Defn: A configuration \((C,D)\) of a TM \(M\)

- the current state \(q\) of \(M\)
- the contents \(W\) of all tapes together
- the head position on each tape.

For one-tape TMs it is convenient to encode \(C,D\) by strings
over the alphabet \(\Gamma' = \Gamma \cup \{\texttt{U}\}\) \((M = (Q, \Sigma, \Gamma, \delta, q_0, q_{\text{acc}}, q_{\text{rej}}) \subseteq \Gamma^*\)

\(D \text{ U C V} \) means:
- \(M\) is in state \(q\), scanning char \(C\)
- \(U\) includes all non-blank cells to the left, \(V\) to the right.
- The initial \(D\) or \(M\) on an input \(x \in \Sigma^*\), \(\emptyset\) (scanning first bit \(x\)).

\[ I_0(x) = \text{ U} x \] \(\text{ or } \Lambda S x \text{ } \) using "UNIX convention"

What if \( x = \varepsilon \)? \( I_0(x) = \text{ U} B \) or just \(S\) or \( \Lambda S \)\text{ } no ambiguity.

\( \star \) Code \( M \) so write \( B \)
- only at left or right end.
- \( U \text{ and } V \) have no Bs
- Use \( \text{ U} \) as alias \( - B \) otherwise
- \( \text{ U} \text{ F C = B} \) then \( U = \varepsilon \text{ or } V = \varepsilon \).

Say we read all \( \text{ C x } \), going to \( B \) OK to be scanning \( B \) temporarily at one end to "move an endmarker" to make more room.

\( I = \text{ U} x \text{ } B \) \( \Rightarrow I' = \text{ U} x \text{ } C \text{ R} \text{ B} \) \( \Rightarrow I'' = \text{ U} x \text{ } C \) \( \text{ R} \text{ B} \text{ } \) \( \text{ R} \text{ C} \) \( \text{ B} \text{ } \)

\( \text{ Defn } I = I' \text{ if } BB \neq J \) \( \text{ and } J \text{ no, left move} \) \( \text{ made } J \text{ once again scan } C \).

Can follow from I by executing one instruction of \( M \).
Definition: A computation $s_0, s_1, \ldots, s_t$ of $M$ on an input $x \in \Sigma^*$ is a sequence $(s_0, s_1, \ldots, s_t)$ of states such that:

- $s_0$ is the starting state on input $x$. (Whatever encoding you use for $\Gamma$-tape this)
- for all $i$, $1 \leq i \leq t$, $s_i \rightarrow s_{i+1}$ (text says this in prose)
- $s_t$ is a halting state, meaning: $(\text{general}) \quad s_t \xrightarrow{\Gamma} s_{t+1}$

The computation accepts if $s_t$ has $\text{acc}$. $(\text{by conv.})$ It has state $\text{acc}$ or $\text{rej}$.

Similarly to grammar derivations: define $s \vdash_M s'$ for any $s, s' \in \Sigma^*$,

- $s \vdash_M s'$ if there is some RD $s \vdash s' \vdash s''$.
- $s \vdash^*_M s'$ if for some $k \geq 0$, $s \vdash^k s'$.

Definition: $L(M) = \{ x \in \Sigma^* : s_0 \xrightarrow{\vdash} s_t \text{ for some accepting RD } s_t \}$

Some "good housekeeping" features:
- If $M$ writes no internal blanks, then when it "wants to" go to $\text{acc}$, it can execute a routine that erases the entire tape except a single $1$, leading to the unique accepting RD $s_t = \text{acc}$. 1.
- Likewise, a halting rejecting computation can only end in $s_t = \text{rej}$. 0.

