The Kleene Star Operation $*^a$

- Let $A$ be a set.

- The **Kleene star** of $A$, denoted by $A^*$, is the set of all strings obtained by concatenating zero or more strings from $A$.
  - For example, suppose $A = \{0, 1\}$.
  - Then
    
    $$A^* = \{ \varepsilon, 0, 1, 00, 01, 10, 11, 000, \ldots \}.$$  

  - Note that every string in $A^*$ must be of finite length.

\(^{a}\text{Kleene (1956).}\)
Decidability and Recursive Languages

- Let $L \subseteq (\Sigma - \{\square\})^*$ be a language, i.e., a set of strings of non-$\square$ symbols, with a finite length.
  - For example, \{0, 01, 10, 210, 1010, \ldots\}.

- Let $M$ be a TM such that for any string $x$:
  - If $x \in L$, then $M(x) =$ “yes.”
  - If $x \notin L$, then $M(x) =$ “no.”

- We say $M$ decides $L$.

- If there exists a TM that decides $L$, then $L$ is recursive\(^a\) or decidable.

\(^a\)Little to do with the concept of “recursive” calls.
Recursive and Nonrecursive Languages: Examples

- The set of palindromes over any alphabet is recursive.\(^a\)
- The set of prime numbers \(\{2, 3, 5, 7, 11, 13, 17, \ldots\}\) is recursive.\(^b\)
- The set of C programs that do not contain a \texttt{while}, a \texttt{for}, or a \texttt{goto} is recursive.\(^c\)
- But, the set of C programs that do not contain an infinite loop is \textit{not} recursive (see p. 155).

\(^a\)Need a program that returns “yes” iff the input is a palindrome.
\(^b\)Need a program that returns “yes” iff the input is a prime.
\(^c\)Need a program that returns “yes” iff the input C code does not contain any of the keywords.
Acceptability and Recursively Enumerable Languages

- Let $L \subseteq (\Sigma - \{\uparrow\})^*$ be a language.

- Let $M$ be a TM such that for any string $x$:
  - If $x \in L$, then $M(x) =$ “yes.”
  - If $x \notin L$, then $M(x) = \uparrow$.\(^a\)

- We say $M$ accepts $L$.

- How to verify that a TM decides/accepts a language is a different matter.\(^b\)

\(^a\)This part is different from recursive languages.
\(^b\)Thanks to a lively discussion on September 23, 2014.
Acceptability and Recursively Enumerable Languages (concluded)

- If $L$ is accepted by some TM, then $L$ is called a **recursively enumerable language**.\(^a\)
  - A recursively enumerable language can be *generated* by a TM, thus the name.\(^b\)
  - That is, there is an algorithm such that for every $x \in L$, it will be printed out eventually.
  - If $L$ is infinite in size, this algorithm will not terminate.

\(^a\)Post (1944).
\(^b\)Thanks to a lively class discussion on September 20, 2011.
Emil Post (1897–1954)
Recursive and Recursively Enumerable Languages

**Proposition 2** If $L$ is recursive, then it is recursively enumerable.

- Let TM $M$ decide $L$.
- Need to design a TM that accepts $L$.
- We will modify $M$ to obtain an $M'$ that accepts $L$. 
The Proof (concluded)

- $M'$ is identical to $M$ except that when $M$ is about to halt with a “no” state, $M'$ goes into an infinite loop.
  - Simply replace any instruction that results in a “no” state with ones that move the cursor to the right forever and never halts.

- $M'$ accepts $L$.
  - If $x \in L$, then $M'(x) = M(x) = “yes.”$
  - If $x \not\in L$, then $M(x) = “no”$ and so $M'(x) = \uparrow$. 
Recursively Enumerable Languages: Examples

- The set of C program-input pairs that do not run into an infinite loop is recursively enumerable.
  - Just run its binary code in a simulator environment.
  - Then the simulator will terminate if and only if the C program will terminate.
  - When the C program terminates, the simulator simply exits with a “yes” state.

- The set of C programs that contain an infinite loop is \textit{not} recursively enumerable (see p. 155).
Turing-Computable Functions

- Let $f : (\Sigma - \{\sqcup\})^* \to \Sigma^*$.
  - Optimization problems, root finding problems, etc.
- Let $M$ be a TM with alphabet $\Sigma$.
- $M$ computes $f$ if for any string $x \in (\Sigma - \{\sqcup\})^*$, $M(x) = f(x)$.
- We call $f$ a recursive function\(^a\) if such an $M$ exists.

