The Ada Numerics Model

J-P. Rosen
Adalog

What is Numerical Analysis?

- The mathematical set $\Re$ cannot be represented on a computer.
  - ☞ Infinite number of values, even for a bounded segment.
  - ☞ A computer can (at most) represent rational values.
- But Π *is* irrational!

Numerical analysis: the art of making not too wrong computations in the *real* world, using only (a finite subset of) rational numbers

Hardware Formats and Languages

- Many hardware formats
  - http://www.quadibloc.com/comp/cp0201.htm describes 76 different floating point formats!
  - IEEE-754: 5 standard formats, (3 binary, 2 decimal), + extensions
  - Vax:

Size	Exponent	Mantissa
32 bits	8	23
64 bits	8	55
	11	52
128 bits	15	112

More accuracy

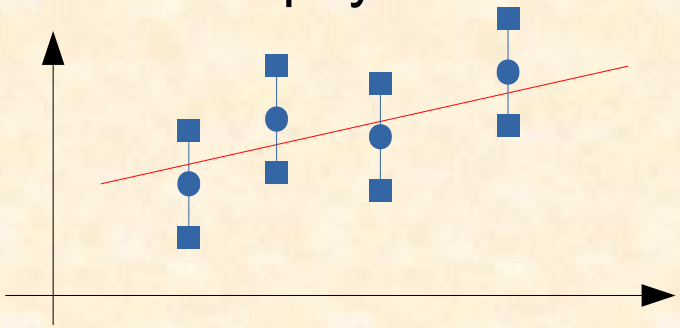
More range

- Programming languages often have a limited number of formats
  - Legacy from FORTRAN (REAL and DOUBLE PRECISION)
    - ✔ It's all what machines of the time had…
  - No requirements about accuracy
  - No portability of computations

Ada Approach

- Requirements for Ada:
  - ☞ Portable results on various architectures.
  - ☞ Without efficiency penalty
- Solution :
  - ☞ Follow the model of approximate values in physics.

    A value represents an interval



  - ☞ Two kinds of approximations :
    - ✔ Relative: $\frac{\Delta X}{X} = Constant$ $\quad U = 5\,V \pm 5\,\%$
    - ✔ Absolute: $\Delta X = Constant$ $\quad U = 5\,V \pm 0.1\,V$

Real types

- In Ada, two kinds of “real” types:
  - ☞ Floating point types (relative approximation):

```
type name is digits number_of_digits [ range min .. max ] ;
type Length is digits 5 range 0.0 .. 40.0E6;
```

  - ☞ Fixed point types (absolute approximation):

```
type name  is delta step range min .. max ;
type Volts is delta 0.01 range 0.0 .. 100.0;

type name is delta step digits number_of_digits [ range  min .. max ];
type Euros is delta 0.01 digits 11;
```

Binary

Decimal

- You specify the *minimal* accuracy requirements, the compiler choses the most appropriate underlying type
  - ☞ All available hardware types can be used
  - ☞ If no appropriate type is available: compilation is *rejected*

Model of Arithmetic

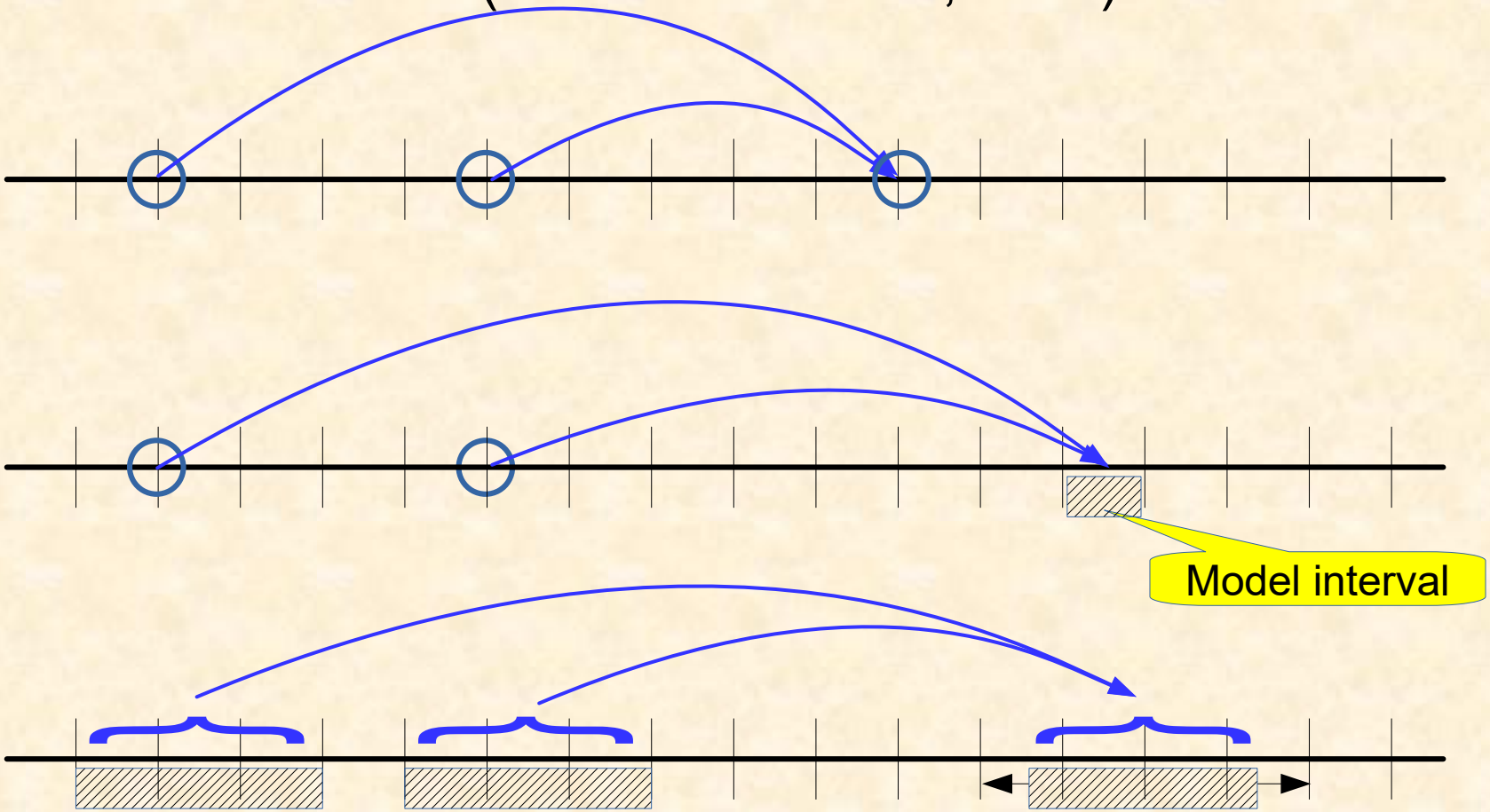
- Static evaluation: mathematically exact
  - ☞ Full portability of static expressions
  - ☞ The compiler must use extended precision rational arithmetic

