which, unlike PDAs, ...
- Has a single read/write tape
- Can look anywhere on that tape
- Has just one accepting state,
- Namely, the halt state.

Turing Machine Example

- Recognizer for $\{ a^n b^n c^n \}$
- Acceptable starting tape: $\square a^n b^n c^n \square$
- While it's possible
 - Go from \square to \square
 - Changing one a , b and c each to X
 - Go back to left \square

Universal TM (UTM)

- **Dilemma**
 - Each TM is rigid,
 - but software makes computers flexible.
- **Solution: 2 TMs**
 - Software as M_1 encoded on tape
 - Input is also on tape
 - M_2 is the TM
 - M_2 executes M_1 on the data
 - And (on tape) keeps track of M_1 's state

Strength of TMs:

Church-Turing Thesis

- For any algorithm, there is an equivalent TM or...
- If a problem has an algorithm, it has a TM or ...
- TMs characterize computability

Support for the
Church-Turing Thesis

Part 1: Its Nature

- **Unrestricted Grammar**
 - **TM equivalent to UG...**
 - **... which exceeds CFG**
- **Universal TM**
 - **incorporates idea of software**
 - **flexible**

Support for the
Church-Turing Thesis
Part 2: Problems Solved

- Languages beyond CFLs
- Mathematical operators
- Everything else tried so far
- Including deliberate complexity

Support for the
Church-Turing Thesis
Part 3: Basic TM can simulate...

- **big alphabets**
- **2-way infinite tape**
- **multiple tapes**
- **array storage**
- **random access (jumps)**
- **nondeterministic TM**

Limits on TMs

- Recursive language: one for which some TM always halts or blocks.
- Recursively enumerable language: one for which a TM exists but may compute forever on a bad string.
- For some languages, there is no TM.

Using Uncomputability

- If TMs are as powerful as any algorithm
- ... but can't do some things,
- Then some things we can't do
- ... and should try (only) what is possible
- ... like an approximate solution.