Learning Objectives

- Principles of Turing Machine
- Evolution of Calculators and Computers
- Programming
- Crisp Sets
  - Properties of collections of objects
- Fuzzy Sets and Control Systems
  - Effects of uncertainty or imprecision

Turing Machine
(Logical Computing Machine)

- Abstract representation of programming for a computing device
  - Attempt to give mathematically precise definition to algorithm or mechanical (or effective) procedure
  - Hardware description as a machine is figurative
- Finite number of internal discrete states of the machine
  - States = steps or instructions of a program
- Unlimited amount of external input data on a tape
Elements of a Logical Computing Machine (LCM)

- **Data Tape**, with bits in bins
  
  ![Data Tape Diagram]

- **Read/Write Register**
  - Senses “1” or “0”
  - Assigns “1” or “0”

- **Set of Instructions (Program) for R/W Register**
  - Define **internal states**, which are identified by state (i.e., program line) number

- **Control Mechanism**
  - Move read/write head over tape
  - Or move tape through read/write head

Operation of a Turing Machine

**Execution begins** in field of “0”s to the left, with device at **State #0**

![Operation Diagram]

- **State #0 instruction** moves device to the right until it encounters a “1” in bin (i.e., on the tape)
- **Action of device** depends on current instruction in a Stored Program (i.e., **Set of Instructions**)
  - Modifies the bin (or not)
  - Identifies next internal state
  - Moves one bin to right or left
Example of a Turing Machine:
Denary (Base 10) Notation for State (Line Number)

- Execution begins in field of “0”s to the left, with device at State #0
- State #0 instruction moves device to the right until it encounters a “1”
- Action of device depends on stored instructions
  - Modifies the bin (or not)
  - Identifies next internal state
  - Moves one bin to right or left

<table>
<thead>
<tr>
<th>[Program] Instruction State #</th>
<th>[Data] Register Contents Next State</th>
<th>New Bin Contents</th>
<th>Direction of Move</th>
</tr>
</thead>
<tbody>
<tr>
<td>0 0 0 0 R</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0 1 13 1 L</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1 0 65 1 R</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1 1 1 0 R</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2 0 0 1 R (Stop)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2 1 66 1 L</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3 0 37 1 L</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>... ... ... ...</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>210 0 3 1 L</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>... ... ... ...</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>258 1 0 0 R (Stop)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>259 0 97 1 R</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>259 1 0 0 R (Stop)</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Penrose, 1989

Same Example:
Binary-coded Program Line Number and Move Direction

- Execution begins in field of “0”s to the left, with device at State #0
- State #0 instruction moves device to the right until it encounters a “1”
- Action of device depends on stored instructions
  - Modifies the bin (or not)
  - Identifies next internal state
  - Moves one bin to right or left

<table>
<thead>
<tr>
<th>[Program] Instruction State #</th>
<th>[Data] Register Contents Next State</th>
<th>New Bin Contents</th>
<th>Direction of Move</th>
</tr>
</thead>
<tbody>
<tr>
<td>0 0 0 1</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0 1 1101 1</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1 0 100001 1</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1 1 1 0 1</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>10 0 0 1 (Stop)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>10 1 1000010 1</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>11 0 100101 1</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>... ... ... ...</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>11010010 0 11 1 0</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Unary, Binary, and Expanded Binary Coding of Data

- Turing Machine to add “1” to a unary number (at right)
  - On data tape: 0111

- Binary coding problems
  - Notation for terminating binary description of a number
  - Definition of space between numbers
  - Recognition of de-limiters (e.g., commas) and logical/arithmetic operators

- Data could be stored in unary format
  - On data tape: 01110

Unary and Expanded Binary Coding of Data

- Solution: Expanded Binary Coding
  - Contraction: Unary expression of small numbers, separated by “0”
  - Encode de-limiters and operators as numbers
  - Terminate numbers with commas (as delimiter)

<table>
<thead>
<tr>
<th>Number</th>
<th>Meaning</th>
<th>Expanded Binary Notation</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Binary &quot;0&quot;</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>Binary &quot;1&quot;</td>
<td>10</td>
</tr>
<tr>
<td>2</td>
<td>Comma</td>
<td>110</td>
</tr>
<tr>
<td>3</td>
<td>Minus Sign</td>
<td>1110</td>
</tr>
<tr>
<td>4</td>
<td>Plus Sign</td>
<td>11110</td>
</tr>
</tbody>
</table>
 Expanded Binary Coding

