A Quantum Data Structure For Classical Computers

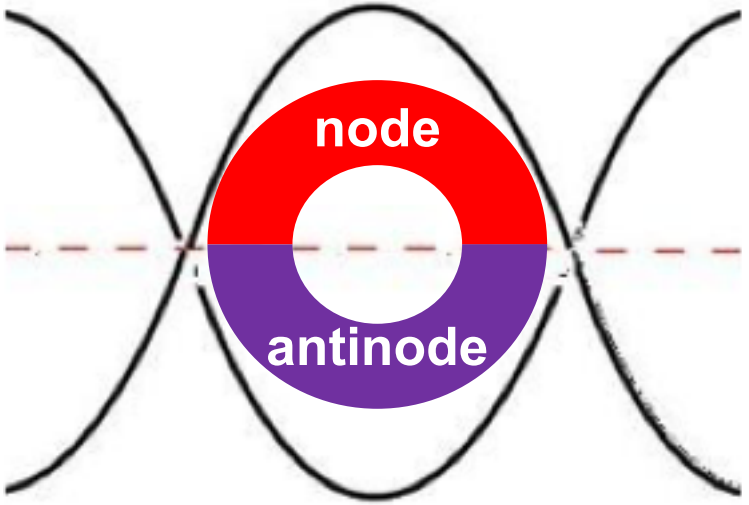
C++ now

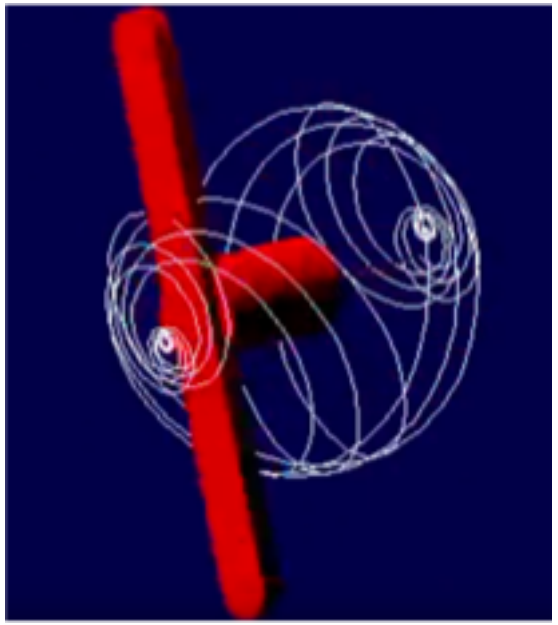
charley bay

charleyb123 at gmail dot com

Today's Agenda

1. **Quantum Behavior** In Classical Systems
2. **State**, **Unknowable State**
3. **Quantization**
   - Defining A Quantum
   - Quantum Jump
4. **Classical vs. Quantum Physics**
   - Range of Continuous vs. Discrete Values
5. **Qubit**, **Valid States**
   - Eigenstate vs. Eigenvalue
   - Wavefunction Collapse
6. **Time Is A Technology**
7. **Quantum Data Structure**
8. **Key Observations**, **Conclusions**





For Today:

You actually *(intuitively!)* ***KNOW*** most of this!
…(except possibly for One Thing)…

*Much of Today is "**Back To Basics**!"*

- We will *(try to)* go *slow*
- *Please:* ***Interrupt!*** ***Ask Questions!***

Primary *Goal:*

Quantum:
Seems "weird", but ends up being *intuitive*, *obvious*, and *natural*

*Consistent with **current practice**, and what we **Know To Be True***

Secondary *Goal:*

Dispel some myths, correct misuse of term "quantum"

Quantum Behavior In Classical Systems

Tomorrow Is Today

C++ now

What is the output?

```
//g++  5.4.0
#include <iostream>
#include <climits>
int foo(int x) {
  return (x + 1) > x;
}
int main() {
  std::cout << ((INT_MAX + 1) > INT_MAX) << "\n";
  std::cout << foo(INT_MAX) << "\n";
  return 0;
}
```

```
$ clang++ -O foo.cpp ; ./a.out
```

What is the output?

```
//g++  5.4.0
#include <iostream>
#include <climits>
int foo(int x) {
  return (x + 1) > x;
}
int main() {
  std::cout << ((INT_MAX + 1) > INT_MAX) << "\n";
  std::cout << foo(INT_MAX) << "\n";
  return 0;
}
```

Either `((x+1) > x)` is always mathematically true, or the result is `undefined` (so it does not matter), so the result must always be `1` (`true`)

*Is **`true`**?*
`(x+1) > (x)`

+1 to `INT_MAX` is undefined

```
$ clang++ -O foo.cpp ; ./a.out
0
1
```

What is the output?

Either `((x+1) > x)` is always mathematically true, or the result is `undefined` (so it does not matter), so the result must always be `1` (`true`)

TWO values
for same *computation*,
in the same *program*,
in the same *process*
invocation

```
//g++  5.4.0
#include <iostream>
#include <climits>
int foo(int x) {
  return (x + 1) > x;
}
int main() {
  std::cout << ((INT_MAX + 1) > INT_MAX) << "\n";
  std::cout << foo(INT_MAX) << "\n";
  return 0;
}
```

Is `true`? `(x+1) > (x)`

+1 to `INT_MAX` is undefined

```
$ clang++ -O foo.cpp ; ./a.out
0
1
```

CppCon 2017: John Regehr
“Undefined Behavior in 2017”
https://www.youtube.com/watch?v=v1COuU2vU_w

What is the output?

```
//g++  5.4.0
#include <iostream>
#include <climits>
int foo(int x) {
  return (x + 1) > x;
}
int main() {
  std::cout << ((INT_MAX + 1) > INT_MAX) << "\n";
  std::cout << foo(INT_MAX) << "\n";
  return 0;
}
```

Is ***true?*** ***(x+1) > (x)***

Either `((x+1) > x)` is always mathematically true, or the result is `undefined` (so it does not matter), so the result must always be `1` (`true`)

+1 to `INT_MAX` is undefined

```
$ clang++ -O foo.cpp ; ./a.out
0
1
```

TWO values
for same *computation*,
in the same *program*,
in the same *process*
invocation

Superposition:
Simultaneously holding all valid states

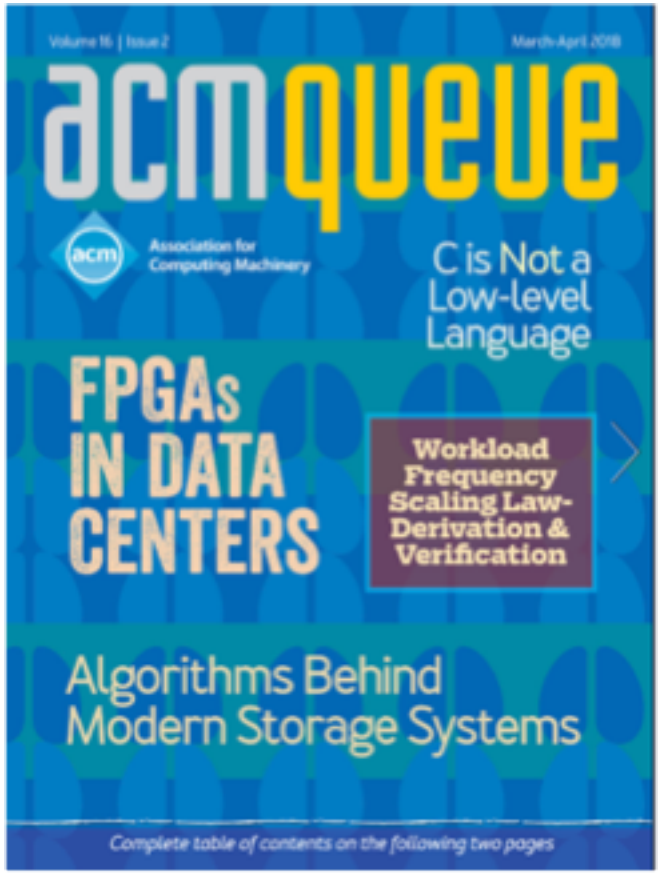
CppCon 2017: John Regehr
“Undefined Behavior in 2017”
https://www.youtube.com/watch?v=v1COuU2vU_w

Uninitialized: Can Be Different Each `read`

In C, `read` from `uninitialized` is `unspecified` and allowed to be ***different each time*** it is read

Important Benefit:
Allows behavior such as Lazy Recycling of pages

```
...
int* ptr = (int*)malloc(sizeof(int));
//'ptr' references allocated-but-not-initialized object
for(int i = 0; i < 10; ++i)
{
  std::cout << *ptr << "\n";//Can be different each time!
}
```



Superposition:
Simultaneously holding all valid states

C Is Not a Low-level Language

Your computer is not a fast PDP-11.

David Chisnall

https://queue.acm.org/detail.cfm?id=3212479

Unknowable State

That which cannot be inspected

C++ now

What Is “State”?

State: An instance of a type with a given *value*

- “Smells” a lot like an “object” – because it is!
- Can be ***small***:

```
bool b = true;
```

What Is "State"?

State: An instance of a type with a given *value*

- "Smells" a lot like an "object" – because it is!
- Can be ***small***:

```
bool b = true;
```

- Can be ***big***:

```
ChildProcess child = {...};
```

State can be (explicitly or implicitly):
- *all **memory** within this process*
- *all **objects** within this process*
- *all **dependencies / connections** to other processes / resources*
- *all **current context(s)** for threads, stacks, resources*
- *all **meta-information**, such as associated CPU ID, current system-load, current competition with other nodes / resources*

What Is "State"?

State: An instance of a type with a given *value*

- "Smells" a lot like an "object" – because it is!
- Can be ***small***:

```
bool b = true;
```

- Can be ***big***:

```
ChildProcess child = {...};
```

- Can be ***very big***:

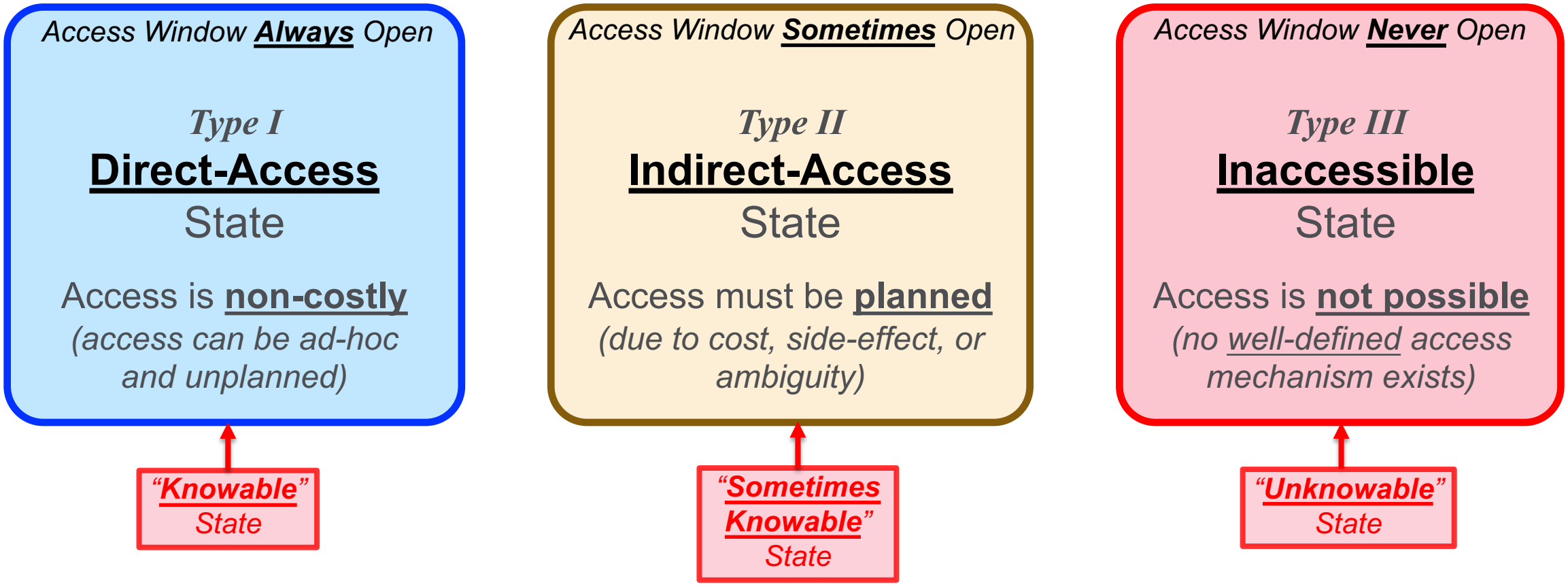
```
NodeCluster cluster     = {...};
DataCenter  data_center = {...};
Internet    i           = {...};
```

State can be (explicitly or implicitly):
- *all **memory** within this process*
- *all **objects** within this process*
- *all **dependencies / connections** to other processes / resources*
- *all **current context(s)** for threads, stacks, resources*
- *all **meta-information**, such as associated CPU ID, current system-load, current competition with other nodes / resources*

Can pack as many values and dependencies into an object, as-is ***feasible***

This is the scale promised with Quantum Computing

Categories Of State



- Access Category is **defined by the "Observer"**
  - *(Example):* An observer may have *Direct-Access* to a data object, while that data object is *Inaccessible* by a different observer

Directly-Accessible State

*Access Window **Always** Open*

Type I

Direct-Access

State

Access is **non-costly**

(access can be ad-hoc and unplanned)

*"**Knowable**" State*

*'**i**' is **Directly-Accessible***

```
int i;
void f();
// Assume single-threaded (i.e., "non-racey")
int main() {
  f(); // can access 'i'
  return 0;
}
```

- Access to '**i**' is cheap-and-direct; no planning is required
  - *(i.e., no contention / data-race, no runtime overhead)*

Global data object may have a cognitive (maintenance) cost

Indirectly-Accessible State

- Explicitly planned access is required when:
  1. Access is **costly** or **ambiguous** *(examples):*
     - **Resource contention** *(i.e., possible performance impact or deadlock)*
     - **Lazy-compute** *(must shift access for when work is to be done)*
     - **Volatile state** *(e.g., may be untrusted outside well-defined contexts)*
  2. Access is **destructive** *(examples):*
     - Access **causes side-effect** *(irreversible system mutation or behavior)*
     - State is **superpositioned** *(access triggers wavefunction collapse: quantum systems, or quantum data structure in Classical systems)*

*Access Window **Sometimes** Open*

Type II

Indirect-Access

State

Access must be **planned**

(due to cost, side-effect, or ambiguity)

*"**Sometimes Knowable**" State*

```
NuclearReactor nr;
nr.startup();
if(nr.isUp()) {
  // can now access core temperature
  ...
}
```

*Only starting here can access '**core_temp**' in '**nr**'*

Quantum systems *(data structures)* fundamentally operate through **destructive "read"**

*The "**Observer Effect**" upon "**Superpositioned**" state*

Inaccessible State

- The State exists; but cannot be accessed
  1. Access is **not possible** *(no access mechanism exists)*
  2. Access is **`undefined`** *(any resulting "value" is unusable)*

```
#include <iostream>
#include <thread>
int main(){
  std::thread t0([](){
    int num_recs = 0;
    while(processNextRec()) {
      ++num_recs;
      ...
    }
  });
  // cannot access 'num_recs' in 't0' !!
  ...
  t0.join();
  return 0;
}
```

Scoped within this lambda, within this thread-object

No access to '`num_recs`' inside object '`t0`'

Usable, but Unknowable!
'`num_recs`' is being "used", but is "unknowable"!