\(^a\)Kurt Gödel (1931, 1934).
Kurt Gödel\(^a\) (1906–1978)

Quine (1978), “this theorem […] sealed his immortality.”

\(^a\)This photo was taken by Alfred Eisenstaedt (1898–1995).
Church’s Thesis or the Church-Turing Thesis

• What is computable is Turing-computable; TMs are algorithms.\(^a\)

• No “intuitively computable” problems have been shown not to be Turing-computable, yet.\(^b\)

\(^a\)Church (1936); Kleene (1953).
\(^b\)Quantum computer of Manin (1980) and Feynman (1982) and DNA computer of Adleman (1994).
Church’s Thesis or the Church-Turing Thesis (concluded)

- Many other computation models have been proposed.
  - Recursive function (Gödel), \( \lambda \) calculus (Church),
    formal language (Post), assembly language-like RAM
    (Shepherdson & Sturgis), boolean circuits (Shannon),
    extensions of the Turing machine (more strings,
    two-dimensional strings, and so on), etc.

- All have been proved to be equivalent.
Alonso Church (1903–1995)
Stephen Kleene (1909–1994)
Extended Church’s Thesis

- All “reasonably succinct encodings” of problems are \textit{polynomially related} (e.g., \( n^2 \) vs. \( n^6 \)).
  - Representations of a graph as an adjacency matrix and as a linked list are both succinct.
  - The \textit{unary} representation of numbers is not succinct.
  - The \textit{binary} representation of numbers is succinct.
    \* \( 1001_2 \) vs. \( 11111111_1 \).

- All numbers for TMs will be binary from now on.

\textsuperscript{a}Some call it “polynomial Church’s thesis,” which Lószló Lovász attributed to Leonid Levin.
Extended Church’s Thesis (concluded)

- Representations that are not succinct may give misleadingly low complexities.
  - Consider an algorithm with binary inputs that runs in $2^n$ steps.
  - Suppose the input uses unary representation instead.
  - Then the same algorithm runs in linear time because the input length is now $2^n$!
- So a succinct representation is for honest accounting.
Physical Church-Turing Thesis

• “[Church’s thesis] is a profound claim about the physical laws of our universe, i.e.: any physical system that purports to be a ‘computer’ is not capable of any computational task that a Turing machine is incapable of.”

• “Anything computable in physics can also be computed on a Turing machine.”

• The universe is a Turing machine.

---

\(^a\) Warren Smith (1998).  
\(^b\) Cooper (2012).  
\(^c\) Edward Fredkin’s (1992) digital physics.
The Strong Turing-Church Thesis\textsuperscript{a}

- The strong Turing-Church Thesis states that:
  A Turing machine can compute any function computable by any “reasonable” physical device with only polynomial slowdown.

- A CPU and a DSP chip are good examples of physical devices.\textsuperscript{b}

\textsuperscript{a}Vergis, Steiglitz, and Dickinson (1986).
\textsuperscript{b}Thanks to a lively discussion on September 23, 2014.
Turing Machines with Multiple Strings

- A $k$-string Turing machine (TM) is a quadruple $M = (K, \Sigma, \delta, s)$.
- $K, \Sigma, s$ are as before.
- $\delta : K \times \Sigma^k \to (K \cup \{h, \text{“yes”}, \text{“no”}\} \times (\Sigma \times \{\leftarrow, \rightarrow, -\})^k$.
- All strings start with a $\triangleright$.
- The first string contains the input.
- Decidability and acceptability are the same as before.
- When TMs compute functions, the output is the last ($k$th) string.
A 2-String TM

\[ \delta \]

\[ \rightarrow 1000110000111001110001110 \]

\[ \rightarrow 111110000 \]

\[ \rightarrow 111110000 \]
PALINDROME Revisited

- A 2-string TM can decide PALINDROME in $O(n)$ steps.
  - It copies the input to the second string.
  - The cursor of the first string is positioned at the first symbol of the input.
  - The cursor of the second string is positioned at the last symbol of the input.
  - The symbols under the cursors are then compared.
  - The two cursors are then moved in opposite directions until the ends are reached.
  - The machine accepts if and only if the symbols under the two cursors are identical at all steps.
\[ \delta \]

\[ \Rightarrow \text{ababbaabbaabbbaba} \]

\[ \Rightarrow \text{ababbaabbaabbaabbbaba} \]

\[ \Rightarrow \text{ababbaabbaabbaabbbaba} \]
PALINDROME Revisited (concluded)