<table>
<thead>
<tr>
<th>Number (code)</th>
<th>Meaning (de-code)</th>
<th>Expanded Binary Code: 0 1 0 0 0 1 0 1 1 0 1 0 1 0 1 1 0 1 1 0 1 0 0</th>
</tr>
</thead>
<tbody>
<tr>
<td>1001</td>
<td>9, 3</td>
<td>1101 1 1 2 1 1 2 1 0 0 3 1 1</td>
</tr>
</tbody>
</table>
| 1110         | 3                | 4                                                                | 3

<table>
<thead>
<tr>
<th>Number</th>
<th>Meaning</th>
<th>Expanded Binary Notation</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Binary &quot;0&quot;</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>Binary &quot;1&quot;</td>
<td>10</td>
</tr>
<tr>
<td>2</td>
<td>Comma</td>
<td>110</td>
</tr>
<tr>
<td>3</td>
<td>Minus Sign</td>
<td>1110</td>
</tr>
<tr>
<td>4</td>
<td>Plus Sign</td>
<td>11110</td>
</tr>
</tbody>
</table>

Expanded Binary Coding and Turing Machines

- Expression of arbitrary denary statement numbers
  - Convert from denary to binary
  - Convert from binary to expanded binary
- Define Turing Machine for operations on expanded binary code

Original: 0 1 13 1 L
Binary: 0 1 1101 1 0
De-Limited: ,0,1,1101,1,0,
Expanded Binary: 00011011010110101001011010110110

Machine code is written in Expanded Binary (or similar) code
Further Evolution of the Turing Machine

- **Universal Turing Machine** *(Computer Program in “Machine Language”)*
  - Turing Machine for control of a Turing Machine
  - Instructions on a separate tape or at beginning of data tape
  - 2nd TM with simple code set reads register contents
  - Instruction tape could be modified just like the data tape
    - Instruction branches can be conditioned on prior results
    - Self-modifying instruction set possible

- **Church-Turing Thesis** *(“Hypothesis”, “Conjecture”)*
  - Turing Machine *(LCM)* defines what we mean by an algorithmic, mechanical, effective, or recursive procedure
  - *LCM* can do anything that could be described as a Rule of Thumb or “purely mechanical”

---

**Calculation and Computing**
Calculation

- Thinking, augmented by
  - Abacus
  - Slide rule
  - Math tables
  - Mechanical calculator

Early Computers - IBM

- 026 Key Punch
- Punch Card
- One line of code
- IBM 7094
- IBM 650 Computer
- IBM 360/91

http://en.wikipedia.org/wiki/IBM_650
http://en.wikipedia.org/wiki/IBM_7094
http://en.wikipedia.org/wiki/IBM_360
Early Computers – Punched Card

Apollo Guidance Computer

- Parallel processor
- 16-bit word length (hexadecimal)
- Memory
  - 36,864 words (fixed)
  - 2,048 words (variable)
- 1st operational solid-state computer
- Identical computers in CSM and LM
  - Different software (with many identical subroutines)

[https://www.youtube.com/watch?v=ULGi3UkgW30](https://www.youtube.com/watch?v=ULGi3UkgW30)
[http://klabs.org/history/build_aqc/](http://klabs.org/history/build_aqc/)
Hardware Architectures

Central Processing Unit (CPU)

- Arithmetic Logic Unit (ALU)
- Processor registers (~ cache memory)
- Control unit

Field-Programmable Gate Array (FPGA)

- Application-Specific Integrated Circuit (ASIC)
- Programmable logic blocks/“gates”
  - Look-up tables, flip-flops (bistable latches), and routing matrix
- Reconfigurable connections
- Data buses, timers, analog components
Hardware Architectures

Graphics Processing Unit (GPU)

- Highly parallel structure for rendering images
- Transformation, clipping, texture mapping, shading, and lighting
- Specialization to vector-matrix operations

Tensor Processing Unit (TPU)