*Access Window **Never** Open*

Type III
Inaccessible
State

Access is **not possible**
(no well-defined access mechanism exists)

*"**Unknowable**"*
State

C++: Any access resulting in **`undefined`** behavior *(by definition)* reflects ***unknowable state***

C++ Examples (**`undefined`**): *Accessing an object…*

C++11 Object Model
- **Before object lifetime** begins
- **After object lifetime** ends
- When object **invariants are not satisfied** *(e.g., "partially-constructed", in-progress mutation or transaction, etc.)*

C++11 Memory Model
- **In "Racey" manner** *(e.g., without synchronization across threads)*

Orthogonal: Unusable, Unknowable

Orthogonal Concepts *(unrelated):*

- **Knowable / Unknowable** State – the ability to **inspect** a value
- **Usable / Unusable** State – the ability to **productively employ** a value

Usable but **Unknowable** State

Common Pattern *(quantum systems, quantum data structures):*

1. State is **being used** *(the system relies upon it)*
2. State is **highly coupled** / entangled *("superpositioned")*
3. State **cannot be inspected** *('`read`' would be destructive)*

Quantum systems and
quantum data structures use the state
(the state exists), but **do not observe** the state
(that would be destructive)

Implications: *Quantum Data Objects…*

1. *Cannot be observed for **conditional processing***
2. *Cannot be observed for **debugging***
3. *Cannot **be classified** (such as to perform "grouping" based on their attributes or internal state)*

Quantization

Counts Of Whole Quantum

C++ now

*There **exists** a "**Unified Theory**"*
For All Things "Software"

The Quantum

Quanta *(noun):* ***plural*** of quantum

Quantum *(noun), commonly defined:*

- The smallest amount of a physical quantity that can exist independently
- A discrete, indivisible manifestation of a physical property
- Is regarded as a unit

Quantum *(noun):* the ***discrete resolution-value*** within a data structure that enables required type invariants

Discrete!
(Individual value can exist)

- Software Engineers do ***quantization*** *All Day, Every Day!*
  1. Data **type selection**: Required ***range***, ***resolution***, use of ***"trap values"***
  2. New **type creation** *(define new interpretation of existing data objects)*
  3. New **type composition** *(compose higher-order aggregates with new rules or context for interpretation)*

- Examples *(what is the quantum?):*

```
int       i;
```

The Quantum

Quantum *(noun), commonly defined:*

- The smallest amount of a physical quantity that can exist independently
- A discrete, indivisible manifestation of a physical property
- Is regarded as a unit

Quanta *(noun):* ***plural*** of quantum

Quantum *(noun):* the ***discrete resolution-value*** within a data structure that enables required type invariants

Discrete!
(Individual value can exist)

- Software Engineers do ***quantization*** *All Day, Every Day!*
  1. Data **type selection**: Required ***range***, ***resolution***, use of ***"trap values"***
  2. New **type creation** *(define new interpretation of existing data objects)*
  3. New **type composition** *(compose higher-order aggregates with new rules or context for interpretation)*

- Examples *(what is the quantum?):*

```
int          i;
```

Quantum: 1 (whole integer)

The Quantum

Quantum *(noun), commonly defined:*

- The smallest amount of a physical quantity that can exist independently
- A discrete, indivisible manifestation of a physical property
- Is regarded as a unit

Quanta *(noun):* ***plural*** of quantum

Quantum *(noun):* the ***discrete resolution-value*** within a data structure that enables required type invariants

Discrete!
(Individual value can exist)

- Software Engineers do ***quantization*** *All Day, Every Day!*
  1. Data **type selection**: Required ***range***, ***resolution***, use of ***"trap values"***
  2. New **type creation** *(define new interpretation of existing data objects)*
  3. New **type composition** *(compose higher-order aggregates with new rules or context for interpretation)*

- Examples *(what is the quantum?):*

```
int      i;
float    f;
```

Quantum: 1 (whole integer)

The Quantum

Quantum *(noun), commonly defined:*

- The smallest amount of a physical quantity that can exist independently
- A discrete, indivisible manifestation of a physical property
- Is regarded as a unit

Quanta *(noun):* ***plural*** of quantum

Quantum *(noun):* the ***discrete resolution-value*** within a data structure that enables required type invariants

Discrete!
(Individual value can exist)

- Software Engineers do ***quantization*** *All Day, Every Day!*
  1. Data **type selection**: Required ***range***, ***resolution***, use of ***"trap values"***
  2. New **type creation** *(define new interpretation of existing data objects)*
  3. New **type composition** *(compose higher-order aggregates with new rules or context for interpretation)*

- Examples *(what is the quantum?):*

```
int     i;
float   f;
```

Quantum: `1` *(whole integer)*

Quantum: ***Machine epsilon***
`std::numeric_limits<float>::epsilon()`
Warning: *Comparing floats is tricky, due to* ***disagreement over the quantum***

The Quantum

Quantum *(noun), commonly defined:*

- The smallest amount of a physical quantity that can exist independently
- A discrete, indivisible manifestation of a physical property
- Is regarded as a unit

Quanta *(noun):* ***plural*** of quantum

Quantum *(noun):* the ***discrete resolution-value*** within a data structure that enables required type invariants

Discrete!
(Individual value can exist)

- Software Engineers do ***quantization*** *All Day, Every Day!*
  1. Data **type selection**: Required ***range***, ***resolution***, use of ***"trap values"***
  2. New **type creation** *(define new interpretation of existing data objects)*
  3. New **type composition** *(compose higher-order aggregates with new rules or context for interpretation)*

- Examples *(what is the quantum?):*

```
int     i;
float   f;
```

Quantum: 1 (whole integer)

Quantum: ***Machine epsilon***
`std::numeric_limits<float>::epsilon()`
Warning: *Comparing floats is tricky, due to* ***disagreement over the quantum***

```
class DayOfWeek {
 public:
  enum class Day {
    Sun, Mon, Tue, Wed,
    Thu, Fri, Sat };
  Day   d_;
};
```

The Quantum

Quantum *(noun), commonly defined:*
- The smallest amount of a physical quantity that can exist independently
- A discrete, indivisible manifestation of a physical property
- Is regarded as a unit

Quanta *(noun):* ***plural*** of quantum

Quantum *(noun):* the ***discrete resolution-value*** within a data structure that enables required type invariants

Discrete!
(Individual value can exist)

- Software Engineers do ***quantization*** *All Day, Every Day!*
  1. Data **type selection**: Required ***range***, ***resolution***, use of ***"trap values"***
  2. New **type creation** *(define new interpretation of existing data objects)*
  3. New **type composition** *(compose higher-order aggregates with new rules or context for interpretation)*

- Examples *(what is the quantum?):*

```
int      i;
float    f;
```

Quantum: `1` (whole integer)

Quantum: ***Machine epsilon***
`std::numeric_limits<float>::epsilon()`
Warning: *Comparing floats is tricky, due to* ***disagreement over the quantum***

```
class DayOfWeek {
 public:
  enum class Day {
    Sun, Mon, Tue, Wed,
    Thu, Fri, Sat };
  Day    d_;
};
```

Quantum: `DayOfWeek::Day`

Quantization

Quantize *(verb):*
1. **To constrain to a discrete set** of values
2. **To count whole units** of quantum

Quantization restricts any measure to a valid value within a **discrete set** of possible values

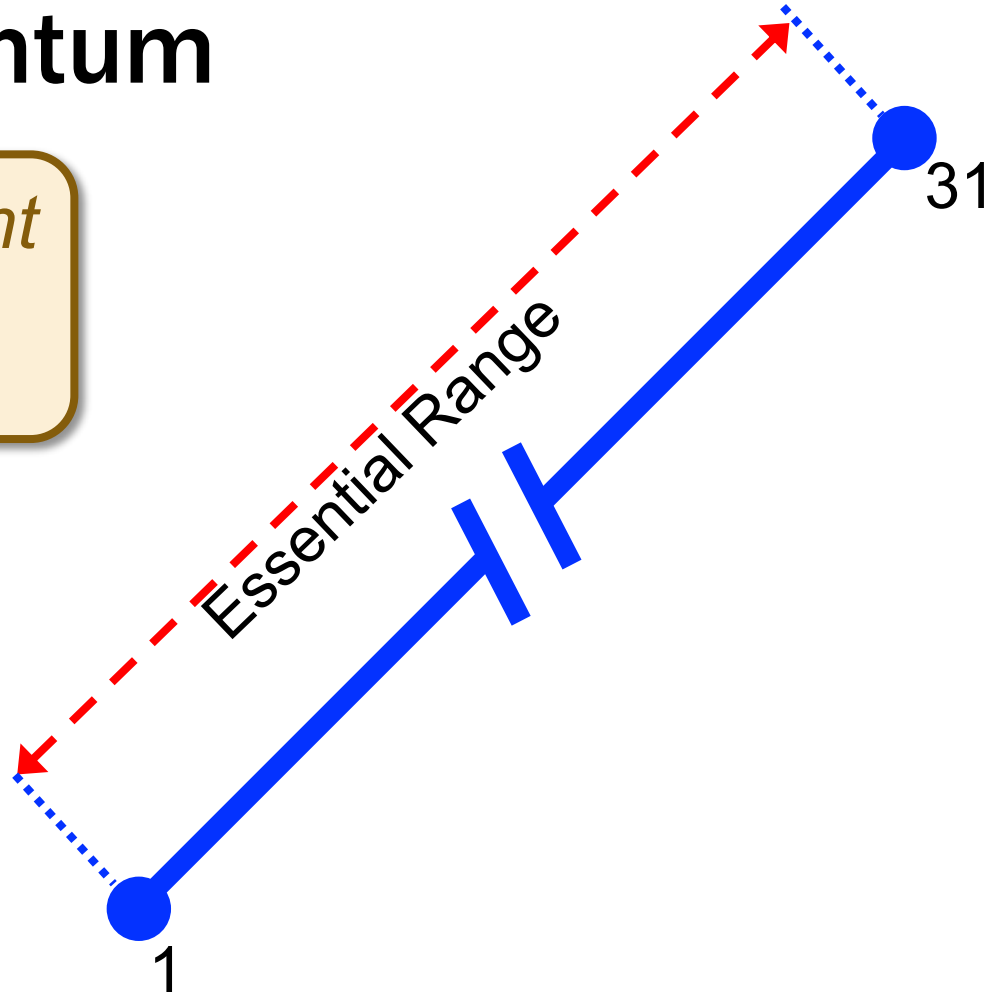
- In Software:
  - ***Defining a Type*** is synonymous with ***defining rules for quantization***
  - ***Computing State*** is synonymous with ***performing quantization***

*"**Assigning**" state …*
*"**Computing**" state …*
*"**Hard-Coding**" state …*

Defining A Quantum

"We need to represent ***day-of-month*** *"*
(Gregorian Calendar)

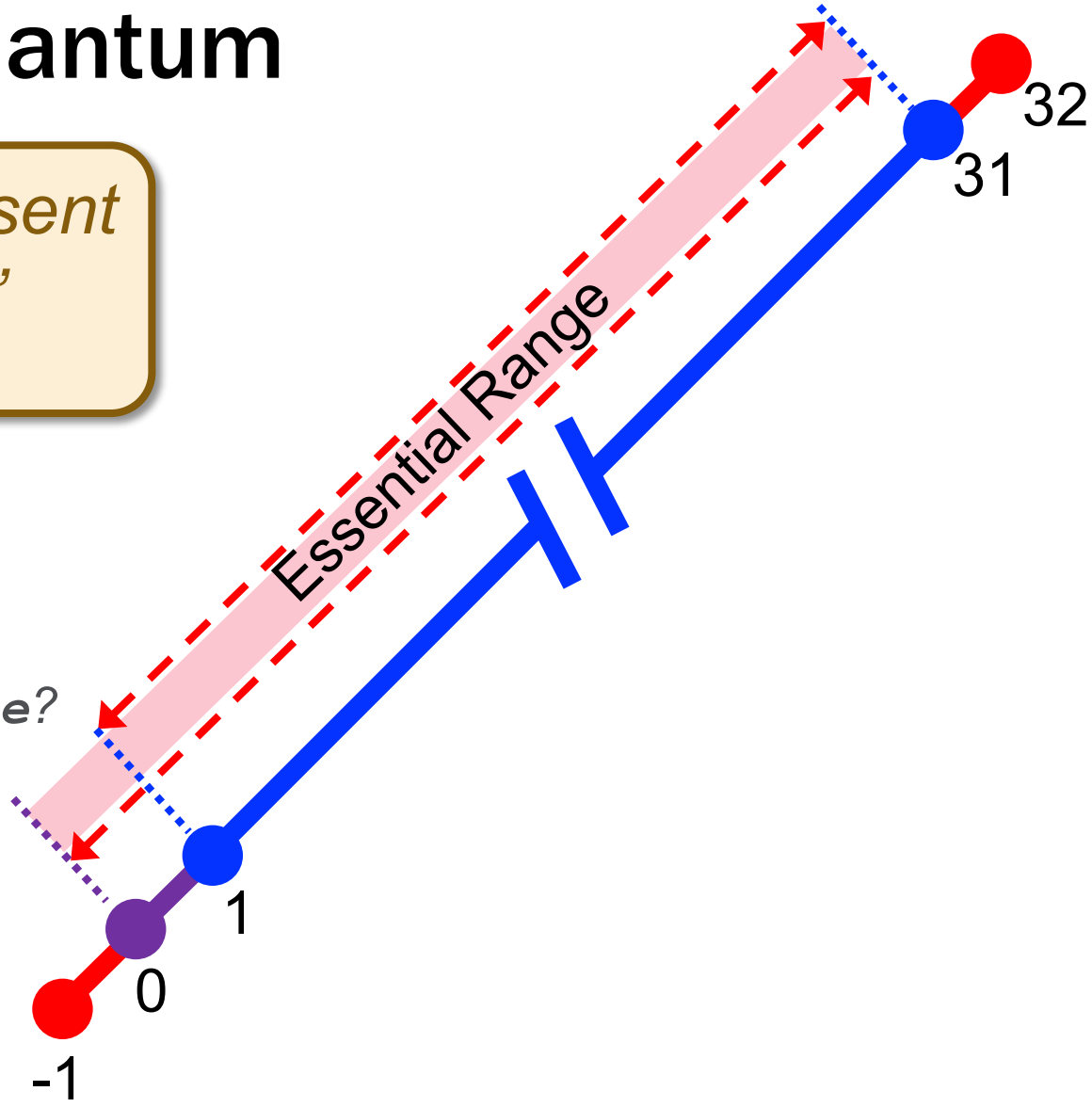
- Requirements:



Defining A Quantum

"We need to represent ***day-of-month*** *"*
(Gregorian Calendar)

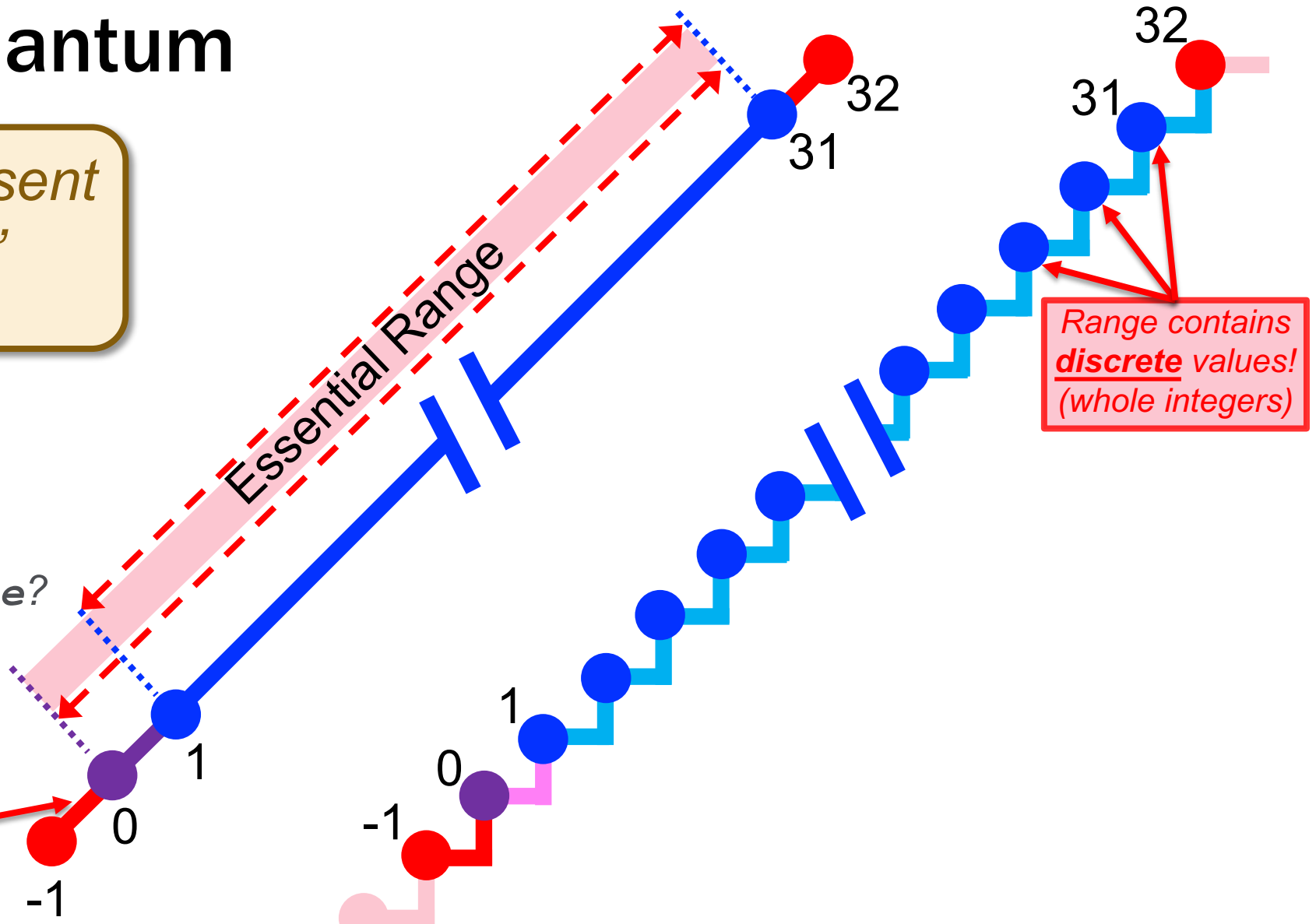
- Requirements:
  - Range: `[1,31]`
    - *...maybe* `[0,31]`
      - *where* `0==trap_value`*?*
  - Whole integer
  - Unsigned



