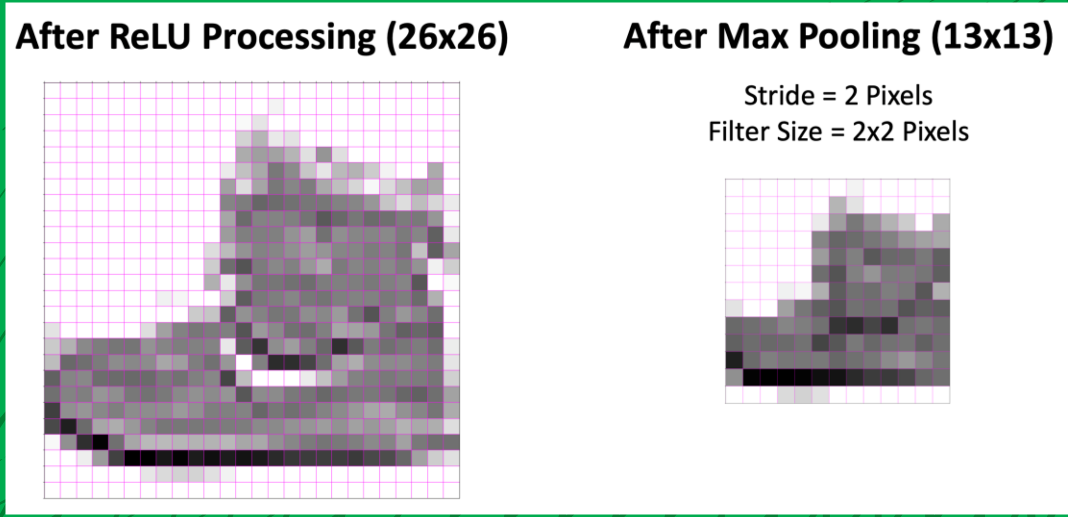
Approximate Computing

Presented by Marvin Sevilla

Overview

- What is Approximate Computing
- Real Life Examples
- Practical Applications
- Under The Hood
- Practical Coding Demo
- Closing Remarks



What is Approximate Computing?

A technique for achieving a satisfactory computational result with reliable and controllable error thresholds, yielding computationally faster algorithms that use less energy at the expense of lower quality results with negligible error

What is Approximate Computing?

A technique for achieving a satisfactory computational result with reliable and controllable error thresholds, yielding computationally faster algorithms that use less energy at the expense of lower quality results with negligible error

What is Approximate Computing?

It is undeniably faster and lower-power as it is unreliable

What is Approximate Computing?

Domains of Application for Approximate Computing

- Logic & Circuit
- Microarchitecture (atomic functions)
- Algorithms
- Parameters

What is Approximate Computing?

Parts of Every Approximate Computing Technique

- Error Occurrence
- Degradation
- Level
- Evaluation

What is Approximate Computing?

Error Occurrence Degradation Level Evaluation

Errors can be either non-deterministic or deterministic

Non-Deterministic: happens randomly

Deterministic: able to consistently reproduce the same error

What is Approximate Computing?

Error Occurrence 「Degradation」 Level Evaluation

Focuses on error rates and outcomes

Error Rate: The frequency and magnitude of errors (toggle-able or gradual degradation)

Error Outcomes: Can either be Bounded (Toggleable), Catastrophic (Toggleable and Gradual) or Graceful (Gradual)

What is Approximate Computing?

Error Occurrence Degradation Level Evaluation

Three Different Levels:

- Transistor
- Logic
- Algorithmic

What is Approximate Computing?

Error Occurrence   Degradation   Level   Evaluation

Evaluation of approximate computing techniques from an atomic or application point-of-view

What is Approximate Computing?