- The running times of a 2-string TM and a single-string TM are quadratically related: $n^2$ vs. $n$.
- This is consistent with extended Church’s thesis.
Configurations and Yielding

- The concept of configuration and yielding is the same as before except that a configuration is a (2k + 1)-tuple

\[(q, w_1, u_1, w_2, u_2, \ldots, w_k, u_k).\]

- \(w_i u_i\) is the \(i\)th string.
- The \(i\)th cursor is reading the last symbol of \(w_i\).
- Recall that \(\triangleright\) is each \(w_i\)’s first symbol.

- The \(k\)-string TM’s initial configuration is

\[
(s, \triangleright, x, \triangleright, \epsilon, \triangleright, \epsilon, \ldots, \triangleright, \epsilon).
\]
Time seemed to be the most obvious measure of complexity.

— Stephen Arthur Cook (1939—)
Time Complexity

- The multistring TM is the basis of our notion of the time expended by TMs.
- If a $k$-string TM $M$ halts after $t$ steps on input $x$, then the time required by $M$ on input $x$ is $t$.
- If $M(x) = \nearrow$, then the time required by $M$ on $x$ is $\infty$. 
Time Complexity (concluded)

- Machine $M$ operates within time $f(n)$ for $f : \mathbb{N} \rightarrow \mathbb{N}$ if for any input string $x$, the time required by $M$ on $x$ is at most $f(|x|)$.
  - $|x|$ is the length of string $x$.

- Function $f(n)$ is a time bound for $M$. 
Time Complexity Classes\textsuperscript{a}

- Suppose language $L \subseteq (\Sigma - \{\|\})^*$ is decided by a multistring TM operating in time $f(n)$.
- We say $L \in \text{TIME}(f(n))$.
- $\text{TIME}(f(n))$ is the set of languages decided by TMs with multiple strings operating within time bound $f(n)$.
- $\text{TIME}(f(n))$ is a complexity class.
  - PALINDROME is in $\text{TIME}(f(n))$, where $f(n) = O(n)$.

\textsuperscript{a}Hartmanis and Stearns (1965); Hartmanis, Lewis, and Stearns (1965).
Juris Hartmanis\textsuperscript{a} (1928–)

\textsuperscript{a}Turing Award (1993).
Richard Edwin Stearns\textsuperscript{a} (1936–)

\textsuperscript{a}Turing Award (1993).
The Simulation Technique

**Theorem 3** Given any $k$-string $M$ operating within time $f(n)$, there exists a (single-string) $M'$ operating within time $O(f(n)^2)$ such that $M(x) = M'(x)$ for any input $x$.

- The single string of $M'$ implements the $k$ strings of $M$. 
The Proof

- Represent configuration \((q, w_1, u_1, w_2, u_2, \ldots, w_k, u_k)\) of \(M\) by this string of \(M'\):

\[
(q, \triangleright w'_1 u_1 \triangleleft w'_2 u_2 \triangleleft \cdots \triangleleft w'_k u_k \triangleleft \triangleleft).
\]

- \(\triangleleft\) is a special delimiter.
- \(w'_i\) is \(w_i\) with the first\(^a\) and last symbols “primed.”
- It serves the purpose of “,” in a configuration.\(^b\)

\(^a\)The first symbol is always \(\triangleright\).

\(^b\)An alternative is to use \((q, \triangleright w'_1 u_1, \triangleleft w'_2, u_2 \triangleleft \cdots \triangleleft w'_k, u_k \triangleleft \triangleleft)\) by priming only \(\triangleright\) in \(w_i\), where “,,” is a new symbol.
The Proof (continued)

- The “priming” of the last symbol of $w_i$ ensures that $M'$ knows which symbol is under each cursor of $M$.\(^a\)

- The first symbol of $w_i$ is the primed version of $\rhd$: $\rhd'$.  
  - Recall TM cursors are not allowed to move to the left of $\rhd$ (p. 21).
  - Now the cursor of $M'$ can move between the simulated strings of $M$\(^b\).