Defining A Quantum

*"We need to represent **day-of-month**"*
(Gregorian Calendar)

- Requirements:
  - Range: `[1,31]`
    - *...maybe* `[0,31]`
      - *where* `0==trap_value`*?*
  - Whole integer
  - Unsigned

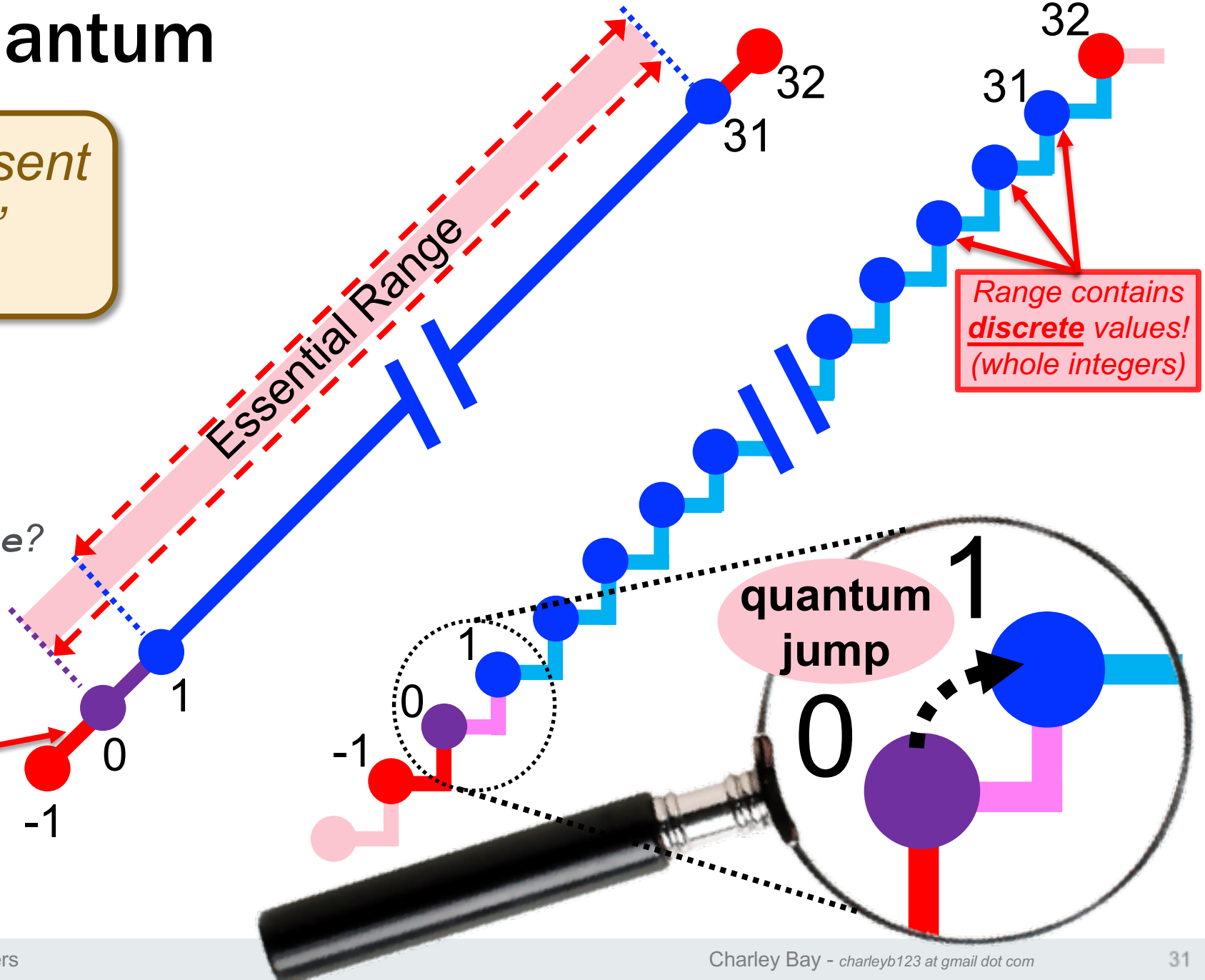


Defining A Quantum

*"We need to represent **day-of-month**"*
(Gregorian Calendar)

- Requirements:
  - Range: `[1,31]`
    - *...maybe* `[0,31]`
      - *where* `0==trap_value`*?*
  - Whole integer
  - Unsigned

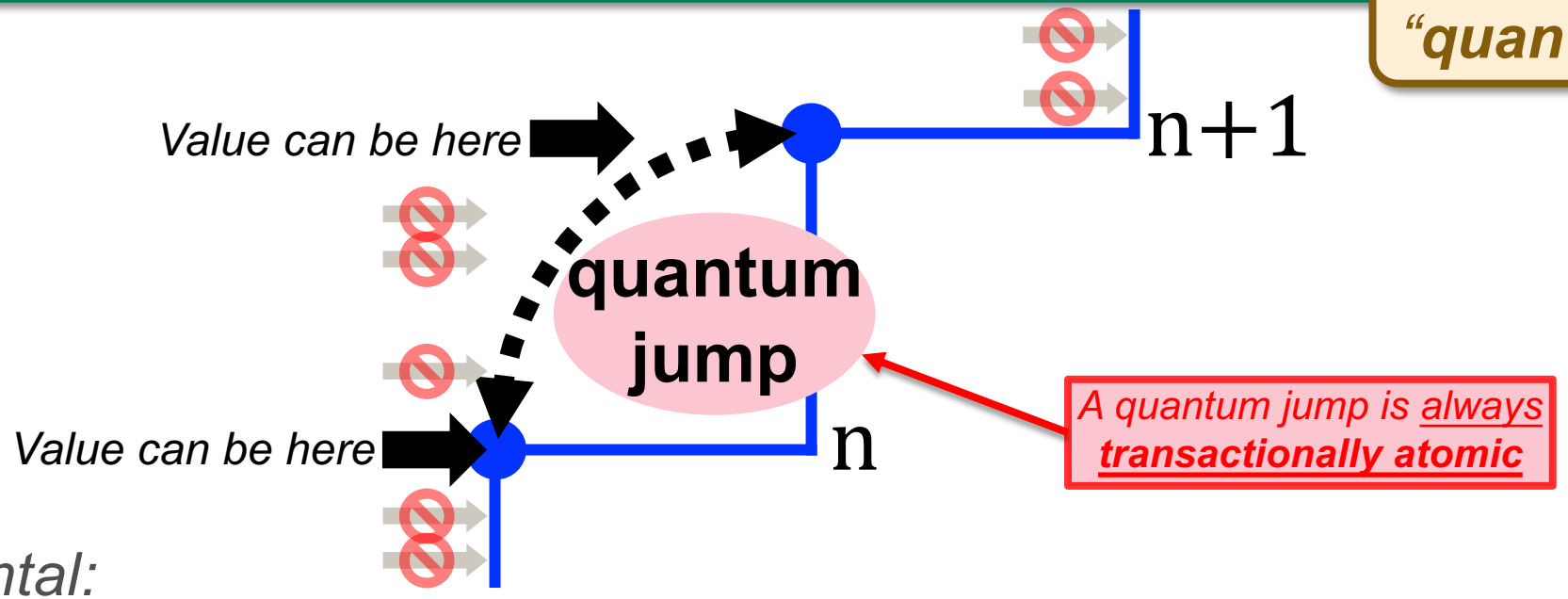
Misleading!
*Values in range are **not continous**!*



Quantum Jump

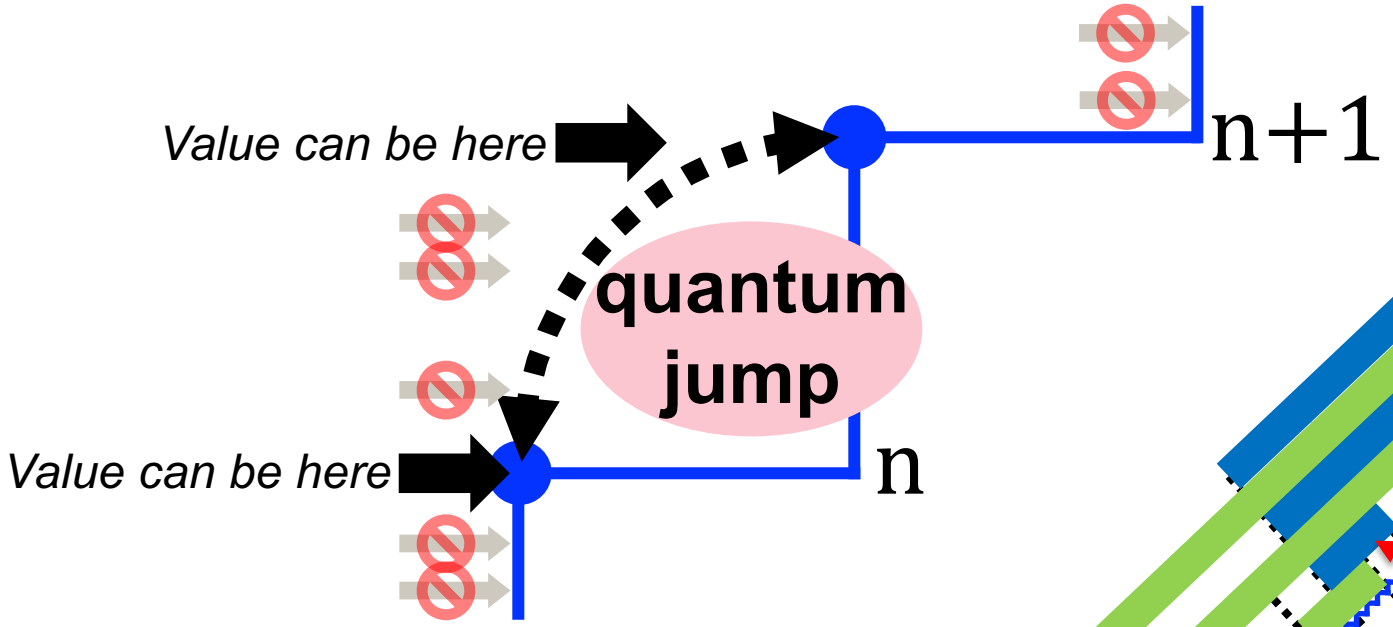
Quantum Jump *(noun, verb):* the ***instantaneous*** transition to a new quantum value ***without the intermediate transition*** through other values

*Also called, "**quantum leap**"*



- *Fundamental:*
  - Quantum Physics **relies upon transitions through quantum jumps**
  - A **quantum data structure** performs *(transactional atomic)* **quantum jumps**

Commonly Used Quantum: Integer



- **Integer** *(whole number)*
- **Ranges** Commonly Used

4,294,967,295
2,147,483,647
unsigned long
65,535
unsigned int
32,767
unsigned short
255
127
unsigned char
0
-128
char
-32,768
short
int
long
-2,147,483,648

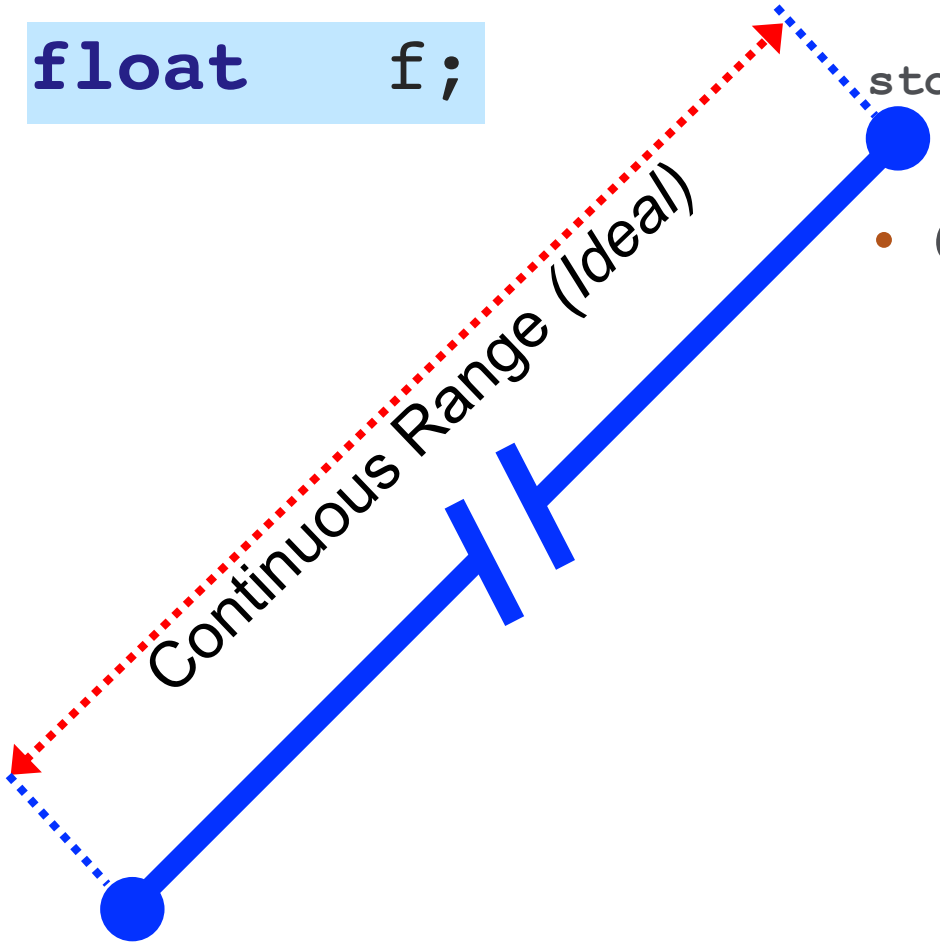
Integer is popular because it has a "**normalized" quantum**!

Well-understood agreement in hardware and software!

What About A “Continuous Range”?