The ultimate success of an approximated application is its reliability

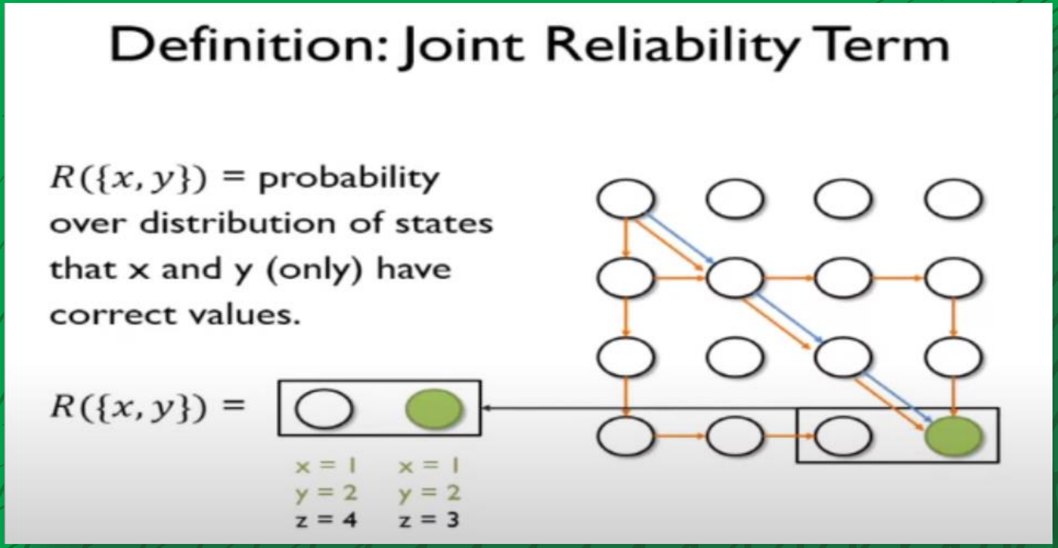
Reliability looks like executing the process multiple times, and the end-state of the relaxed version matches the original version a specified number of times, we deem the program reliable to that point

「What is Approximate Computing?」

```
uint interpolation(int x, int y, int src[][], int dest[][])
{
    int x_src = map_x(x, src, dest),
        y_src = map_y(y, src, dest);

    int xs[MAX_N], ys[MAX_N];
    uint n = get_neighbors(i_src, j_src, src, xs, ys);

    uint val = 0;

    relax (n) with (n <= old(n));

}
```

n in the relaxed program is less than in the original

```
n<r> <= n<o>
```



What is Approximate Computing?

Approximation does not have to be a fixed feature in the system.

Practical Application

How Can You Use Approximate Computing Techniques?

Practical Application

Such as Code Perforation (e.g. Loops), Function Substitution, Approximate Memoization, Relaxed Synchronization, Approximate Hardware

Practical Application

Such as Code Perforation (e.g. Loops), Function Substitution, Approximate Memoization, Relaxed Synchronization, Approximate Hardware

Choosing between the use of single and half precision numbers

Practical Application

Such as Code Perforation (e.g. Loops), Function Substitution, Approximate Memoization, Relaxed Synchronization, Approximate Hardware

Choosing between the use of single and half precision numbers

A difference of **~3 Bytes/13 bits**

Practical Application

Such as Code Perforation (e.g. Loops), Function Substitution, Approximate Memoization, Relaxed Synchronization, Approximate Hardware

An ALU that concludes **2+2 = 6** or Secondary memory that interprets memory location **0x3452** as **13394** in decimal, but retrieves memory location **40**

Practical Application

Voltage Scaling + Dynamic Voltage and Frequency Scaling (DVFS)

Reducing the operating voltage of the processor, going even as far as adjusting the voltage dynamically based on the input/workload

Practical Application

Error Resilient Algorithms

Like the Newton-Raphson method, lossy compression (e.g. JPEGs), or sorting algorithms like TeraSort

Practical Application

Application Approximation

Instead of approximating the entire application, approximate only parts of it and grant the rest higher levels of accuracy

