Suppose tape has a hard left end. We want to move "caterpillar" on tape.

Suppose \( \Sigma = \{\text{c}, \text{i}, \text{d}\} \)

Looking at \( X_i \), see read or "start again."

Theorem (alluded to but not formally stated in text): For every TM \( M \) with a 2-way infinite tape, we can build a TM \( M' \) that has only a one-way infinite tape that simulates \( M \) step-for-steps.

Let: \( M' \) sees \( \lambda \) exactly when \( M \) moves into a previously unvisited blank cell on the left.

\( M' \) then overlays, at every state of \( M \), a modified caterpillar routine that is triggered by \( \lambda \) and inserts \(-\)....
Theorem (in 5.3.2): Given any K-tape TM $M$, we can build a single-tape TM $M'$ that simulates $M$ step-for-step.

$M'$ allocates a new symbol, $\#$, not used by $M$. If needed or helpful, we can have $M'$ allocate $K$ many symbols $\#, \#_2, \ldots, \#_K$ instead.

$M'$ has blank content $\theta_j$ at each tape $j$ between $\#_{j-1}$ and $\#_j$.

Now consider any configuration of a computation by $M$.

$M'$ also has a "dot alias" $C$ for every symbol used by $M$ to mark head locations on other tapes.

You can see the "caterpillar daemon" activate leftward.
Multiply number by 10, 100, etc. Use Tills can do:

* Copy strings — shift bytes if needed.
  * Maintain virtual registry on the tape or files.

Add/Subtract binary numbers, also compares.

Example:

```
L. 
M 
divide M
```

End the corresponding instruction.
Significance: Turing Machines are a General Universal Model.

Define: A language \( L \subseteq \Sigma^* \) is \((\underline{\text{Turing}}-)\) acceptable if there is a TM \( M \) s.t. \( L = L(M) \).

If in addition \( M \) halts for all inputs, then \( L \) is \underline{decidable}.

Example: \( \{ a^n b^n c^n : n \geq 0 \} \) is decidable but not a CFL. Because a DFA or a PDA "Is-A" TM that always halts, all regular languages, in deed all CFLs, are decidable too.

* These definitions have the same effect for programs in any other (known) High-level Language in place of Turing Machines.

Church-Turing Thesis: This will remain true for any model we can build, or any formulation of human or alien decision making.

Footnote: From now on, given any clear procedure-in pseudo code related to a high-level language or just in clear steps-we can assert that a Turing Machine can do it. This is how "appeals to the C-T Thesis" work in practice. For a simplifying example, can you imagine precisely how to simulate a Nondeterministic TM \( N \) by a high-level program that exhaustively tries all options available to \( N \)? Well, by C-T Thesis that program can in so-fact be converted into a Deterministic TM \( M \) such that \( L(M) = L(N) \).