- When whole integers are “too coarse”, we use **floating-point resolution**

```
float    f;
```

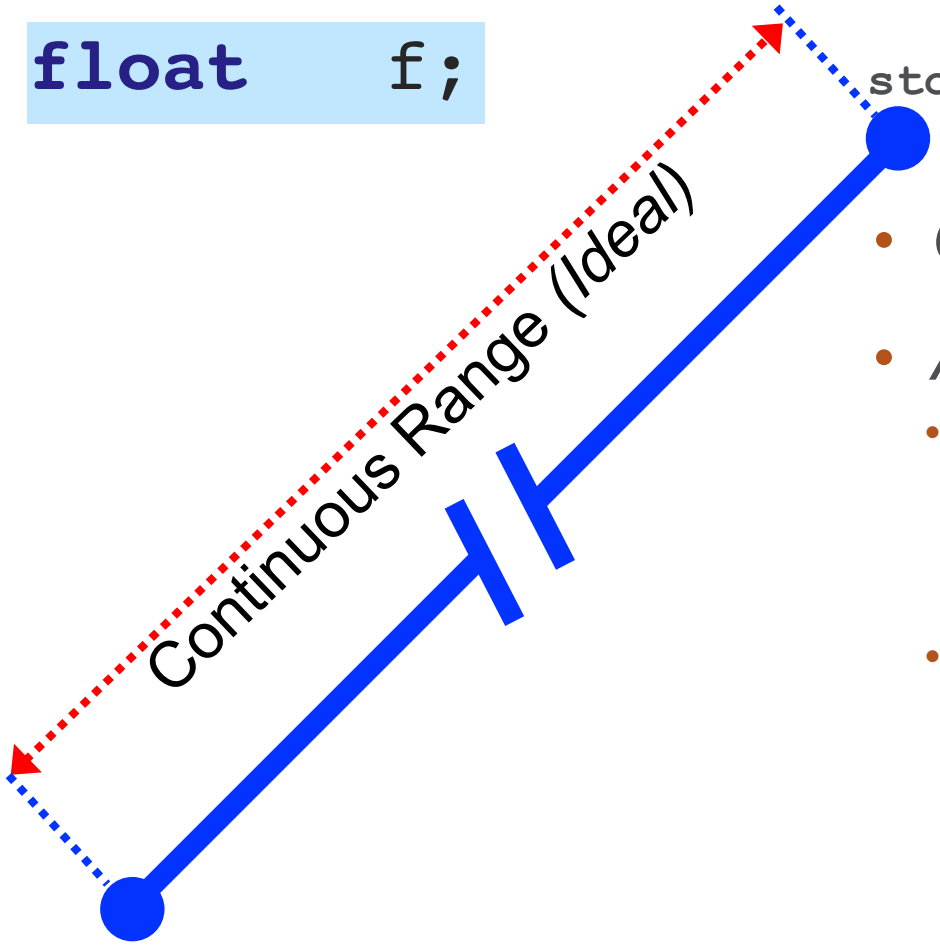


- *Q:* What is the *quantum*?

What About A “Continuous Range”?

- When whole integers are “too coarse”, we use **floating-point resolution**

```
float    f;
```



`std::numeric_limits<float>::max()`

`std::numeric_limits<float>::min()`

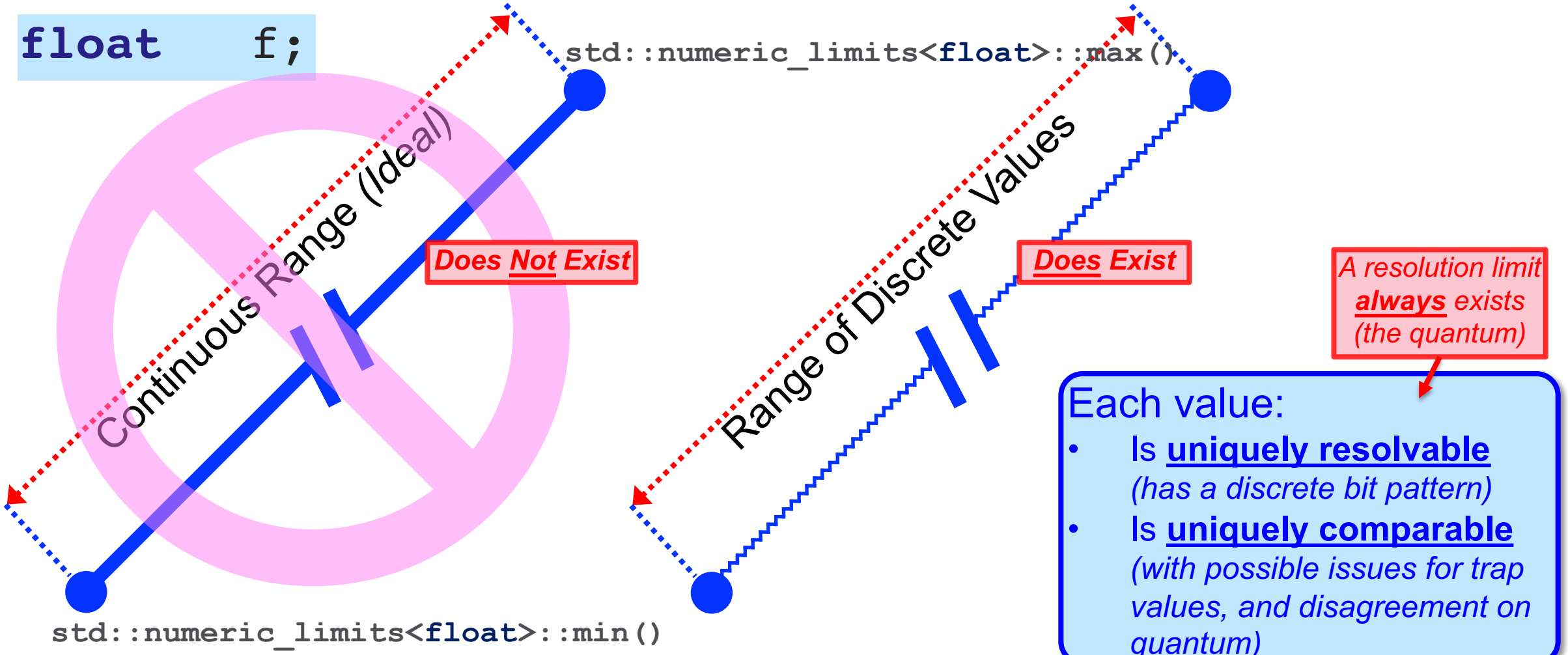
- *Q:* What is the *quantum*?
- *A:* It *depends*!
  - `std::numeric_limits<float>::epsilon()`
    - Returns the ***difference between 1.0 and the next representable value***
  - *Different* quantum:
    - `[0.0 … +1.0]`
    - `[-1.0 … +1.0]`
    - `[-2,000,000,000 … +2,000,000,000]`
    - `[-inf … +inf]`

A resolution *limit* **always** exists *(the quantum)*

Warning: *Comparing* `floats` *is tricky, due to* ***disagreement over the quantum***

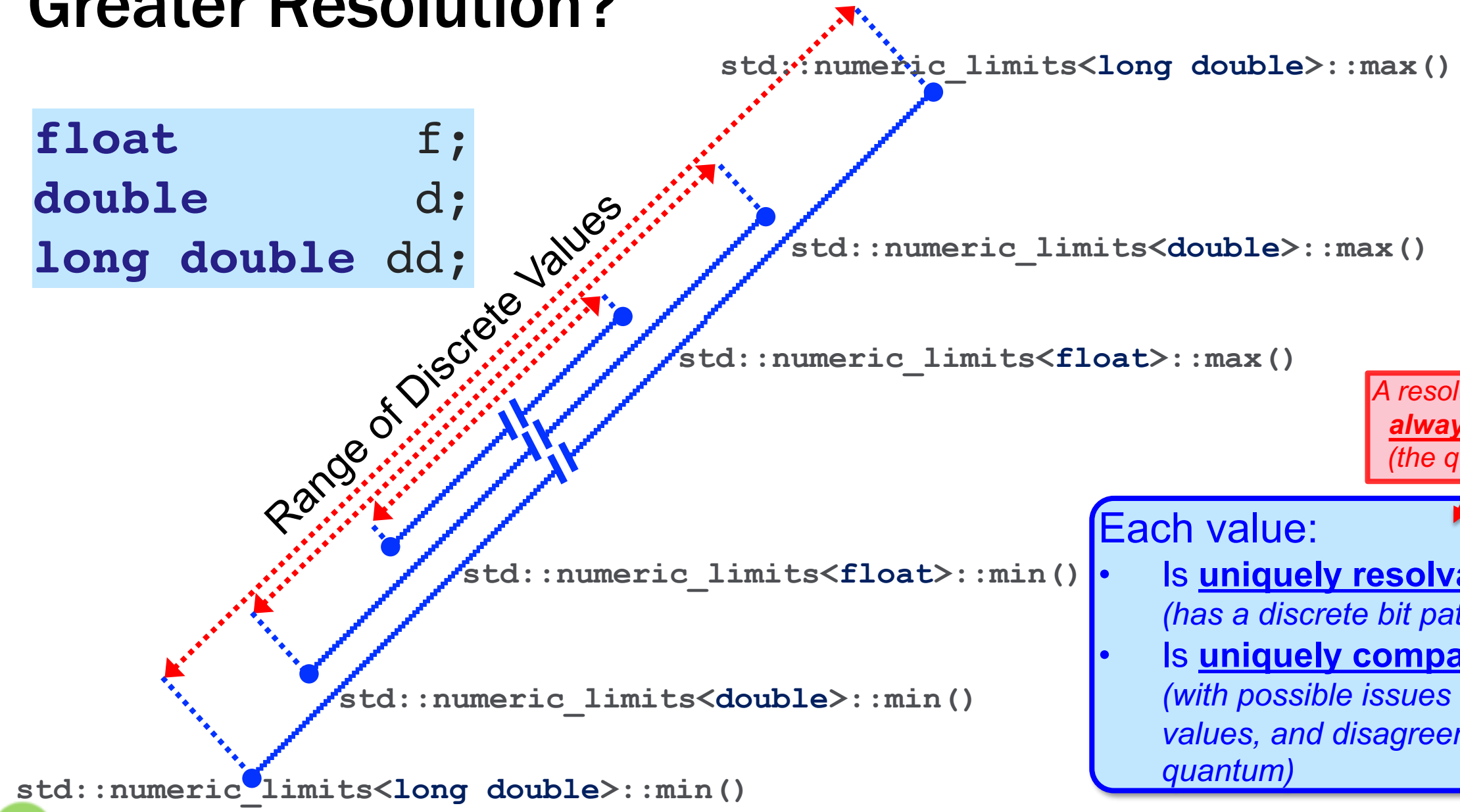
A “Continuous Range” Does Not Exist

- **Floating-point** resolution produces **discrete values within a range**



Greater Resolution?

```
float       f;
double      d;
long double dd;
```



*A resolution limit **always** exists (the quantum)*

Each value:

- Is **uniquely resolvable** *(has a discrete bit pattern)*
- Is **uniquely comparable** *(with possible issues for trap values, and disagreement on quantum)*

Take-Away:

A range of ***Continuous*** values does *not* exist

A range of ***Discrete*** values *does* exist

*(all ranges are composed of **discrete** values)*

Data Objects Are Always Quantized

- Data **objects are always quantized**:
  - A "Value" always exists
  - "Meaning" may be missing *(e.g., "uninitialized", or lacking units or context)*
- **Value** of a Data Object is ***(always!)* discrete**
  - **Is physically represented** within the system *(for Classical systems)*
  - **Is discretely represented** by a *(reproducible)* bit-pattern *(for Classical systems)*

An **illusion** of non-discrete is possible:
- When data object resolution is sufficiently "low"
- When disagreement exists regarding the quantum

*Similar to "**type-punning**":*
Two `float` objects with different ranges can be compared (but disagree on the quantum)

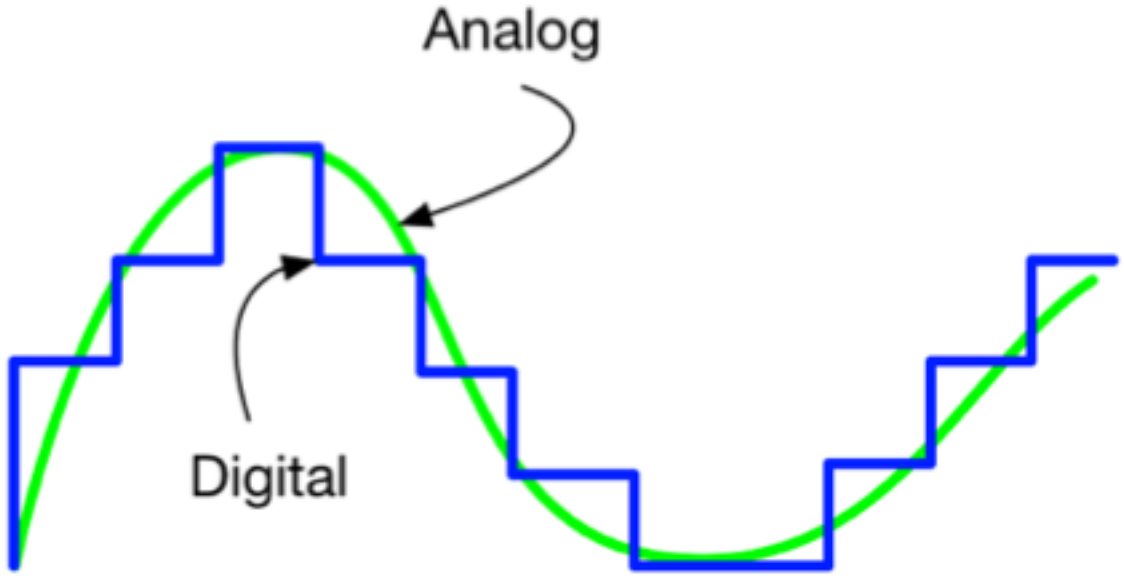
The “Natural” World Is Quantum

High Resolution (But NOT Continuous)

Analog vs. Digital World

Analog: ***continuous range*** of values

Digital: ***discrete set*** of values



- Can **measure using "analog"** technologies:
  - *(example):* Film Camera performs chemical capture of light exposure *(resulting in photographs)*
  - Has a ***finite resolution***: 36mm x 24mm frame of ISO 100-speed film was initially estimated to contain ~20 Megapixels
- Can **measure using "digital"** technologies: ← ***Digitize***: *Convert to digital format*
  - *(example):* Digital Camera uses (CCD or CMOS) sensors *(resulting in digital images)*
  - Has a ***finite resolution***: ~20 Megapixels in commodity devices *(in 2018)*

"Measure" == "Describe"

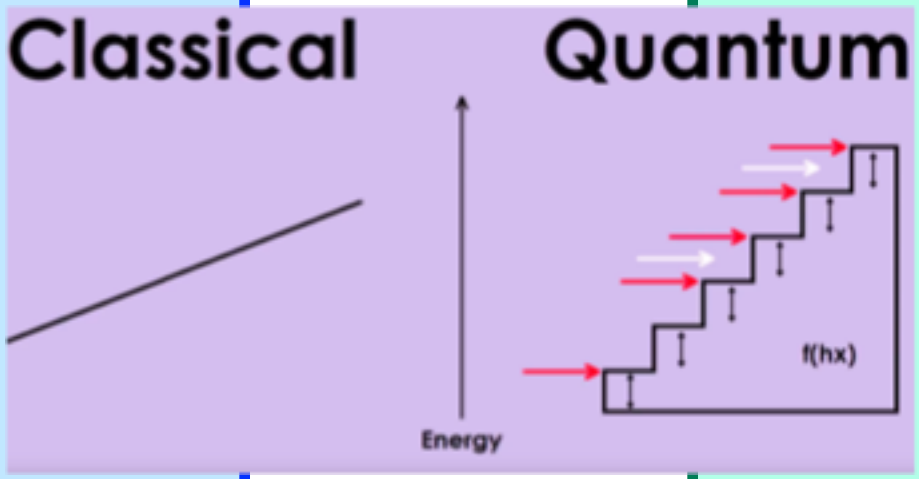
Resolution ***limits exist*** and ***are measurable***; and vary based on technology type

Physics

Classical Physics

*Proven **NOT** true!*
*(Mathematically **inconsistent** and **not-predictive** with observed reality)*

Assumed
Values in range are
Continuous



(image from):
Quantum Theory Made Easy
https://www.youtube.com/watch?v=e5_V78SWGF0

Quantum Physics

*Demonstrated to be **True**!*
*(Mathematically **consistent** and **predictive** with observed reality)*

