Floating Point

- Generally use floating point, which is a "finite precision" system
  - introduced "rounding" errors
- standard is IEEE 754 (1985)
  - adherence made numerical code more portable and reliable
- as opposed to fixed point: point is always after the 10^0 place
  - 1234.567
  - 1.3
  - 0.001
- floating point: point can "float"
  - 1.234567 * 10^-3
  - 1.3 * 10^0
  - 1.0 * 10^-3

- General floating point system

  b  base
  p  number of digits of precision
  [U, L] exponent range

<table>
<thead>
<tr>
<th></th>
<th>b</th>
<th>p</th>
<th>L</th>
<th>U</th>
<th>field width</th>
</tr>
</thead>
<tbody>
<tr>
<td>IEEE SP</td>
<td>2</td>
<td>23 +1</td>
<td>-126</td>
<td>127</td>
<td>(1+8+23=32)</td>
</tr>
<tr>
<td>IEEE DP</td>
<td>2</td>
<td>52 +1</td>
<td>-1022</td>
<td>1023</td>
<td>(1+11+52=64)</td>
</tr>
</tbody>
</table>

- Floating point number \( x \)

  \[
  x = \pm \left( \frac{d_0 + d_1 + d_2 + \ldots + d(p-1)}{b^{p-1}} \right) \times b^E
  \]

  \( 0 \leq d_i \leq b-1, \quad i = 0, \ldots, p-1 \) (p digits)

  \( L \leq E \leq U \)

  - mantissa: \( d_0d_1\ldots d(p-1) \)
  - exponent: \( E \)

Example 1 (1):

b = 2
p = 3
L = -1
U = 1

start enumerating possibilities:

\[
\begin{align*}
\pm 0.00 & \rightarrow 0 \\
\pm 0.01 & \rightarrow 0.001 \\
\pm 0.10 & \rightarrow 0.1
\end{align*}
\]

- lots of duplicates
- non-unique representation

Normalization

- require the leading digit to be non-zero
- so mantissa, \( m \)

\( 1 \leq m < b \)

- nice because:
  - representation is now "unique"
  - don't waste digits on any leading 0's
  - for binary base, leading digit must be 1
  - so don't need to store it, just assume number is 1.d1d2.dp
  - gain an extra bit of precision!

Properties

- finite and discrete system
- finite: how many (normalized) numbers can be represented?
  count them:

\[
2 \times (b - 1) \times b^{(p-1)} \times (U - L + 1)
\]

- what's the smallest (positive) normalized number? or "underflow level (UFL)"

\( 1.0 \ldots 0 \times b^L \times b^U \)

- what's the biggest normalized number? or "overflow level (OFL)"

\[
(b-1)\ldots (b-1) \ldots (b-1) \times b^U
\]

Example 1 (2):

b = 2
p = 3
L = -1
U = 1
- number of normalized
  \[ 2 \cdot (b - 1) \cdot b^{(p-1)} \cdot (U - L + 1) + 1 \]
  \[ = 2 \cdot (2 - 1) \cdot 2^{(3-1)} \cdot (1 - -1 + 1) + 1 \]
  \[ = 2 \cdot 1 \cdot 4 \cdot 3 + 1 \]
  \[ = 25 \]
- UFL
  \[ b^L \]
  \[ = 2^-1 \]
  \[ = .5 \]
- OFL
  \[ (1 - b^{(U+1)}) \cdot b^{(U+1)} \]
  \[ = (1 - 2^2) \cdot 2^2 \]
  \[ = 3.5 \]

PICTURE of representable numbers
- note evenly spaced only for a given exponent

|    |    |    |  |  |  |  |||||   |   |||||  |  |  |  |    |    |    |
|-----|-----|-----|---|---|---|---|----------------|---|----------------|---|---|---|---|---|---|
-4   -3  -2       -1       0       1           2         3         4

Subnormals
- normalized numbers: gap between 0 and \( b^L \)
- fill in by allowing denormalized or subnormal numbers
- can make use of capacity for non-normalized numbers by allowing leading 0's
- though precision won't be full precision, since have leading 0's

Example 1(3):

\[ \begin{array}{cccccccc}
-4 & -3 & -2 & -1 & 0 & 1 & 2 & 3 & 4 \\
\end{array} \]
- allows 6 new numbers around 0
- new smallest number is \((0.01)_2 \cdot 2^{-1} = (0.125)_10\)
- called "gradual underflow" because we gradually lose precision
- implementation: reserved value of exponent field
- leading bit not stored

Exceptional values
- \( \text{Inf} \)
  - dividing finite number by 0
  - exceeding OFL
- \( \text{NaN} \)
  - undefined operation \( 0/0, \text{Inf}/\text{Inf}, 0*\text{Inf} \)
  - implemented through reserved values of exponent field