Practical Application

Quality-Aware Programming

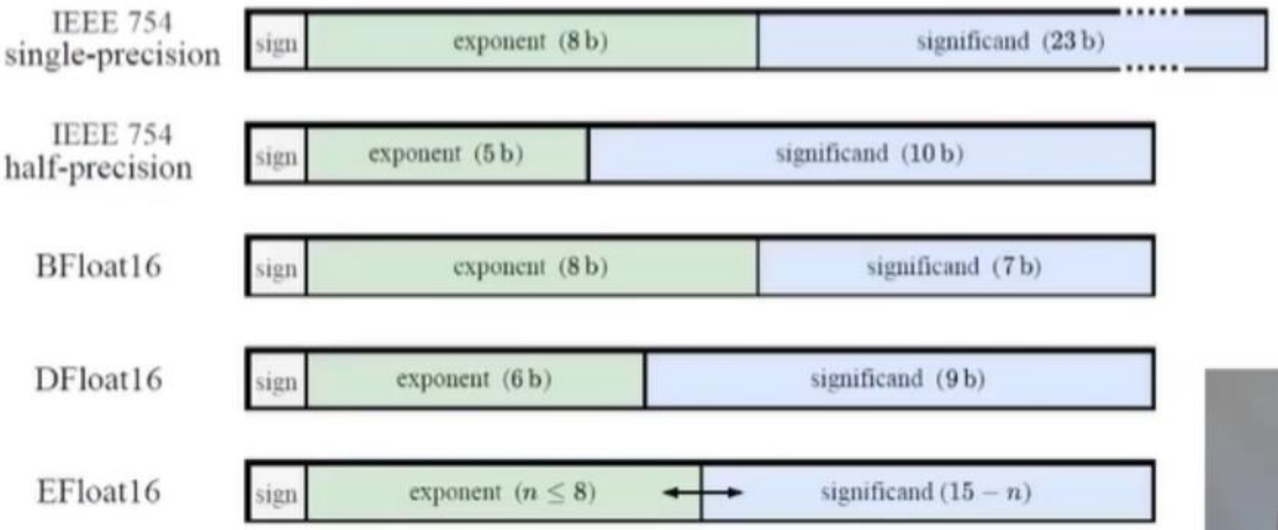
A programming tactic which requires that software developers be keenly aware and constantly attempting to reduce a program's space complexity