- Application-Specific Integrated Circuit (ASIC) for machine learning
- *Google TensorFlow* symbolic math CISC s/w library
- High-volume, reduced-precision logic (e.g., 256 x 256 8-bit matrix multiply, on-chip memory and accumulators)
- Arrays of arrays of TPU chips ~ 11.5 PFLOPS performance

https://en.wikipedia.org/wiki/Tensor_processing_unit
Evolution of Programming

- History of programming languages
- The song, “99 Bottles of Beer on the Wall”, programmed in 1,500 computer languages
  - In BASIC:

```
10 REM BASIC Version of 99 Bottles of beer
20 FOR X=100 TO 1 STEP -1
30 PRINT X;"Bottle(s) of beer on the wall,";X;"bottle(s) of beer"
40 PRINT "Take one down and pass it around,"
50 PRINT X-1;"bottle(s) of beer on the wall"
60 NEXT
```
Programming Language Classes

• Expert systems can be programmed in almost any language
• Language is the interface between the programmer and the computer
• Higher-order
  – Lower-order
    • Interpreter
    • Compiler
      – Assembly language
        » Machine code
• Critical differences
  – Instruction set
  – Execution speed
  – Memory use

• Procedural (e.g., FORTRAN, LISP, MATLAB, Python)
  – Imperative
  – Functional
• Non-Procedural, Query-based Languages (e.g., PROLOG)
  – Declarative
  – Non-Declarative

Ultimately, it is all machine code (“0”s and “1”s)
Naive (or Intuitive) Set Theory (1870s)

- Deals with the properties of well-defined collections of objects
- Universal set = Universe of discourse = \( U \)
  - Contains all elements of possible concern in a particular context
- \( A \) = a particular set in \( U \)
  - defined in a list
  - by a rule, or
  - by a membership function describing elements (or members) of the set

Venn Diagrams (1881)

- A: All mammals
- B: All aquatic animals
- C: All gray, hairless objects
- A & B: Whales, dolphin, seals, ...
- B & C: Fish, clams, whales, dolphins, ...
- A, B, & C: Whales, dolphins, ...
Three Criteria for Membership in a Set

• **List**
  \[ A = \{x, y, z, \ldots\} \]
  Ordering of elements is not important

  \[ A = (x, y, z, \ldots) \]
  Ordering of elements is important

• **Rule**
  \[ A = \{ x \in U \mid x \text{ meets some conditions} \} \]

• **Membership function, e.g., categorical description**
  \[ \mu_A(x) = \begin{cases} 
  1, & \text{if } x \in A \\
  0, & \text{if } x \not\in A 
\end{cases} \]

Membership in a Set

• **A** = a particular set in **U**
  – defined in a list or rule, or
  – by a membership function describing elements (or members) of the set

• **Universal set** = guests at a party

• **Particular sets**
  – Current graduate students
  – Alumni
  – Spouses
  – Friends of students
  – Children
  – Same family
  – Visitors
  – Pilots
  – Teachers
  – Managers
  – Military officers
  – Women and men
  – US citizens or foreign nationals
Operations on Sets

- **Union of sets**
  \[ C = A \cup B \]

- **Intersection of sets**
  \[ C = A \cap B \]

- **Proper Subset**
  \[ B \subset A \]
  \[ B \subseteq A, \text{ and } B \neq A \]

- **One-to-one correspondence**
  \[ A = B \]

  For example, as when
  \[ A = (x, y, z) \]
  \[ B = (4, 3, 9); \text{ then } x = 4, y = 3, \text{ and } z = 9 \]

Properties of Sets

- **Complement**
  \[ A' = U - A \]

- **Reflexive property**
  - Relationships that bear same effect on own set as on other sets
  \[ e.g., \quad A = A, \quad A \geq A, \quad A \leq A \]

- **Transitive property**
  - Two sets bear same relationship to a third set
  \[ e.g., \text{ if } A > B \text{ and } B > C, \text{ then } A > C \]

- **Empty (null) set**
  \[ \emptyset = A - A \text{ or } U - U \]

- **Symmetry property**
  - Relationship of first to second set is the same as second to first set
  \[ e.g., A = B, B = A \]

- **Equivalence**
  - Reflexivity + Symmetry + Transitivity
  \[ A \sim B \text{ or } A \equiv B \]
What If Sets Have Uncertain or Contradictory Membership?