\(^{\text{a}}\) Added because of comments made by Mr. Che-Wei Chang (R95922093) on September 27, 2006.

\(^{\text{b}}\) Thanks to a lively discussion on September 22, 2009.
The Proof (continued)

- The initial configuration of $M'$ is

\[
(s, \triangleright \triangleright'' x \triangleleft \triangleright'' \triangleleft \cdots \triangleright'' \triangleleft \triangleright). \]

- $\triangleright$ is double-primed because it is the beginning and the ending symbol as the cursor is reading it.\(^a\)

\(^a\)Added after the class discussion on September 20, 2011.
The Proof (continued)

- We simulate each move of $M$ thus:
  1. $M'$ scans the string to pick up the $k$ symbols under the cursors.
     - The states of $M'$ must be enlarged to include $K \times \Sigma^k$ to remember them.\(^a\)
     - The transition functions of $M'$ must also reflect it.
  2. $M'$ then changes the string to reflect the overwriting of symbols and cursor movements of $M$.

\(^a\)Recall the TM program on p. 27.
The Proof (continued)

- It is possible that some strings of $M$ need to be lengthened (see next page).
  - The linear-time algorithm on p. 40 can be used for each such string.

- The simulation continues until $M$ halts.

- $M'$ then erases all strings of $M$ except the last one.$^a$

\hspace{1in}\footnote{$^a$Because whatever appears on the string of $M'$ will be considered the output. So those $\triangleright$'s and $\triangleright''$s need to be removed.}
The Proof (continued)

- Since $M$ halts within time $f(|x|)$, none of its strings ever becomes longer than $f(|x|)$.

- The length of the string of $M'$ at any time is $O(kf(|x|))$.

- Simulating each step of $M$ takes, per string of $M$, $O(kf(|x|))$ steps.
  - $O(f(|x|))$ steps to collect information from this string.
  - $O(kf(|x|))$ steps to write and, if needed, to lengthen the string.

---

\textsuperscript{a}We tacitly assume $f(n) \geq n$. 
The Proof (concluded)

- $M'$ takes $O(k^2 f(|x|))$ steps to simulate each step of $M$ because there are $k$ strings.

- As there are $f(|x|)$ steps of $M$ to simulate, $M'$ operates within time $O(k^2 f(|x|)^2)$. 
Linear Speedup\textsuperscript{a}

**Theorem 4** Let $L \in \text{TIME}(f(n))$. Then for any $\epsilon > 0$, $L \in \text{TIME}(f'(n))$, where $f'(n) = \epsilon f(n) + n + 2$.

\textsuperscript{a}Hartmanis and Stearns (1965).
Implications of the Speedup Theorem

- State size can be traded for speed.\(^a\)

- If \( f(n) = cn \) with \( c > 1 \), then \( c \) can be made arbitrarily close to 1.

- If \( f(n) \) is superlinear, say \( f(n) = 14n^2 + 31n \), then the constant in the leading term (14 in this example) can be made arbitrarily small.
  - *Arbitrary* linear speedup can be achieved.\(^b\)
  - This justifies the big-O notation in the analysis of algorithms.

\(^a\) \( m^k \cdot |\Sigma|^{3mk} \)-fold increase to gain a speedup of \( O(m) \). No free lunch.

\(^b\) Can you apply the theorem multiple times to achieve superlinear speedup? Thanks to a question by a student on September 21, 2010.
By the linear speedup theorem, any polynomial time bound can be represented by its leading term $n^k$ for some $k \geq 1$.

- If $L$ is a **polynomially decidable language**, it is in $\text{TIME}(n^k)$ for some $k \in \mathbb{N}$.
  - Clearly, $\text{TIME}(n^k) \subseteq \text{TIME}(n^{k+1})$.

- The union of all polynomially decidable languages is denoted by $\mathsf{P}$:
  
  $$\mathsf{P} = \bigcup_{k>0} \text{TIME}(n^k).$$

- $\mathsf{P}$ contains problems that can be efficiently solved.
Philosophers have explained space. They have not explained time.

— Arnold Bennett (1867–1931), *How To Live on 24 Hours a Day* (1910)

I keep bumping into that silly quotation attributed to me that says 640K of memory is enough.

— Bill Gates (1996)