E.g. Datatype consideration

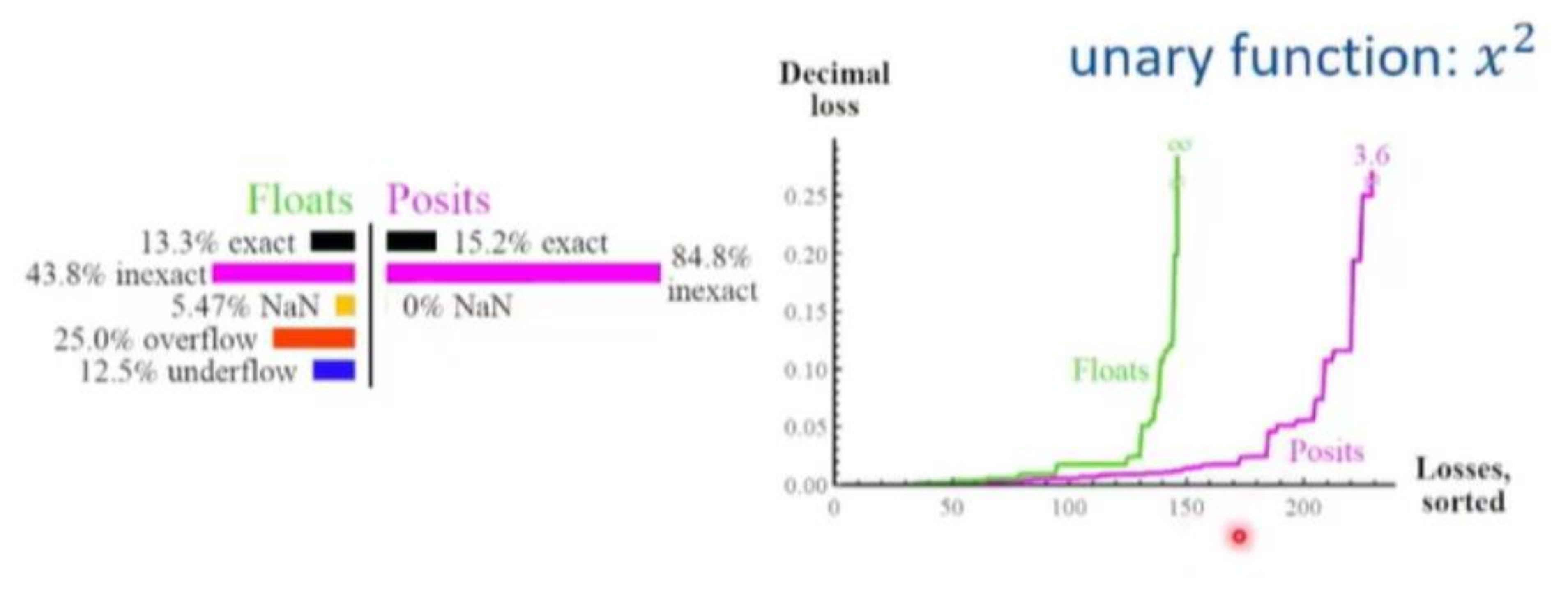
Practical Application

Quantization

- Fixed Point
- Block Floating Point
- Floating Point



「Practical Application」



Real Life Examples

Where is Approximate Computing Used Right Now?

Soft Examples

Generally used to meet project budget or time requirements

Fixed point, block floating point, floating point, and posit-representation

Concrete Examples

Helps computing clusters used for deep-learning reduce energy consumption and cost associated with IC-density

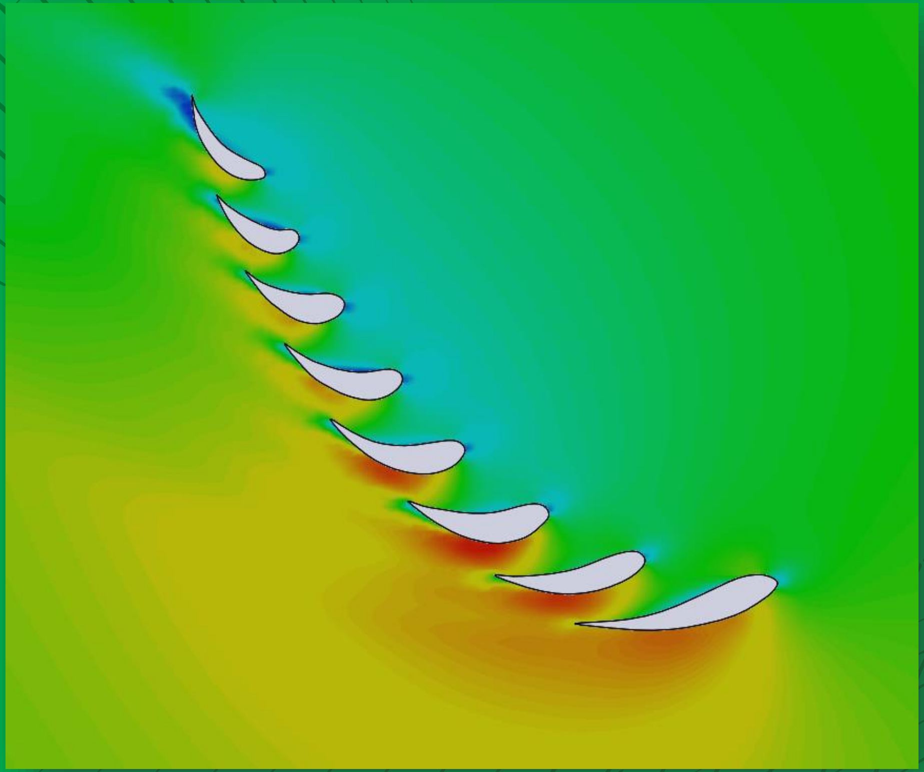
"...approximate computing techniques need to exploit the error-tolerance of humans and neural networks. This optimization can lead to lightweight neural networks." (21, Srivastava, Srishti, et al. "A Survey of Deep Learning Techniques for Vehicle Detection from UAV Images." CSE Department, IIT Dharwad, India, ECE Department, NIT Trichy, India, ECE Department, IIT Roorkee, India, 2024.)

Concrete Examples

"...approximate computing techniques need to exploit the error-tolerance of humans and neural networks. This optimization can lead to lightweight neural networks."