Recognizes
Values in range are
Discrete

Quantum Physics

Quantum: Electromagnetic energy *exists only in quantized form*

Work done in 1900

Max Karl Ernst Ludwig Planck

23-Apr-1858 – 04-Oct-1947

Demonstrated: $E = hf$

- E – energy
- h – Planck's constant
- f – frequency

- *"Planck-Einstein relation"*
- *"Einstein relation"*
- *"Planck's energy-frequency relation"*
- *"Planck relation"*
- *"Planck equation"*
- *"Planck formula"*

- Describes the **quantized nature of Electromagnetic radiation** *(light)*
- Energy exists in **discrete units** *(the quantum)*
- **Quantum of action**: Planck's constant
  - Unit: (*energy * time*), or (*angular momentum*)
  - $6.626070040(81) \times 10^{-34}$ Js (Joule-seconds)

Not continuous: A fraction of a quantum *cannot* exist

Not Continuous: The "Natural" World Is Quantum

Planck equation: $E = hf$
- **E** – energy
- **h** – Planck's constant
- **f** – frequency

hf *is now called "**Photon**"*
- ***photon** of frequency **f** will have specific energy **E***

Einstein postulated Planck's quanta were real particles (not math fiction) Year: 1905

Prior to Planck's Work	• ***Assumed Continuous***: It was *assumed* that energy could take on any value • ***"Utraviolet catastrophe"*** – Math could not account for observed behavior *(of black body radiation)*
After Planck's Work	• ***Discrete***: Energy exists only as a whole count of individual quantum

Modern C++ began with **C++11**
(Prior is known as Classical C++)

"Post-Newtonian" *Physics*

Modern Physics began with **Quantum Physics**
(Prior is known as Classical Physics)

Photoelectric Effect

Photoelectric Effect: The transfer of energy from a *photon* to an *electron*

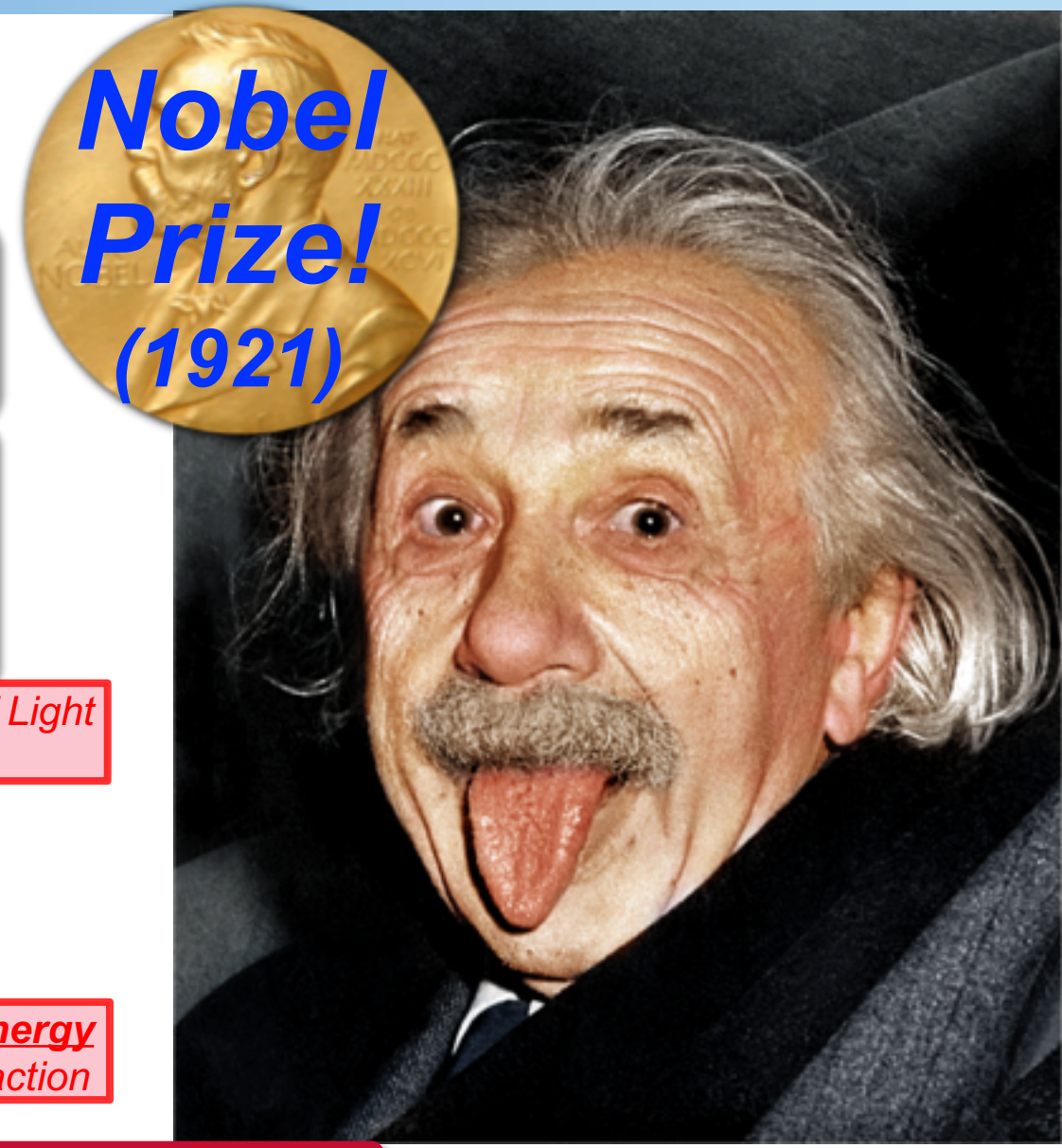
Demonstrated: An electron is dislodged from a metal plate when hit by a photon of sufficient energy; or ***no*** interaction occurs.

Wave-Particle Duality of Light
Year: 1905

- Light travels in **discrete packets** *(photon)*
- Energy interactions are **discrete based on the *Work Function*** (φ)

*Minimum **Threshold Energy** required to cause interaction*

Not continuous: A photon dislodges an electron, or does *not*



Albert Einstein
14-Mar-1879 – 18-Apr-1955

Bremsstraahlung

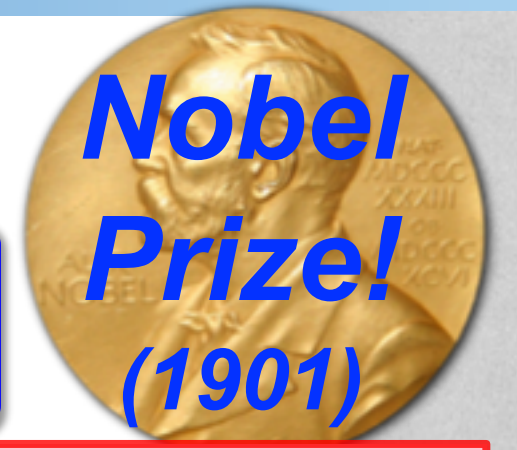
Bremsstraahlung: The transfer of energy from an *electron* to a *photon*

Demonstrated: "**Braking Radiation**": Particle collides with another particle, and emits photon (i.e., "radiation")

Bremsstraahlung: (German)
- *Bremsen "to brake"*
- *Strahlung "radiation"*

- Describes **kinetic energy conversion to a photon emission**, including "perfect" conversion
- Is the **"opposite" of the Photoelectric Effect**

Not continuous: A particle collision emits a photon, or does *not*



Nobel Prize! (1901)

*The **first recipient** of the **Nobel Prize in Physics** (awarded for producing and detecting X-rays)*

Wilhelm Conrad Röntgen

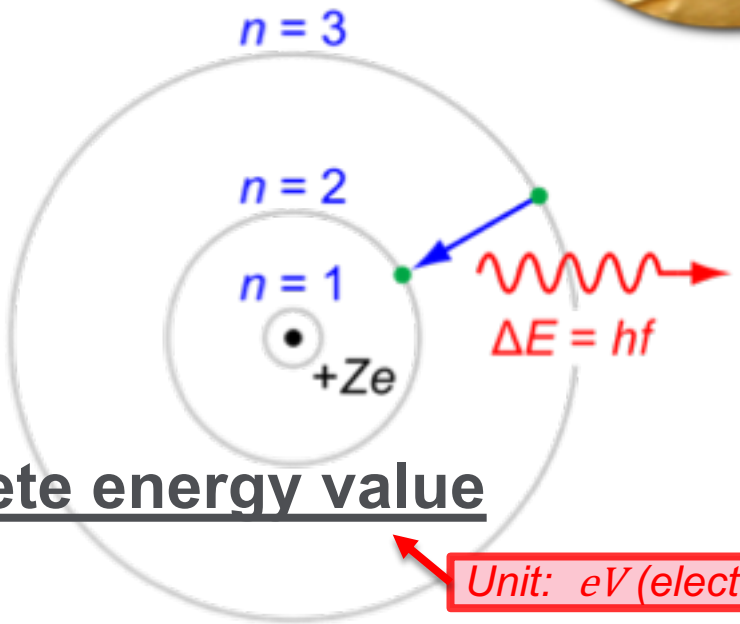
27-Mar-1845 – 10-Feb-1923

The Atomic Model

Bohr Model: The first *quantized model* of the atom

Demonstrated: An electron drops to a lower orbital and **emits a photon** of discrete frequency

Did not (yet) make accurate predictions for multi-electron atoms

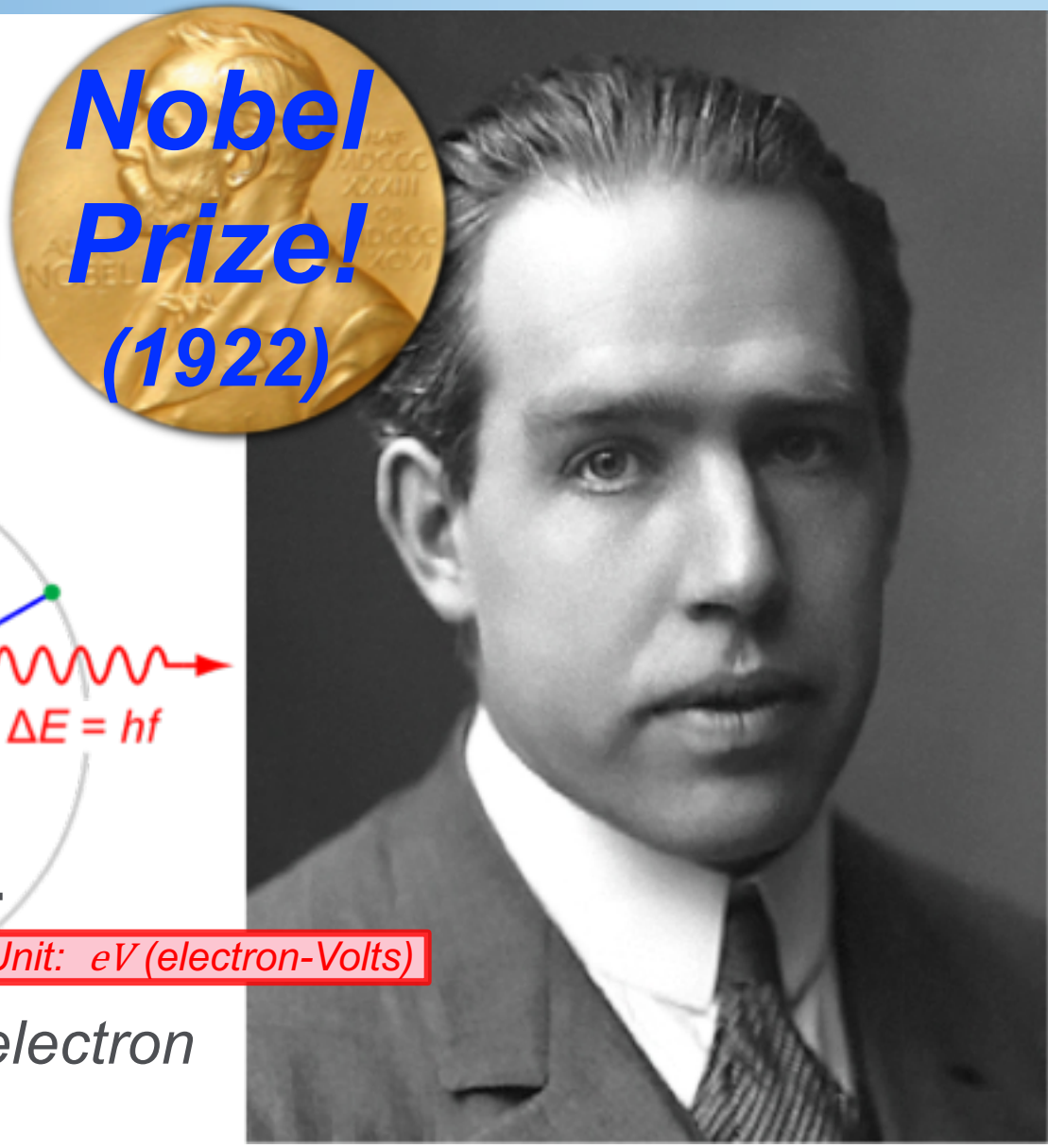


- An electron only has a **discrete energy value** *(defined by its orbital)*

Unit: eV (electron-Volts)

- Photon is **emitted with a discrete value** *(as electron drops orbital)*

Not continuous: Electrons exist in discrete orbitals



Niels Bohr

7-Oct-1885 – 18-Nov-1962

X-ray Spectroscopy

Electron Shell: Defines principal *energy level as orbit* around an atom's nucleus

Demonstrated: X-ray emission spectra corresponds to electron transition dropping to lower orbital *(releasing photon)*

Accurate predictions for multi-electron atoms!

- **Elements** exhibit unique **spectral lines** *(based on photon emitted as electron transitions to lower orbital)*
- Element "fingerprint" is **basis for spectroscopy** *(because photon emission is **discrete** for that element)*

Not continuous: An electron drops to a lower orbital, or does *not*

Karl Manne Sigbahn

3-Dec-1866 – 26-Sep-1978

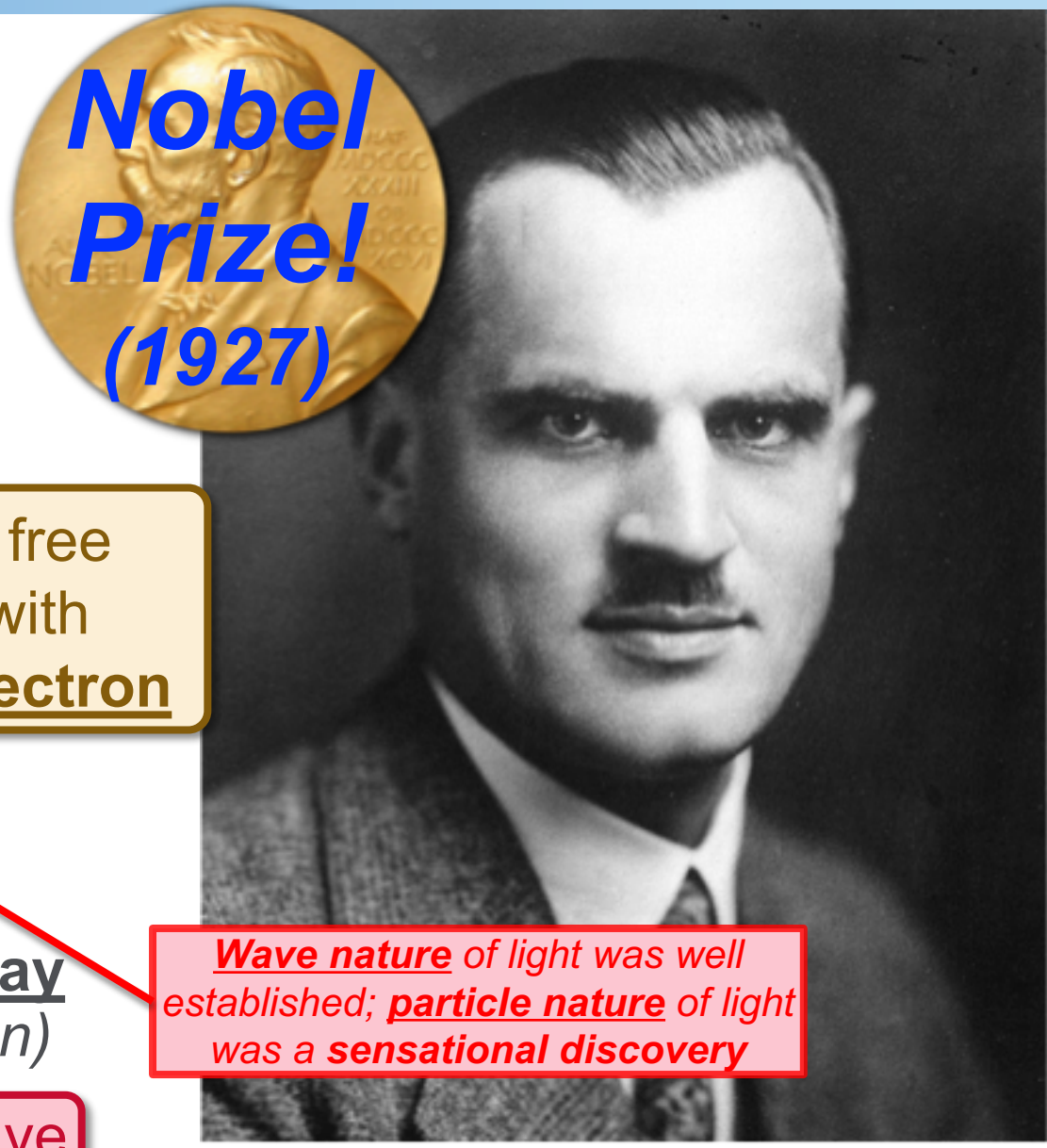
Compton Scattering

Compton Effect: Demonstrates the particle nature of electromagnetic radiation *(sensational at the time)*