Forward reference: In any accepting RD $u \vdash v$, we can let $v'$ be the maximum prefix of $v$ having only $|u|$ only, and let $|v|$, $\text{output} = C u'$.
- $u \vdash v$, output $\in \Sigma$. 2. If $y'$ is the intended output, can arrange $s_t = \text{acc}$.
Key Definitions:

- A language \( A \subseteq \Sigma^* \) is Turing recognizable (text) if there is a Turing machine \( M \) such that \( A = L(M) \).
- A language \( A \subseteq \Sigma^* \) is Turing acceptable (standard) if \( \exists M \) s.t. \( A = L(M) \). (equivalent)
- A language \( A \subseteq \Sigma^* \) is computably enumerable (c.e.) if \( \exists M \) s.t. \( A = L(M) \). (equivalent)
- A language \( A \subseteq \Sigma^* \) is recursively enumerable (r.e.) if \( \exists M \) s.t. \( A = L(M) \). (equivalent)
- A is decidable if in addition for all \( x \), \( I_0(x) \Downarrow \) (or \( I_0(x) \) halts) if \( x \in A \), and \( I_0(x) \) does not halt if \( x \notin A \).

Equivalently, say \( M(x) \Downarrow \) (or \( M(x) \) halts) if \( \exists x \in \Sigma^* \) and \( M \) is total in \( \forall x \) \( M(x) \).

Then \( A \) is decidable \( \iff \) \( A = L(M) \) for some total \( \forall x \) \( M(x) \).

Synonym: Recursive. Furthermore, suppose that for all \( x \in \Sigma^* \):

\[
\begin{align*}
& x \in A \iff M(x) \text{ halts and outputs } 1. \\
& x \notin A \iff M(x) \text{ halts and outputs } 0.
\end{align*}
\]

Then \( A \) is decidable.

Then as a transducer, \( M \) computes the total characteristic function \( \chi_A \) such that \( \chi_A(x) = \begin{cases} 1 & \text{if } x \in A \\ 0 & \text{if } x \notin A \end{cases} \)

which for Myhill-Nerode we just write \( A(x) \), i.e., possibly \( M(x) \) halts and equals \( 1 \).

Note: If we only have \( \forall x : x \in A \iff M(x) \Downarrow = 1 \), i.e., possibly \( M(x) \) halts and equals \( 1 \), then \( A \) is Turing recognizable.

A might be decidable by another machine \( M' \), or not [Thu. \( \rightarrow \) not!]

by default total, \( \text{dom}(f) = \Sigma^* \).

Definition: A function \( f : \Sigma^* \to \Sigma^* \) is (total) computable if there is a total \( \forall x \in \Sigma^* \) such that computes \( f : \forall x \in \Sigma^* : I_0(x) \Downarrow \) \( \Downarrow f(x) \).

Domain

If \( f \) is a partial function and \( \forall x \in \text{dom}(f) : M(x) \Downarrow = f(x) \), then \( f \) is partial computable. A is decidable \( \iff \) \( A(x) \) is total computable (with \( \text{dom}(A) \)). A is c.e. \( \iff \) \( A(x) \) is partial computable.
Rest of Ch. 3: TMs and Programs. Moreover they compute the same function.

Theorem: For every K-tape TM $M = (Q, \Sigma, \Gamma, S, B, \delta, F)$ we can design a 1-tape TM $M' = (Q', \Sigma, \Gamma', S', B, s', F')$ so $L(M) = L(M')$.

Diagram:

Use $\bullet$ as an extra char marker for the current location of each head.

Hence $\Gamma' = \Gamma \cup (\Gamma \cup \{\bullet\})^K$ where $\Gamma'$ is a dotted copy of $\Gamma$.

In this manner, even 2D $I$ of $M$ is "replicated" by an 1D $I'$ of $M'$.

If $I \subseteq J$, then $M'$ can build the corresponding $J'$ from $I'$, but it usually will require one L-to-R-to-L pass between $\land$ and $\lor$.

Thus $M'$ simulates $M$. (i.e., $M'$ is operationally equivalent to $M$.)

Initially $s = 0$, always $s \leq \sigma$.

Note: If $s$ is the current/approximate distance from $\land$ to $\lor$,
then each pass requires up to 4s steps (can do in $2s^2$).
Hence if it takes $M$ $t$ steps to halt, $M'$ halts in $O(s^3)$ steps.