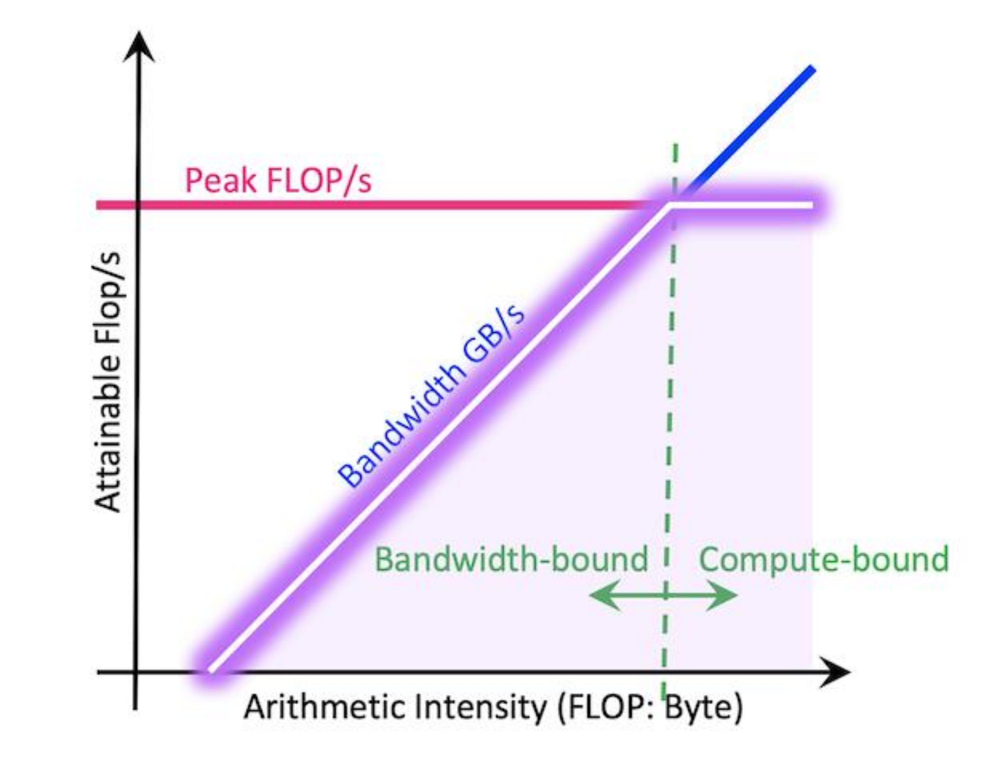
(21, Srivastava, Srishti, et al. "A Survey of Deep Learning Techniques for Vehicle Detection from UAV Images." CSE Department, IIT Dharwad, India, ECE Department, NIT Trichy, India, ECE Department, IIT Roorkee, India, 2024.)