Demonstrated: X-ray quantum scattered by a free electron had longer wavelength *(less energy)*, with surplus **energy quantum transferred to an electron**

- Gave rise to the **"wave-particle" duality**
- Mathematical relationship between **shift in wavelength and scattering angle of the X-ray** *(one photon interacts with exactly one electron)*

Not continuous: Light is demonstrated to behave as a *particle*



Wave nature *of light was well established;* ***particle nature*** *of light was a* ***sensational discovery***

Arthur Holly Compton

10-Sep-1892 – 15-Mar-1962

de Broglie hypothesis

All Matter Has Wave Properties: All matter must be transported by a wave into which it is incorporated

Demonstrated: Wave-Particle Duality is similarly exhibited by all particles of arbitrary mass, and to all laws of nature

- An electron has an "**internal clock**" used as a mechanism by which **a pilot wave guides a particle**
- The "**true mass**" of a particle **is not constant**

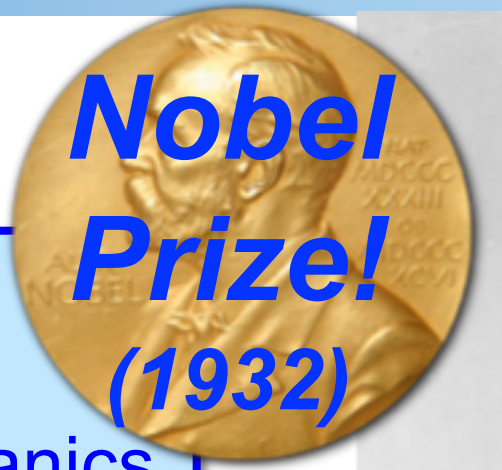
Not continuous: A particle of arbitrary mass is discretely guided by a wave function

The velocity of a particle always equals the group velocity of the corresponding wave

All matter exhibits wave-like behavior
(Can be similarly diffracted: beam of light, beam of electrons, wave of water…)

Louis de Broglie
15-Aug-1892 – 19-Mar-1987

Establish Quantum Mechanics



Nobel Prize! (1932)

Matrix mechanics: The first conceptually autonomous and logically consistent formulation of quantum mechanics

Demonstrated: Account for quantum jumps by interpreting physical particles as matrices that evolve in time

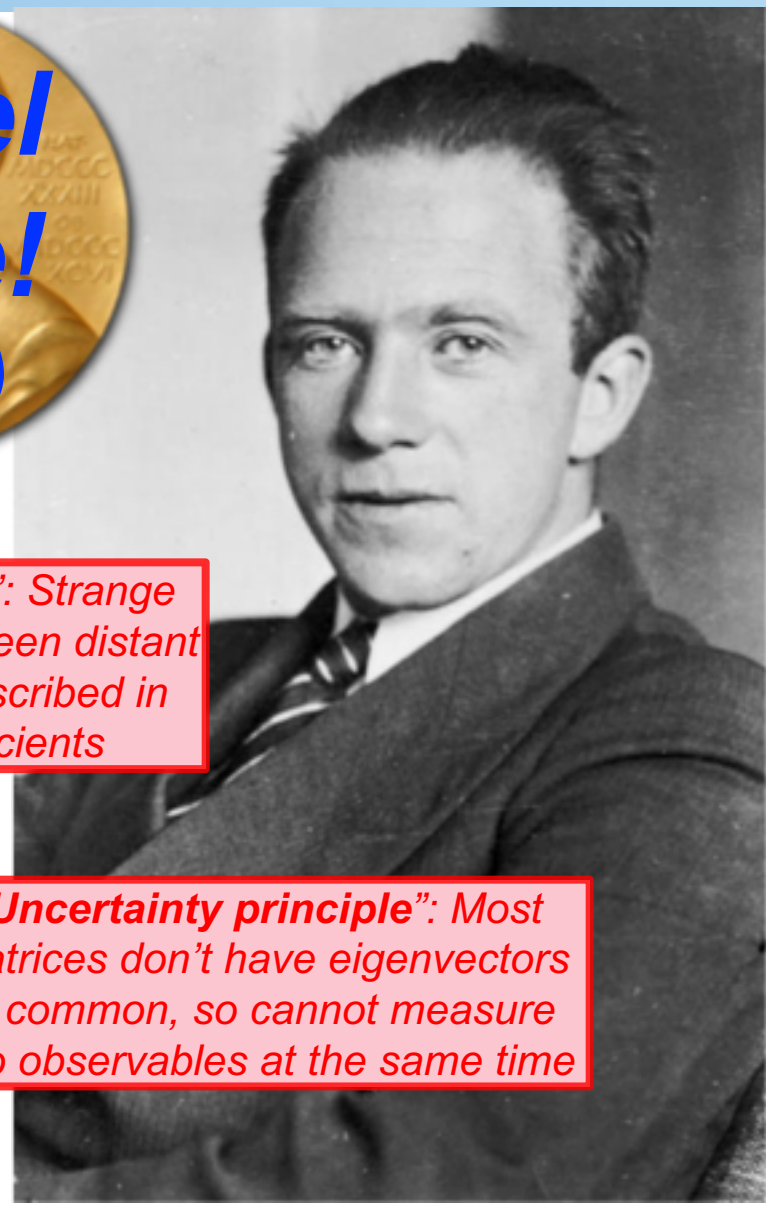
*"**Entanglement**": Strange correlations between distant particles, as described in matrix coefficients*

- **Introduced matrices math** to physics *(previously belonged to pure mathematics)*, credited as, "The Creation of Quantum Mechanics"

*"**Uncertainty principle**": Most matrices don't have eigenvectors in common, so cannot measure two observables at the same time*

- **Introduced Waveform Collapse**: In matrix mechanics, the **act of measurement *"collapses"* system state**

Not continuous: Quantum jumps occur, or do *not*



Werner Karl Heisenberg

05-Dec-1901 – 01-Feb-1976

Schrödinger equation

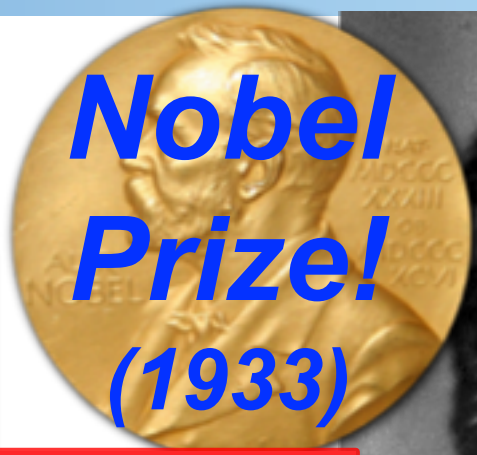
Schrödinger equation: Wave-equation can compute eigenvalues *independent of time*

Demonstrated: Measurements are quantized; but measurements correlate with uncertainty

*"**Characteristic values**", such as interpreting ionization potentials in molecular orbitals*

- Caused mathematical **shift to complex numbers** *(from real numbers)*
- Equation describes ***(deterministic)*** **particle evolution**, but yielding **probabilistic unknowns**

Not continuous: Measurements are discrete, and correlated with probabilities



"The father of quantum mechanics"

On Quantum Mechanics, quoted:
"I don't like it, and I'm sorry I had anything to do with it."

Erwin Rudolf Josef Alexander Schrödinger

12-Aug-1877 – 04-Jan-1961

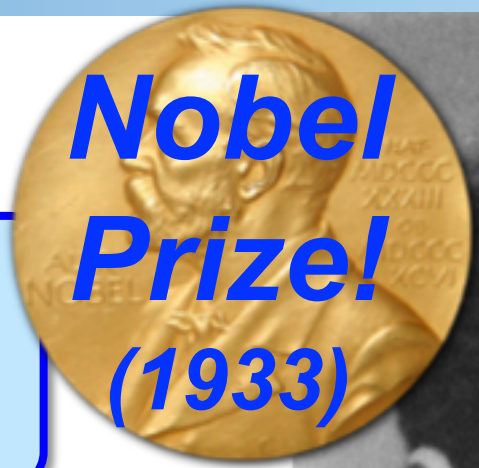
Dirac equation

Dirac equation: Describes behavior of fermions and predicted the existence of antimatter *(previously unsuspected)*

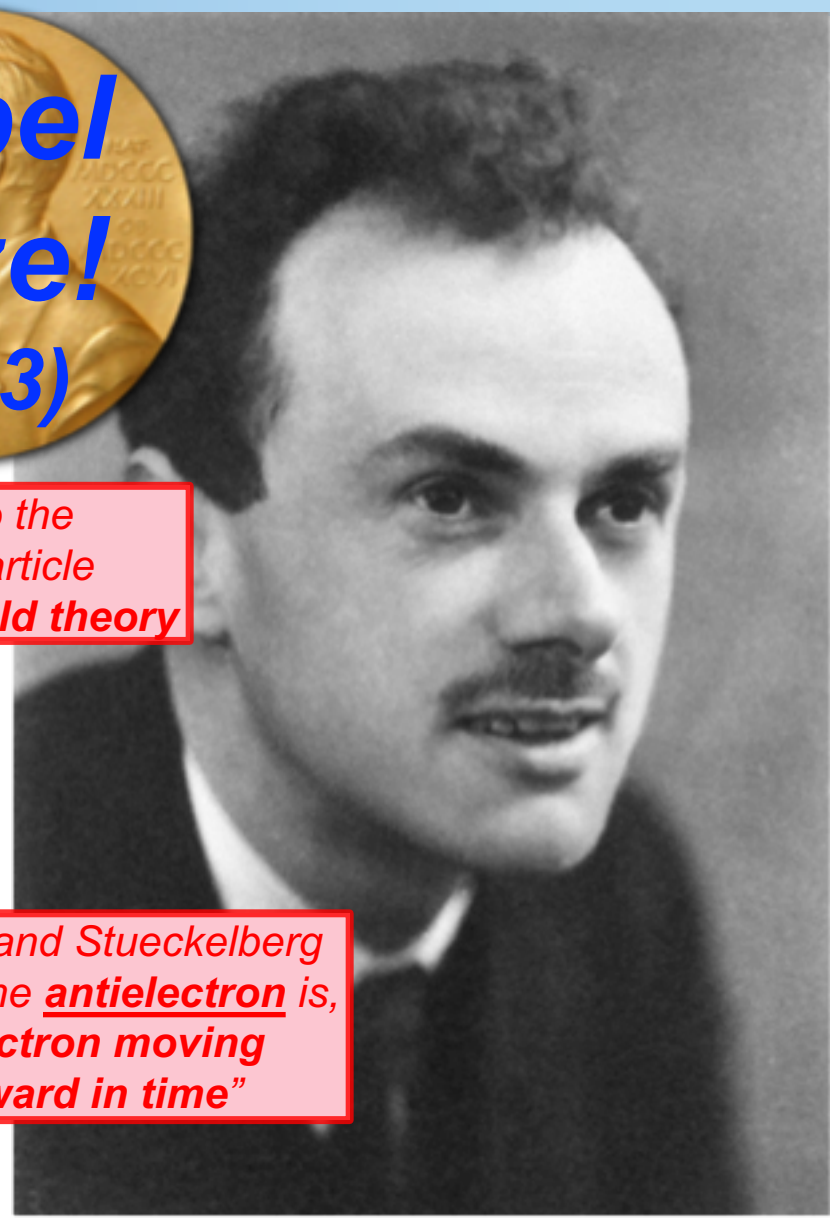
Demonstrated: Antimatter exists: Annihilation when electron and antielectron collide, producing two or more gamma ray photons

- Equation predicted the magnetic moment of electron spin, sometimes yielding **negative energy value**, concluding existence of **antielectron**
- Introduced the *(new)* concept of **electron spin**

Not continuous: Antiparticles "mirror" matter particles *(mutual annihilation when electron and antielectron collide)*



Lead to the many-particle ***quantum field theory***



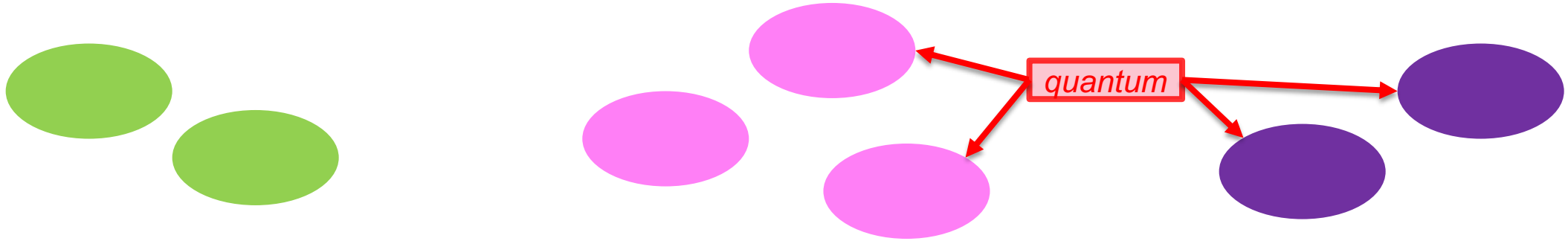
Feynman and Stueckelberg proposed the ***antielectron*** *is,* ***"an electron moving backward in time"***

Paul Adrien Maurice Dirac

08-Aug-1902 – 20-Oct-1984

Indivisible: Quantum

You have a photon, or you do *not*



- *Discrete*: Individual quantum **cannot be subdivided**!
  - *"Half-a-photon"* ***does not exist***!