• Example: \( U = \text{All the cars in Berkeley} \)
  
  \[ A = \{ x \in U \mid \text{x has 4 cylinders} \} \]
  
  \[ B = \{ x \in U \mid \text{x has 6 cylinders} \} \]

• What makes a car “US” or “Foreign”?
  – US cars may contain foreign parts
  – Foreign cars may contain US parts

How should we define Sets \( E \) and \( F \)?

---

Fuzzy Sets
Hard and Soft Thinking

- Problem-solving approaches
  - Logical / Metaphorical
  - Reasonable / Dream-like
  - Serious / Humorous
  - Definite / Ambiguous
  - Consistent / Paradoxical
  - Laborious / Playful
  - Exact / Approximate
  - Real / Fantastic
  - Focused / Diffuse
  - Analytical / Illogical
  - Specific / General
  - Mature / Immature
- Crisp / Fuzzy

A Notional Fuzzy Experiment

- What does each term mean?
  - A few \( \leq x \)
  - A lot \( \geq x \)
  - Several = \( x \)

Normalize results so that the maximum is 1
Normalized plots are fuzzy membership functions
Fuzzy Sets

- Fuzzy membership function, $\mu_A(x) = 1$
  - takes any value in $[0, 1]$
- Fuzzy set, $E$
  $$E = \{ x \in U \big| \mu_E(x) \}$$
- $U = \text{All the cars in Berkeley}$
  - $p(x) = \text{percentage of domestic parts} = \mu_D(x) = 1$
  - $[1 - p(x)] = \text{percentage of foreign parts} = \mu_F(x) = 1$

- Membership functions express a subjective utility that may be rigorous (e.g., based on probability) or not
- "If 46% of a car’s parts are domestic, it probably is a foreign car"

Fuzzy Membership Functions

- "Close to zero"
- "About $y$"
- "Probably not $y$"
- "Young/old"
Fuzzy Set Definitions

- **Support of a fuzzy set**
  \[
  \text{supp}(A) = \{ x \in U \mid \mu_A(x) > 0 \}
  \]

- **Center of a fuzzy set**
  - Two-sided: center = mean value
  - Left-sided: center = \( \max \{ x \} \) for which \( \mu(x) = 1 \)
  - Right-sided: center = \( \min \{ x \} \) for which \( \mu(x) = 1 \)

Fuzzy Logic Operations

**Union of sets**

\[
A \cup B \\
\text{if and only if } \mu_{A \cup B}(x) = \max]\{\mu_A(x), \mu_B(x)\} \forall x \in U
\]

**Intersection of sets**

\[
A \cap B \\
\text{if and only if } \mu_{A \cap B}(x) = \min]\{\mu_A(x), \mu_B(x)\} \forall x \in U
\]

**A contains B**

\[
A \supseteq B \\
\text{if and only if } \mu_A(x) \geq \mu_B(x) \forall x \in U
\]
More Fuzzy Logic Operations

Equivalence of sets

\[ A \sim B \]

if and only if \( \mu_A(x) = \mu_B(x) \forall x \in U \)

Complement of sets

\[ A' = U - A \]

if and only if \( \mu_A(x) = 1 - \mu_A(x) \forall x \in U \)

Fuzzy logic is a generalization of crisp logic based on the definition of the membership function

Example: Cement Kiln Control
Linguistic (Mamdani) Fuzzy Control Systems (after Schramm)

- **Antecedent** and **consequent** are both fuzzy propositions
  - e.g., "If error is small and error rate is negative, then control command is small"
  - What are “small”, “medium”, and “large”?
- Must “fuzzify” physical error/rate, apply fuzzy rules, and “de-fuzzify” control command

