Work Done

● Cross-platform build of C library
  ○ Supporting different configurations of precision
  ○ Basic algebraic operations \{+, -, x, /\}
● Runtime memory and code profiling
● Test harnesses
  ○ Mathematica verification
  ○ Unit testing
  ○ Code coverage analysis
● Documentation generation
● C++/Python wrapper
● Runtime algorithm analysis
A Handy way to look at it

\{±∞, …, -2, 1, 0, 1, 2, …\}

\{1000, …, 1111, 0001, 0010, …\}

\{□, …, □, □, □, □, □, …\}
Introduce arithmetic

- \[ \begin{array}{c}
\textcolor{green}{+} & \textcolor{orange}{\times} & = & \textcolor{purple}{\times} \\
\textcolor{orange}{\times} & \textcolor{blue}{\times} & = & \textcolor{blue}{\times} \\
\end{array} \]

- Handle uncertainty (◧◨)
- Giant tables
SORNs

\{ [\text{\textcolor{green}{\textbf{\small{\large{\blacksquare}}}}}, (\text{\textcolor{purple}{\footnotesize{\textbf{\large{\blacksquare}}}}}), \text{\textcolor{yellow}{\footnotesize{\textbf{\large{\blacksquare}}}}}, (\text{\textcolor{red}{\footnotesize{\textbf{\large{\blacksquare}}}}})] \}

- Split non-contiguous SORNs
- Look at low/high for basic arithmetic
  - Similar to intervals
Existing types

- Conversion into Unum System
  - Primitive datatypes
  - Integers/Rationals/Floating Point

- Conversion out
  - Interval range
    - Ex. (1.5, 6]
  - Arrays of intervals (non-contiguous SORNs)
Lattice Definition

- Lattice is defined by traditional primitives
  - Integer, rational, irrational (floating point)
- Called backing types in code
- Massive optimization point
  - Binary tree lookup
  - Initial estimations
  - Caching
Future/Current Development

- Automatic algorithmic analysis
  - Using C++ library
    - Wrapper class (NumberSystem<double>, NumberSystem<Unum>)
- Remove symmetry
Ways to Hardware

1. Simulate computation
   a. Use traditional primitives
   b. Get Co-domain/Bounds from executions

2. Non-linear data regression
   a. Output $f(x)$
      i. Monotonically increasing
      ii. Conform to symmetry constraints

\[ f(x) = \{ \text{⬛}, \ldots, \text{⬛}, \text{⬛}, \text{⬛}, \text{⬛}, \text{⬛}, \ldots \} \]
Ways to Hardware

Numerical integration using a taylor series

Arbitrary Precision

\[ \int_{0}^{\pi} \cos x \, dx \]
Possible modifications

\{\ldots, \square,(\text{blue}), \text{green},(\text{purple}), \text{purple},(\text{yellow}), \text{yellow},(\text{blue}), \text{blue}, \ldots\} \\
\downarrow \\
\{\ldots, \text{orange},(\text{green}), \text{green},(\text{purple}), \text{purple},(\text{yellow}), \text{yellow},(\text{blue}), \text{blue}, \ldots\}

closer to intervals
Thank you.