Concrete Examples



ANSYS Mesh

Concrete Examples



Single v. Double Precision GPU Computation
With Roofline Graph

Concrete Examples

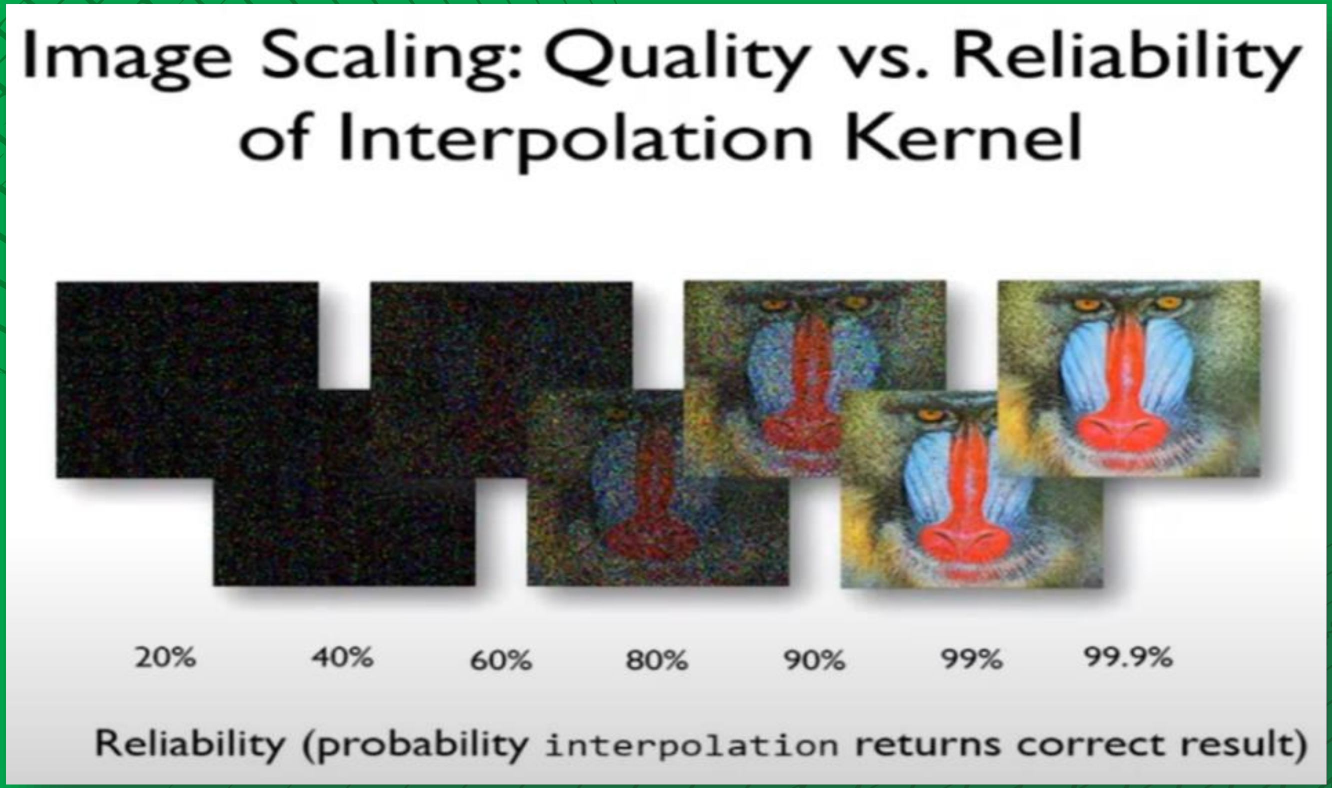


Image Compression

Under The Hood

How Does Approximate Computing Generally Work?

General Steps

Achieve Minimal Error Result
Establish a Safe Execution Envelope
Relax the Semantics of the Program
Verify the Program’s Result

General Steps

Achieve Minimal Error Result

Develop quantitative verification systems

Utilize error-resilient domains

- Human Perception
- Data Redundancy
- Generally, areas with "no golden result"

General Steps

Establish a Safe Execution Envelope

- Prominent concerns with radically inaccurate results, rarely correct results, crashes or other malicious actions
  - Mitigated via error bounding, for example, using assertions (e.g. **a / b** such that **b ≠ 0**, where **b≠0** is the assertion)

General Steps

Relax the Semantics of the Program

If the relaxed version of a program is different to the original, when will they converge? When can they be related again?

- There are synchronization points where the 2 converge
- We bring the assertion or result to a previous (or sometimes future) synchronization point and verify it there

Verification reuses existing reasoning from the original program

- Languages such as Coq analyze raw source code using user-given inference rules to validate logic

General Steps

Verify the Program Result

- End performance can be measured via things like data processing and feature extraction
  - Data processing refers to the quality of filtering, compression, or equivalent action which can be measured by human perception
  - Feature extraction refers to the identification of properties or characteristics of a data instance using algorithms/methods
- All operations’ reliabilities are evaluated (reading, arithmetic, etc.)