### Linguistic Fuzzy Control System Table

<table>
<thead>
<tr>
<th>Case</th>
<th>Condition</th>
<th>Action to be taken</th>
<th>Reason</th>
</tr>
</thead>
<tbody>
<tr>
<td>10</td>
<td>BZ ok</td>
<td>a. Increase air fan speed</td>
<td>To raise back-end temperature and increase oxygen for action ‘b’</td>
</tr>
<tr>
<td></td>
<td>OX low</td>
<td>b. Increase fuel rate</td>
<td></td>
</tr>
<tr>
<td></td>
<td>BE low</td>
<td>a. Decrease fuel rate slightly</td>
<td>To maintain burning zone temperature</td>
</tr>
<tr>
<td></td>
<td>BE ok</td>
<td>To raise oxygen</td>
<td></td>
</tr>
<tr>
<td>11</td>
<td>BZ ok</td>
<td>a. Increase fuel rate speed slightly</td>
<td></td>
</tr>
<tr>
<td></td>
<td>OX low</td>
<td>b. Increase air fan speed</td>
<td></td>
</tr>
<tr>
<td></td>
<td>BE ok</td>
<td>To raise back-end temperature</td>
<td></td>
</tr>
<tr>
<td>12</td>
<td>BZ ok</td>
<td>a. Reduce fuel rate</td>
<td>To increase oxygen for action ‘b’</td>
</tr>
<tr>
<td></td>
<td>OX low</td>
<td>b. Reduce air fan speed</td>
<td>To lower back-end temperature and maintain burning zone temperature</td>
</tr>
<tr>
<td>13</td>
<td>BZ ok</td>
<td>a. Increase air fan speed</td>
<td>To raise back-end temperature</td>
</tr>
<tr>
<td></td>
<td>OX ok</td>
<td>b. Increase fuel rate</td>
<td>To maintain burning zone temperature</td>
</tr>
<tr>
<td></td>
<td>BE ok</td>
<td></td>
<td></td>
</tr>
<tr>
<td>14</td>
<td>BZ ok</td>
<td>None. However, do not get overconfident and keep all conditions under observation</td>
<td></td>
</tr>
<tr>
<td></td>
<td>OX ok</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>BE ok</td>
<td></td>
<td></td>
</tr>
<tr>
<td>15</td>
<td>BZ ok</td>
<td>a. Reduce air fan speed</td>
<td>To reduce back-end temperature and oxygen</td>
</tr>
<tr>
<td></td>
<td>OX ok</td>
<td>b. Reduce fuel rate</td>
<td></td>
</tr>
<tr>
<td></td>
<td>BE high</td>
<td>c. Reduce air fan speed</td>
<td>To raise oxygen for action ‘c’</td>
</tr>
<tr>
<td></td>
<td>When oxygen is in lower part of range</td>
<td></td>
<td></td>
</tr>
<tr>
<td>16</td>
<td>BZ ok</td>
<td>a. Increase air fan speed</td>
<td>To lower back-end temperature and maintain burning zone temperature</td>
</tr>
<tr>
<td></td>
<td>OX high</td>
<td>b. Increase fuel rate</td>
<td></td>
</tr>
<tr>
<td>17</td>
<td>BE low</td>
<td>a. Reduce air fan speed slightly</td>
<td>To lower oxygen</td>
</tr>
<tr>
<td></td>
<td>BZ ok</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>OX high</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>BE ok</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Mamdani Fuzzy Controller for Cement Kiln

- **Linguistic Controller**
  - **Antecedents**
    - BZ: Temperature in burning zone
    - OX: Oxygen in exhaust gas
    - BE: Temperature at end of kiln
  - **Consequents**
    - CR: Coal feed rate
    - DP: Exhaust damper position
  - 27 fuzzy rules, e.g.,
    - If BZ is OK and OX is low and BE is low, then set CR to large, and DP to large

*Controller is apparently symbolic, but symbols must have values for computation, i.e., Fuzzy Membership Functions*
Probable Cause

- The National Transportation Safety Board determines that the probable cause of the USAir flight 427 accident was
  - a loss of control of the airplane resulting from the movement of the rudder surface to its blowdown limit.
- The rudder surface most likely deflected in a direction opposite to that commanded by the pilots as a result of
  - a jam of the main rudder PCU servo valve secondary slide to the servo valve housing offset from its neutral position and
  - overtravel of the primary slide.