Especially when $t = O(n)$, $M'$ can take $O(n^2)$ time.

At the end we will care about the time and space to solve problems.
For now note: Your 2-tape machines typically take $O(n)$ time, $O(n^2)$ 1-tape.
Example (writing end of lecture again on clean sheet) (expanded a bit)

$L = \{a^n b^n c^n : n \geq 1 \}$. Recall our one-type TM first verified that the input $x$ belongs to $a^+ b^+ c^+$. It then maintains the invariant that the tape always belongs to $\wedge X^* a^* X^* b^* X^* c^*$, e.g.,

$\wedge \underbrace{XXXaaaxXXbbbXXccc} \$ \quad \text{with } n = 5$

Here the distance between $\wedge$ and $\$" stays at $3n+1$, fixing the space as $s = 3n+1$. Each pass uses $2s$ steps. The TM does $n$ passes. Thus total time $t = n(6n+2) = \Theta(n^2)$.

A two-tape TM $M_2$ can execute a simpler & quicker strategy in $O(n)$ time:

If you convert $M_2$ into a one-type TM $M_1$, the time becomes $\Theta(ns) = \Theta(3n^2n) = \Theta(n^3)$

*"Polynomial time" ignores quadratic time loss, so 1-type and K-types are equivalent for it.*

The "Universal RAM Simulator" handout sketches how this equivalence extends to a "mini-assembly" language, and then to time in real programming languages a-la CSE331 "Church-Turing thesis."
Two two-tape Turing Machines, the first a DPDA, the second not

A DPDA that recognizes the language of balanced paren strings. Convention: start with heads on blank to left of input.

\[ \Sigma: \{\} \]
\[ \Gamma: \{\} \land \text{blank} \]

\[ \text{Input Alphabet: } \{0, 1\} \]
\[ \text{Work Alphabet: } \{0, 1, \land, \$, \text{blank}\} \]
Initially, write input \( x \) on Tape 1, with head on its first bit.
\[ L(M) = \{xx : x \in (0, 1)^*\} \]

x = the empty string—accept.

\[ |x| \text{ is even, } x = vv, \text{ copy } v \text{ to Tape 2} \]

Write \$ marker on Tape 2 to use as "goal" for "w = v".

Note—lead bit of v overwrites the \land marker.

Success—all chars match.

Mismatch i.e., w \neq v

Homer-Selman
Exercise 1.1
CS 396 Spring 2016: To Skim for G3.2 of text

& 396 Universal RAM Simulator

Fall 2014

Program Syntax: INSTR = (CR) DIG, ALPH, DIG

Register Syntax: REG = [01G#(-1)] DIG

ALU Syntax: ALU = (-2) DIG

All tape set as: # = # delimiters

Initial: ... !INS1; INS2; ... INSm; # X1, X2, ..., Xn ...

Input convention: Start P(i) with X in ALU

Output: Last instruction LOR 1; putting P(x) in ALU

Start

Set tape labels (just for "show"

and AS)

Move X to

ALU and

lay $s

Move all heads Left to AS

Tape 1 head now scans "1"

Find Label

From ALU

Copy Label

From ALU

HALT

Blank out

rest of ALU

(CR)

Copy n From

Tape 2 to Tape 3

Decide Instruction

Find Register

From Program

Copy Register

Value Over ALU

Copy Register

Value Above ALU

Copy ALU

Into Register

Find Register

From ALU

Find Register

From Program

Add into ALU

Allocate New

Reg. From ALU

Also see "Collate into Machine Overleaf"
"3n+1 Binary"
By K.W. Regan
Illustrates the "Collatz Problem."

Start with a binary number n on the tape, and position the head anywhere on the left.

Delete trailing 0s (i.e., divide by 2)

Div by 2

Test n == 1
If so, then accept.

Test for single '1'

Carry 0

Carry 1

Carry 2

Move head to rightmost bit.

Multiply n by 3 and add 1

Erase any leading 0s in n, skip over leading blanks.