General Steps

Verify the Program Result

Put junk in, get junk out

General Steps

Verify the Program Result

Example of Quantified Reliability

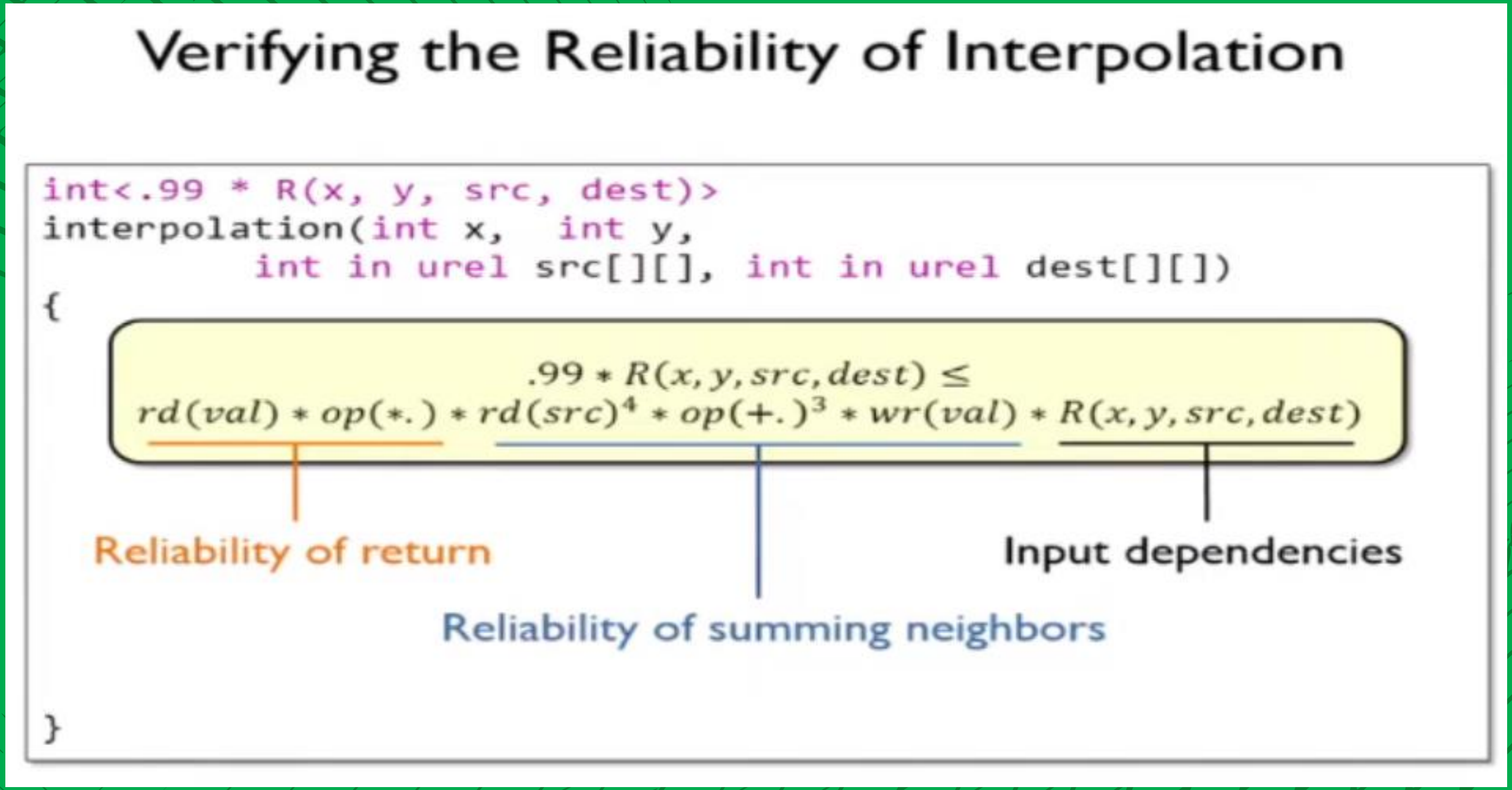
.99 * R(x,y,src,dest) <= rd(val) * op(*.) * R(val)

In English:

The Specified Bound * Reliability of Operation <= the probability we read the value right * the probability the operand performs right * the probability that the input value itself is right

General Steps

Verify the Program Result



Practical Coding Demo

How Datatype Choice Affects Performance

Closing Thoughts

Key Takeaways for Your Computer Science Journey

Closing Thoughts

Probability operations must stay within the established acceptable error threshold

Computations must still meet reliability requirements which can be ensured using error detection and correction mechanisms or algorithmic adjustments that compensate for potential errors

Error-tolerant software must be tested, validated, or verified at an appropriate frequency to ensure reliable results despite the uncertainties introduced by the computer's hardware or lower-level processes

Questions?