Gain-Scheduling (Takagi-Sugeno) Fuzzy Control Systems
(Schramm, Gopisetty, and Stengel, 1998)
Failure Detection for Simulated Rudder Failure

- Rudder reversal occurs at \( t = 10 \) s
- Heading angle change commanded at \( t = 20 \) s

Simulated Reconfiguration

- Failure detection logic detects nothing until rudder effect is expected
- Once detected, control signal is reversed
Fuzzy Logic ≠ Fuzzy Thinking

- Quantitative approach to reasoning under uncertainty
- “Possibility theory” vs. Probability theory (Lotfi Zadeh, 1978)
- Relationship to other uncertainty belief systems of artificial intelligence, e.g.,
  - Bayesian belief network
  - Dempster-Shafer theory
  - Transferable belief model
  - Certainty factors
- Propositions are true or false only within the context of a paradigm

Next Time: Probability and Statistics
Supplemental Material

Turing Machine for Euclid’s Algorithm

Penrose, R., *The Emperor’s New Mind*, 1989, p. 41

Since we wish to be able to include numerical data as part of our input, we shall want to have a way of describing ordinary numbers (by which I here mean the natural numbers 0, 1, 2, 3, 4, . . .) as part of the input. One way to do this might be simply to use a string of \( n \) 1s to represent the number \( n \) (although this could give us a difficulty with the natural number zero):

\[
1 \rightarrow 1, \quad 2 \rightarrow 11, \quad 3 \rightarrow 111, \quad 4 \rightarrow 1111, \quad 5 \rightarrow 11111, \quad \text{etc.}
\]

This primitive numbering system is referred to (rather illogically) as the *unary* system. Then the symbol ‘0’ could be used as a space to separate different numbers from one another. It is important that we have such a means of separating numbers from one another since many algorithms act on sets of numbers rather than on just single numbers. For example, for Euclid’s algorithm, our device would need to act on the *pair* of numbers \( A \) and \( B \). Turing machines can be written down, without great difficulty, which effect this algorithm. As an exercise, some dedicated readers might perhaps care to verify that the following explicit description of a Turing machine (which I shall call *EUC*) does indeed effect Euclid’s algorithm when applied to a pair of unary numbers separated by a 0:

\[
\begin{align*}
00 & \rightarrow 00L, \quad 01 & \rightarrow 11L, \quad 10 & \rightarrow 101R, \quad 11 & \rightarrow 11L, \quad 00 & \rightarrow 1010R, \\
01 & \rightarrow 110R, \quad 10 & \rightarrow 1000L, \quad 11 & \rightarrow 111R, \quad 00 & \rightarrow 1000R, \quad 10 & \rightarrow 1010R, \\
00 & \rightarrow 1110L, \quad 01 & \rightarrow 1101L, \quad 11 & \rightarrow 1101L, \quad 10 & \rightarrow 1110L, \\
11 & \rightarrow 1000LL, \quad 100 & \rightarrow 10001L, \quad 1001 & \rightarrow 1001LL, \quad 1000 & \rightarrow 1000R, \\
1001 & \rightarrow 111L, \quad 1000 & \rightarrow 0STOP, \quad 1001 & \rightarrow 10101R.
\end{align*}
\]
Turing Machines in Biological Cells

- Human body
  - ~100 trillion cells
  - typical diameter: ~10 µm

The Central Dogma:
Core Process of Protein Production

- A gene is a sequence of nucleotides on a chromosomal strand of DNA
- Single-stranded messenger RNA (mRNA) expresses information from DNA to form a protein, a sequential molecule of amino acids
- Information coded in nt triplets (codons), with open reading frames defined by Start and Stop codons