- *Energy of photon:* **Planck's constant * frequency**

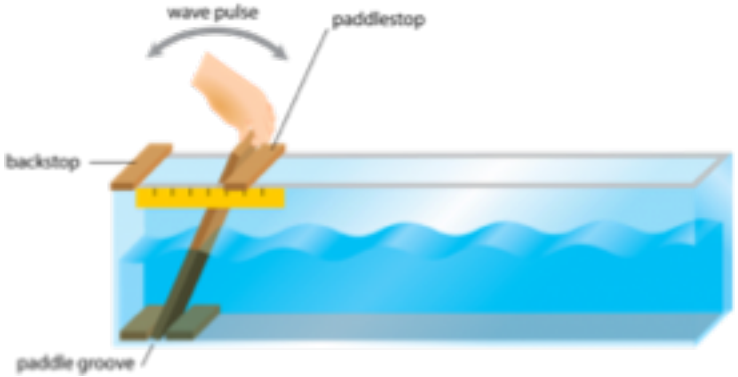
$E = hf$ — *photon*

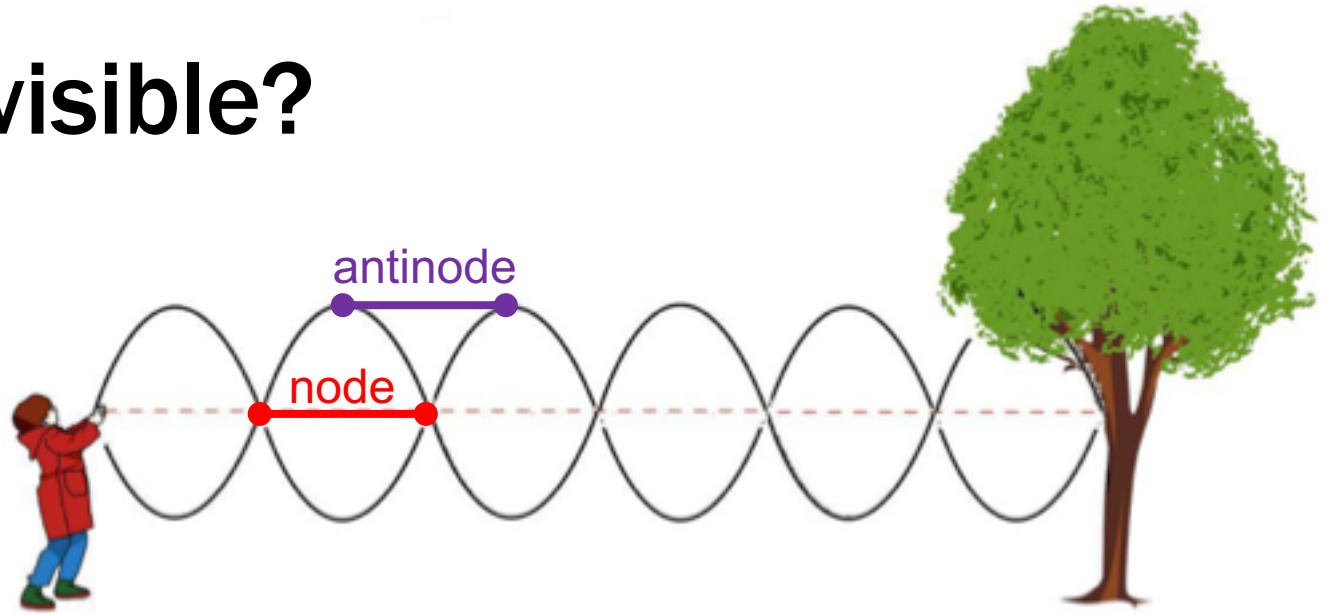
- **E** – energy
- **h** – Planck's constant
- **f** – frequency

*Is our "**epsilon**" (there is no fraction of this value)*

- **Energy** is **discrete** *(not continuous)*
- **Energy interactions** are **discrete** *(not continuous)*

Why Is Quantum Indivisible?





- All matter and energy in the universe **exists as a standing wave**
  - A "*stationary wave*": does not move spatially
  - *Amplitude may vary* with time
  - *Phase remains constant*
- **Different frequencies** for different standing waves
- **Quantum is the wavelength**
  - "half-a-wavelength" does not exist

A "solid" thing ***does not exist***

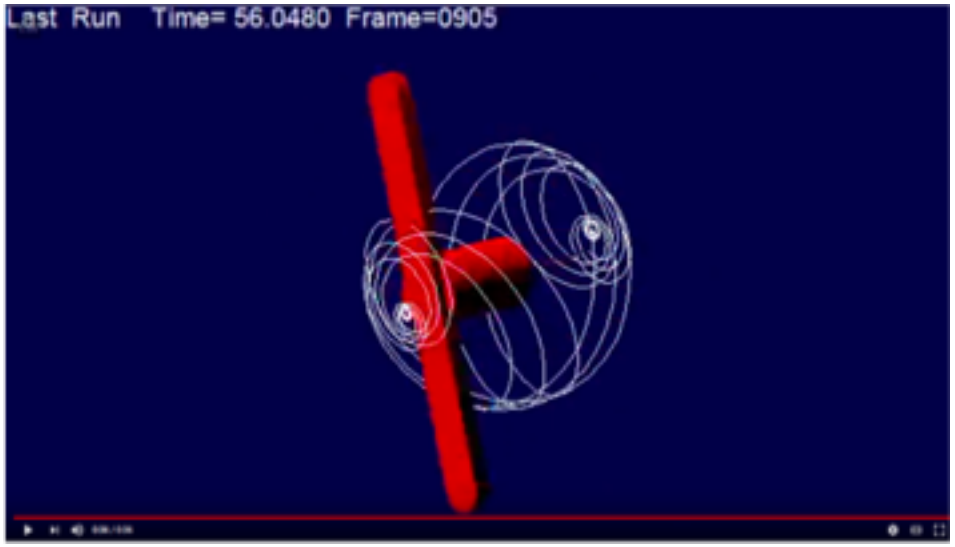
Standing waves:

- *Sound waves*
- *Visible light*
- *X-rays*
- *Mechanical waves*
- *Seismic waves*
- *Faraday waves*

Particle: A standing wave with itself

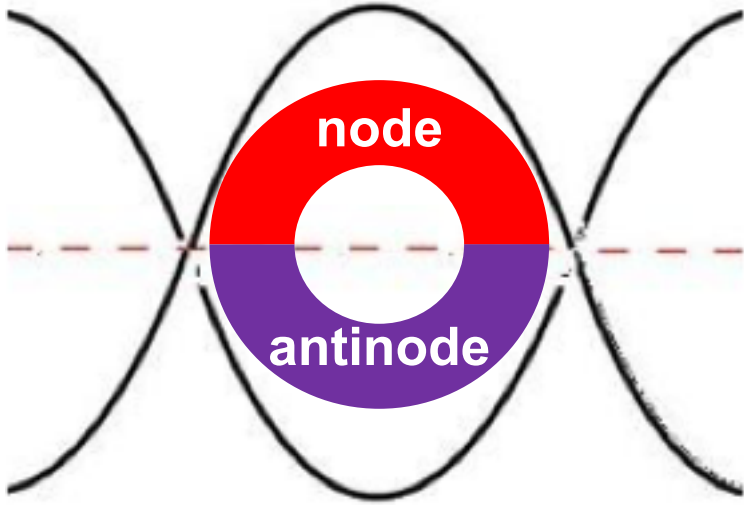
Entanglement: Multiple particles participating in the same standing wave

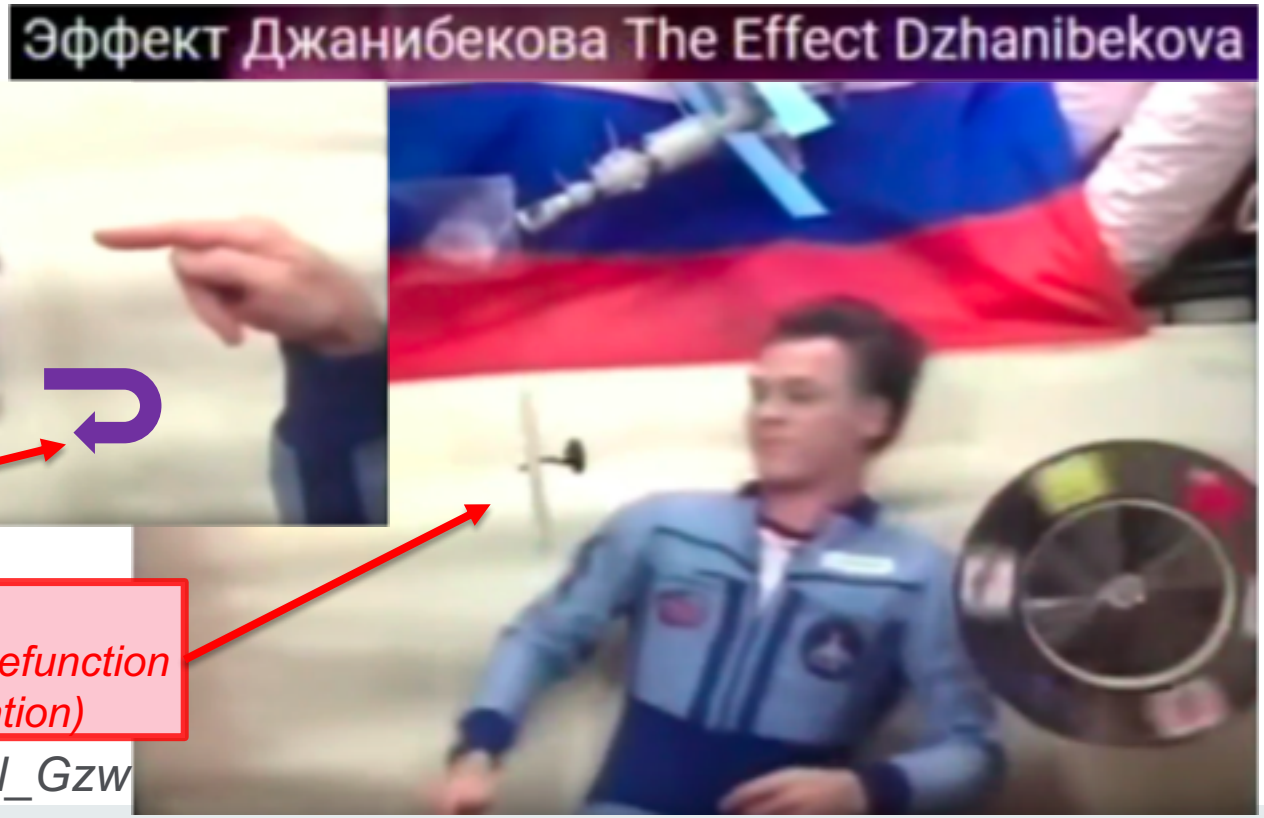
Demonstrated: Particle Standing Waveform



36 second video mathematically generating a particle standing wave in 3D space *(Russian):*
https://www.youtube.com/watch?v=LR5hkgfRPno

The "**Intermediate Axis Theorem**"



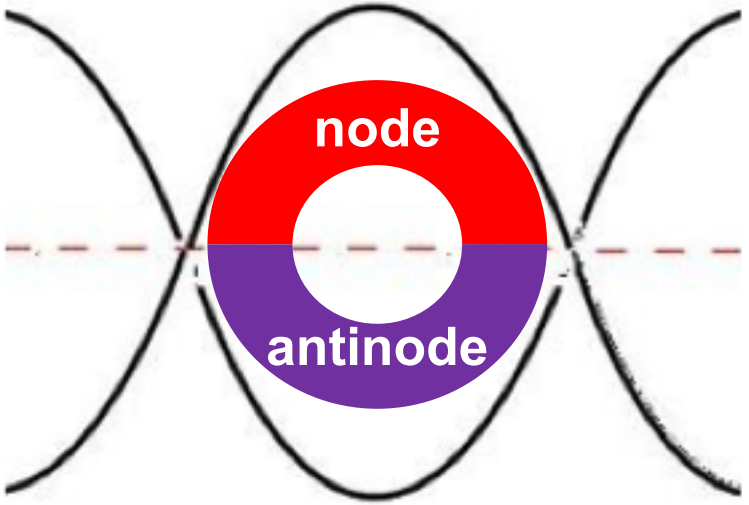


Flips left-and-right!
(Just like mathematical model)

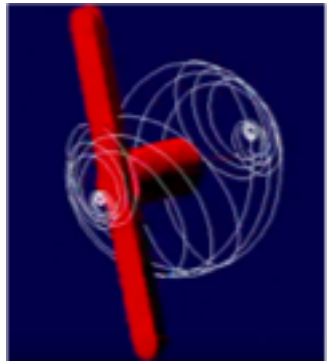
Built and demonstrated!
Spinning metal T-bar demonstrates particle wavefunction
(Russian physics experiments on space station)

https://www.youtube.com/watch?v=L2o9eBl_Gzw

What Is Quantized?



- **Everything In The Natural World** Is Quantized
  1. **Energy**: A photon is a discrete quantized value
  2. **Mass**: A sub-atomic particle is a discrete quantized value
  3. **All Other Units** *(example):*
     - ***Angular Momentum***: Spin of sub-atomic particle *(is a discrete quantized value)*



- **Computers** physically exist *(and thus are based on Natural Phenomenon)*
  - Rely upon discrete quantized values:
    - All mechanisms for **internal state**, and **state-transitions**
  - Discrete quantized values are the basis for *(all):*
    - **Hardware** specifications, **Wire** specifications, **Software** protocols

*An **Illusion Of Continuous** can only be maintained with a sufficiently low-resolution:*
- *Clock speed*
- *Data (bit-)resolution*

Not continuous:
A "Continuous" measure is *illogical*, and **has never been demonstrated** to exist

The "Natural" World: High Resolution

The **"Natural" World exists at high resolution** (*appears* continuous)

The Natural World is "High Resolution Digital": ***Discrete Quantized Values***

- Legacy Analog technologies supported higher resolution than legacy Digital technologies, because:
  - **Analog relied upon natural phenomenon** *(that we often did not understand)* which more closely matched the phenomenon-under-observation
  - **Digital was fundamentally limited** *(clock rate, bit-resolution)*
- Our ability to "describe" *("measure")* has improved
  - **Digital technologies are improved** *(clock-rate, bit-resolution)*
  - New technologies increasingly **rely upon (quantum) natural phenomenon**

At today's manufacturing scale for memory and electronics *(single atoms, or smaller):* ***Classical Physics*** **does not work**

Software state space **must model** *(new)* hardware increasingly resolving **quantum effects**

*Recall: "**Ultraviolet Catastrophy**" Where Classical Physics "broke down" (could **not** model, describe, nor predict observed phenomenon)*

Reality is “pixelated”