(1.0 / 3.0) * 3.0 = 1.0 *Exactly*

- Dynamic evaluation: two modes
  - ☞ Relaxed modes (all compilers)
    - ✔ Guarantee on the accuracy of data
  - ☞ Strict mode (Numerics annex - G):
    - ✔ In addition, guarantee on the accuracy of operations
    - ✔ Including for the numerics library: elementary functions, linear algebra, and properties of the random number generators

Required Accuracy in Strict Mode

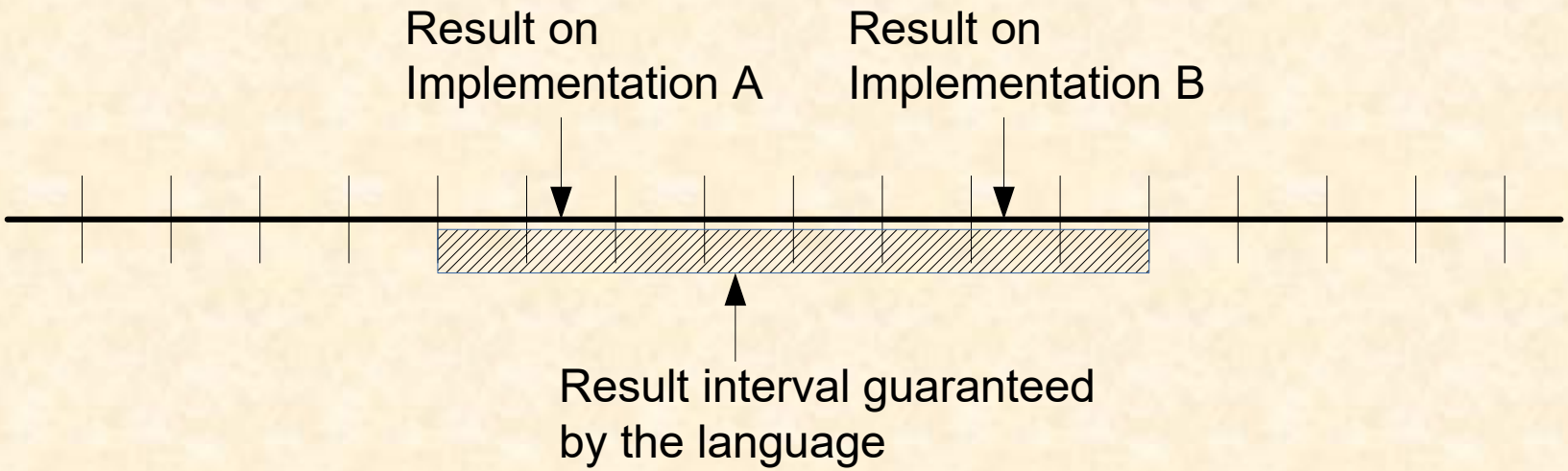
- Brown's model (interval arithmetic, 1980)



No underflow!

Portability of Real Computations

- The result of a computation may vary across implementations



- But these different results must belong to an interval guaranteed by the language
  - ☞ The interval can be computed independently of the implementation
  - ☞ Compatible with any hardware

Fixed Point vs. Floating Point

- Floating point:
  - ☞ More accuracy close to 0.0
  - ☞ (absolute) accuracy diminishes with greater values

- Fixed point
  - ☞ Same accuracy all over the range

  - ☞ Typical uses of fixed points:
    - ✔ Time
    - ✔ Physical measures
    - ✔ Money

Numerical Portabilities

- *A posteriori* accuracy

```
type Scale is new Float;
type Accurate_Scale is new Long_Float;
```

Attributes allow for the determination ***afterwards*** of the accuracy of computations, which depend on the machine.

- *A priori* accuracy

```
type Scale is digits 7;
type Fixed_Scale is delta 0.01 range 0.0 .. 1000.0;
```

Required accuracy is ***guaranteed*** independently of implementation

Conclusion

- A number with a decimal point is not always a Float!
- Ada offers access to all floating point types supported by hardware
  - ☞ Not just two!
- Ada offers access to other forms of real types
  - ☞ Not just floating points!

The choice is yours

- ... and tell your numerical friends!

Questions?