<table>
<thead>
<tr>
<th>First Position</th>
<th>The Genetic Code</th>
<th>Second Position</th>
<th>Third Position</th>
</tr>
</thead>
<tbody>
<tr>
<td>U</td>
<td>F</td>
<td>S</td>
<td>Y</td>
</tr>
<tr>
<td>F</td>
<td>S</td>
<td>Y</td>
<td>C</td>
</tr>
<tr>
<td>L</td>
<td>S</td>
<td>Stop</td>
<td>Stop</td>
</tr>
<tr>
<td>U</td>
<td>C</td>
<td>A</td>
<td>G</td>
</tr>
<tr>
<td>C</td>
<td>L</td>
<td>P</td>
<td>H</td>
</tr>
<tr>
<td>L</td>
<td>P</td>
<td>H</td>
<td>R</td>
</tr>
<tr>
<td>L</td>
<td>P</td>
<td>Q</td>
<td>R</td>
</tr>
<tr>
<td>A</td>
<td>I</td>
<td>T</td>
<td>N</td>
</tr>
<tr>
<td>I</td>
<td>T</td>
<td>N</td>
<td>S</td>
</tr>
<tr>
<td>I</td>
<td>T</td>
<td>K</td>
<td>R</td>
</tr>
<tr>
<td>G</td>
<td>M (start)</td>
<td>T</td>
<td>K</td>
</tr>
<tr>
<td>V</td>
<td>A</td>
<td>D</td>
<td>G</td>
</tr>
<tr>
<td>V</td>
<td>A</td>
<td>E</td>
<td>G</td>
</tr>
<tr>
<td>V</td>
<td>A</td>
<td>E</td>
<td>G</td>
</tr>
<tr>
<td>U</td>
<td>C</td>
<td>G</td>
<td>C</td>
</tr>
</tbody>
</table>

U A C G G C
System View of The Core Process

DNA → Transcription Process → mRNA → Translation Process → Protein

- Polymerase unzips dual-stranded DNA
- Exons of DNA code mRNA sequence with help of transcription factors
- Introns and extragenic DNA are ignored (deleted) in RNA

mRNA enters ribosome, which is composed of ribosomal RNA (rRNA) and proteins
- Transfer RNA (tRNA) transports amino acids to ribosome
- Amino acids assembled into protein according to RNA sequence (3 nt per codon)

~20,000 human genes
~3 billion base pairs/human gamete

Block Diagram of the Protein Process

DNA → Transcription Process → mRNA → Translation Process → Protein

Proteins
- RNA polymerase III
- Transcription Factors

Proteins
- Ribonucleasees
- Transferase

Proteins
- Cleave, Chem. Mod.

Proteins
- Spliceosome, polyA polymerase

Proteins
- Synthesis Enzymes
- Initiation Factors
- Peptide transferase
- Elongation Factors
- Release Factor

Proteins
- Cleavage
- Methylation
- Phosphorylation
- Acetylation
- Hydroxylation
- Lipids
- Oligosaccharides

Proteins
- Ribonucleasees

Proteins
- RNA polymerase II
- Transcription Factors

Proteins
- RNA polymerase I
- Transcription Factors

Proteins
- RNA polymerase
- Transcription Factors

Proteins
- RNA polymerase
- Transcription Factors

Proteins
- RNA polymerase
- Transcription Factors
Protein Production is Dynamic

Transcription

Translation

~50 bases/sec

~10 amino acids/sec

Polymerases and Ribosomes as Turing Machines
DNA/RNA Molecules

Alternative Splicing
How Do Slide Rules and Calculators Work?

- **Abacus**
  - Unary number system
  - [http://gwydir.demon.co.uk/jo/numbers/machine/abacus.htm](http://gwydir.demon.co.uk/jo/numbers/machine/abacus.htm)

- **Mathematical Tables**
  - Manual calculations

- **Slide rule**
  - Logarithmic scales
  - [https://www.youtube.com/watch?v=uUzSSlVnAHk](https://www.youtube.com/watch?v=uUzSSlVnAHk)

- **Mechanical Calculator**
  - Add, subtract, and shift
  - [https://www.youtube.com/watch?v=7S0BETniokI](https://www.youtube.com/watch?v=7S0BETniokI)
Friden STW-10 Calculator Mechanism

Bi-Quinary Control Panel Lights for the IBM 650

Computation with 10-digit words
Early Computers - DEC

PDP-1

SpaceWar

https://en.wikipedia.org/wiki/PDP-1

PDP-6

Time-Sharing

https://en.wikipedia.org/wiki/PDP-6

Computer Transistor Counts

Moore’s Law – The number of transistors on integrated circuit chips (1971-2016)

Moore’s law describes the empirical regularity that the number of transistors on integrated circuits doubles approximately every two years. This advancement is important as other aspects of technological progress – such as processing speed or the price of electronic products – are strongly linked to Moore’s law.


The data visualization is available at OurWorldinData.org. There you find more visualizations and research on this topic.

Licensed under CC-BY-SA by the author Max Krieger.