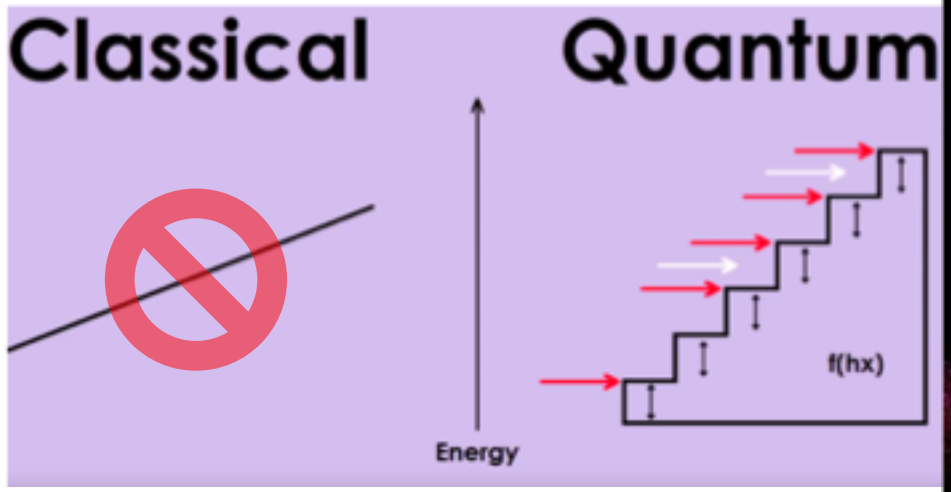
through ~~energy~~ interactions

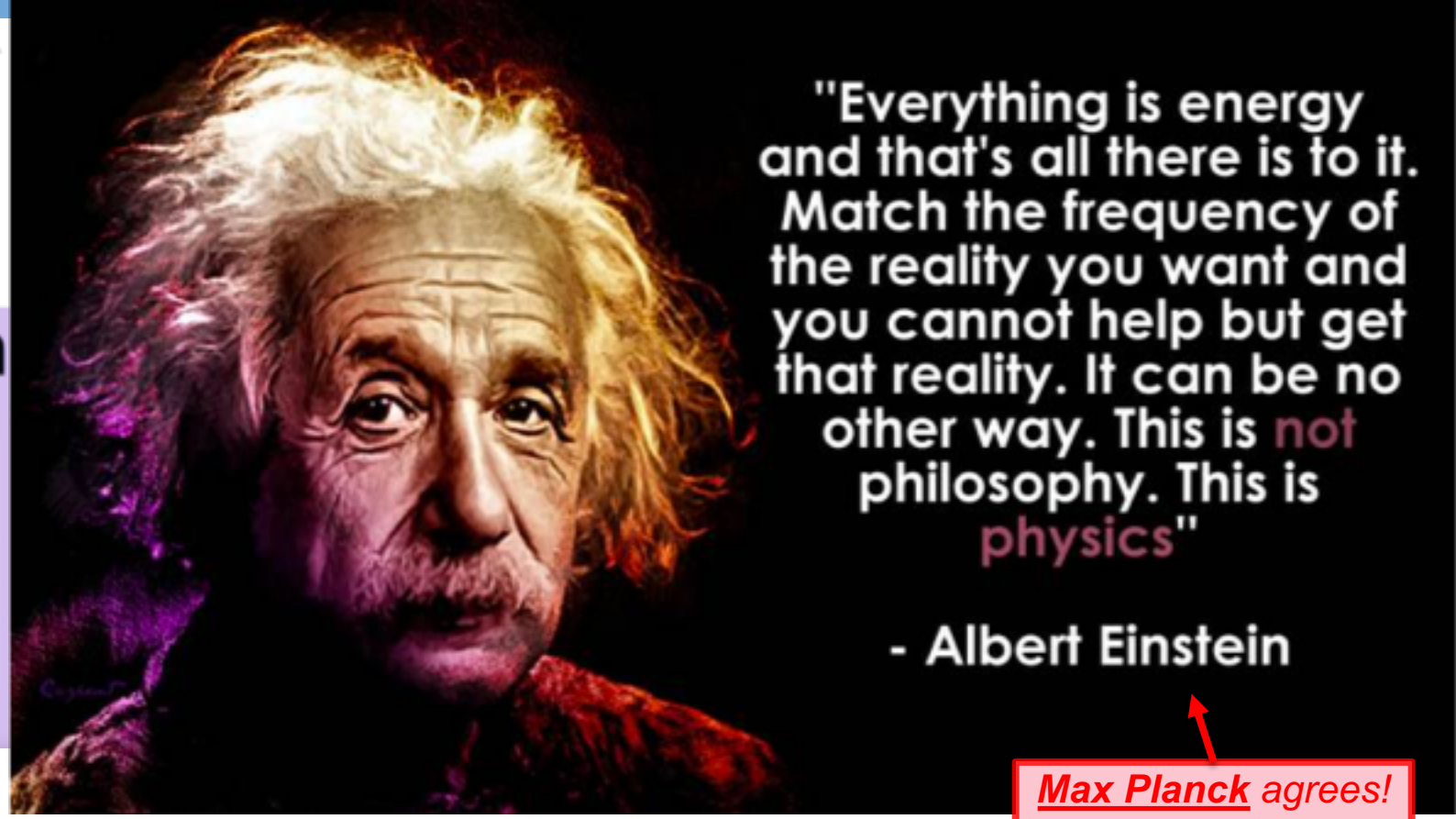
wave-form

with ~~the atomic structure~~

wave-forms

All Is ~~Energy~~ wave-forms





Max Planck *agrees!*
(…and goes further…)

- All is quantized *(**not** ranges of continuous values):*
  - Cold ⟷ Hot *(**energy**)*
  - Dim ⟷ Bright *(**energy**)*
  - Light ⟷ Heavy *(**mass**)*

Recall:
- Perfect convertibility between ***mass*** and ***energy*** *(and the reverse)*
- Mass is ***not*** constant

Applications of Quantum Mechanics

- **Microscopic** *(atomic scale):*
  - Electric current
  - Transistor, Diode
  - Flash memory *(uses quantum tunneling to erase)*
  - Laser, Maser, Phaser
  - Electron microscope
  - Night Vision Devices
  - Medical X-rays
  - Cell phones
  - Magnetic Resonance Imaging (MRI)
  - Materials Science
  - Ultrasound
  - Radiation therapies
  - Spectroscopy
  - Quantum computer

Quantum Mechanics is ***How Stuff Works***

*"The **Laws Of Nature**"*

- **Macroscopic** *(practical and visible without magnifying devices):*
  - Superfluidity – *fluid with zero viscosity*
  - Superconductivity – *exactly zero electrical resistance*
  - Quantum Hall effect – *precise measurement (i.e., "exact quantization")*
  - Proton tunneling – *quantum tunneling of proton across a barrier*

Valid States

...In Quantum vs. Classical Systems

What's the Radix?

C++ now

Radix: The Number Of Unique States

Name	Radix	Example
Binary	Base 2	[0,1] *(or similar)*
Ternary	Base 3	[0,1,2] or [-1,0,+1] *(or similar)*
…	…	…
Decimal	Base 10	[0..9] *(or similar)*
…	…	…
Sexagesimal	Base 60	[0..59] *(or similar)*
…	…	…

Multi-Valued Logic (**MVL**)

*No technically essential reason must be **Radix 2***

Radix 2 *(Binary, Bivalent logic):*

- Most ***Classical*** Systems
- Most ***Quantum*** Systems

Assertion for why **MVL is minimally required** *(for all software):*
The smallest "logical" thing that exists is a **Boolean** value, tagged with the possible attribute of "**`unknown`**"
(so 3+ states are minimally required to properly model all logic)

"Perhaps the prettiest number system of all is the *balanced ternary notation*…"

\- Donald Knuth

Art of Computer Programming, Vol 2

The Qubit

	Classical: "Bit"	Quantum: "Qubit"
Radix	2 *(i.e., "Binary" with Bivalent Logic)*	2 *(i.e., "Binary" with Bivalent Logic)*
Possible Values		
Value At Any Given Time		

The Qubit

	Classical: "Bit"	Quantum: "Qubit"
Radix	2 *(i.e., "Binary" with Bivalent Logic)*	2 *(i.e., "Binary" with Bivalent Logic)*
Possible Values	[0,1]	[0,1]
Value At Any Given Time		

The Qubit

Superposition:
Simultaneously holding all valid states

	Classical: "Bit"	Quantum: "Qubit"
Radix	2 *(i.e., "Binary" with Bivalent Logic)*	2 *(i.e., "Binary" with Bivalent Logic)*
Possible Values	[0,1]	[0,1]
Value At Any Given Time	`0 || 1`	`0 && 1`

Discrete!
(i.e., "Classical")

Superpositioned!
(i.e., "Quantum")

*A complex linear combination of a **0 state** and a **1 state***

Discrete* vs. *Superpositioned
is the defining characteristic between ***Classical*** and ***Quantum*** Systems

Disclaimer: Qubit

Disclaimer: Qubit does not *(really)* mean:
`0 && 1`

Qubit:
1. A ***complex*** linear combination of a `0` state and a `1` state
2. An **ontological category** *(a way of combining things)* that ***does not*** map onto any classical concept *(of "definite" or "discrete" state)*

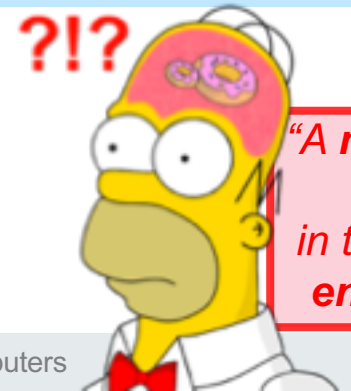
*Qubit has **no corollary** to classical concepts!*

```
//...is kind of like:
bool b;
std::complex<float> cn;

using Qubit = decltype( b * cn );
```

Complex number

*"A **qubit** is just a **unit vector** in two-dimensional **Hilbert Space**"*



*"A **monad** is just a **monoid** in the category of **endofunctors**."*

Quantum Mechanics
is *(merely)* a
generalization of probability

Quantum Mechanics supports:
- *Probability **Amplitudes***
- *Can be **positive** or **negative***
- *Can be **complex***

Qubit Is A Complex Number (based on i)

Real Number

i

Complex Number: $a + b * \sqrt{-1}$

- *"**Imaginary**" number, because no "real" number satisfies equation*

```
using ComplexNum = std::complex<float>;
class Qubit {
  bool          b_;
  ComplexNum* cn_;//entangled with storage elsewhere
  public:
    //'value()' mutates 'b_' (for waveform collapse)
    auto value() {
      b_ = static_cast<bool>(b_ * (*cn_));
      cn_ = nullptr; //decohere – drop entanglement
      return b_;
    }
};
```

"Read" is a **mutating** *(destructive)* **lazy-compute**

***Example is Incomplete**, should also:*

1. Mutate '`b_`' state in other entangled `Qubit` instances
2. Mutate other "entangled" members *(example: reading `position` or `momentum` should mutate both values)*
3. Sever entanglement among other `Qubit` instances

Eigenstate vs. Eigenvalue

"**eigen**" *(Dutch):*
"inherent" or "characteristic"

- For a quantum data object, **terminology becomes important** to describe the "status" of the "object-state"
- State of a quantum object:
  1. **May exist in "discrete" state** *(i.e., "pinned" to certain value as a result of measurement)*
  2. **May exist in "superpositioned" state** *(i.e., as a value-with-probability)*
- *Example: an electron has a "spin" – perhaps "up", and perhaps "down"*

Eigenstate vs. Eigenvalue

"**eigen**" *(Dutch):*
"inherent" or "characteristic"

- For a quantum data object, **terminology becomes important** to describe the "status" of the "object-state"
- State of a quantum object:
  1. **May exist in "discrete" state** *(i.e., "pinned" to certain value as a result of measurement)*
  2. **May exist in "superpositioned" state** *(i.e., as a value-with-probability)*
- *Example: an electron has a "spin" – perhaps "up", and perhaps "down"*

	Description	Example
Eigenstate	The **"pinned" value**	*the spin is "up"*
Eigenvalue	A **definite value** that may be observed	*one of "up" or "down"*

*Object State that is "**certain**"*

*Also called the **quantum state** (is "**uncertain**")*

A Possible object state that may be observed (one of the superset of all possible eigenstates)

An object state is *(always)* one of:
1. **Eigenstate** *(i.e., "pinned")*
2. **Eigenvalue** *(i.e., "superpositioned")*

An **eigenvalue becomes an eigenstate** upon wavefunction collapse

Wavefunction Collapse

Wavefunction Collapse: Mutation to replace an uncertain value with a definite state

*Is the "**essence of measurement**" in a quantum system*

Wavefunction collapse:
- Is the **essence of measurement** in a quantum system
- Is a **Black Box for irreversible interaction** within a Classical environment

- **Wavefunction** (ψ)– a *(mathematical)* description of the **superposition of several eigenstates**
  - Describes probabilities for the possible results of measurements
  - Probability: Is a complex-value probability amplitude
- **Collapse**: Reducing to a single eigenstate *(from an eigenvalue)*
  - Occurs: "By observation" *(By "measurement")*
  - Is by-product of matrix-math evaluation *(matrix coefficients are discarded in computing the discrete result value)*

Wavefunction Collapse

```
using ComplexNum = std::complex<float>;
class Qubit {
  bool          b_;
  ComplexNum* cn_;//entangled with storage elsewhere
  public:
    //'value()' mutates 'b_' (for waveform collapse)
    auto value() {
      b_ = static_cast<bool>(b_ * (*cn_));
      cn_ = nullptr; //decohere – drop entanglement
      return b_;
    }
};
```

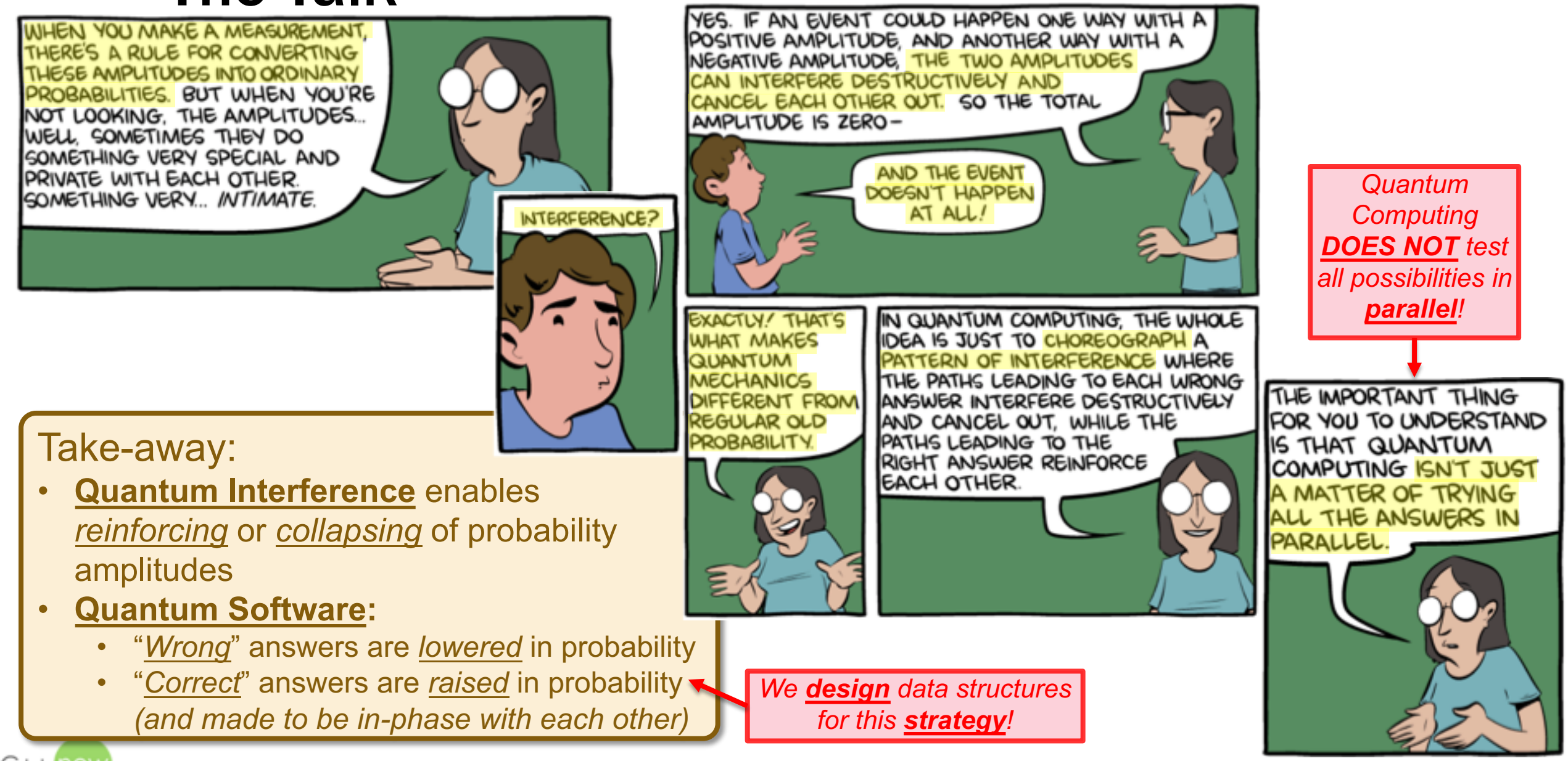
Mutation
(assign definite value from uncertain value)

Wavefunction Collapse: Mutation to replace an uncertain value with a definite state

“The Talk”

https://www.smbc-comics.com/comic/the-talk-3

Scott Aaronson & Zach Weinersmith



*Quantum Computing **DOES NOT** test all possibilities in **parallel**!*

Take-away:

- **Quantum Interference** enables *reinforcing* or *collapsing* of probability amplitudes
- **Quantum Software**:
  - “*Wrong*” answers are *lowered* in probability
  - “*Correct*” answers are *raised* in probability *(and made to be in-phase with each other)*

*We **design** data structures for this **strategy**!*

Reading A Value

```
uint32_t value = ...;
// How many states in 'value'?
```

- Classical: **Read** number of states:

- Quantum: **Read** number of states:

Reading A Value

```
uint32_t value = ...;
// How many states in 'value'?
```

- Classical: **Read** number of states:

(2^32)

0 through 4,294,967,295

- Quantum: **Read** number of states:

Reading A Value

```
uint32_t value = ...;
// How many states in 'value'?
```

- Classical: **Read** number of states:

(2^32)

0 through 4,294,967,295

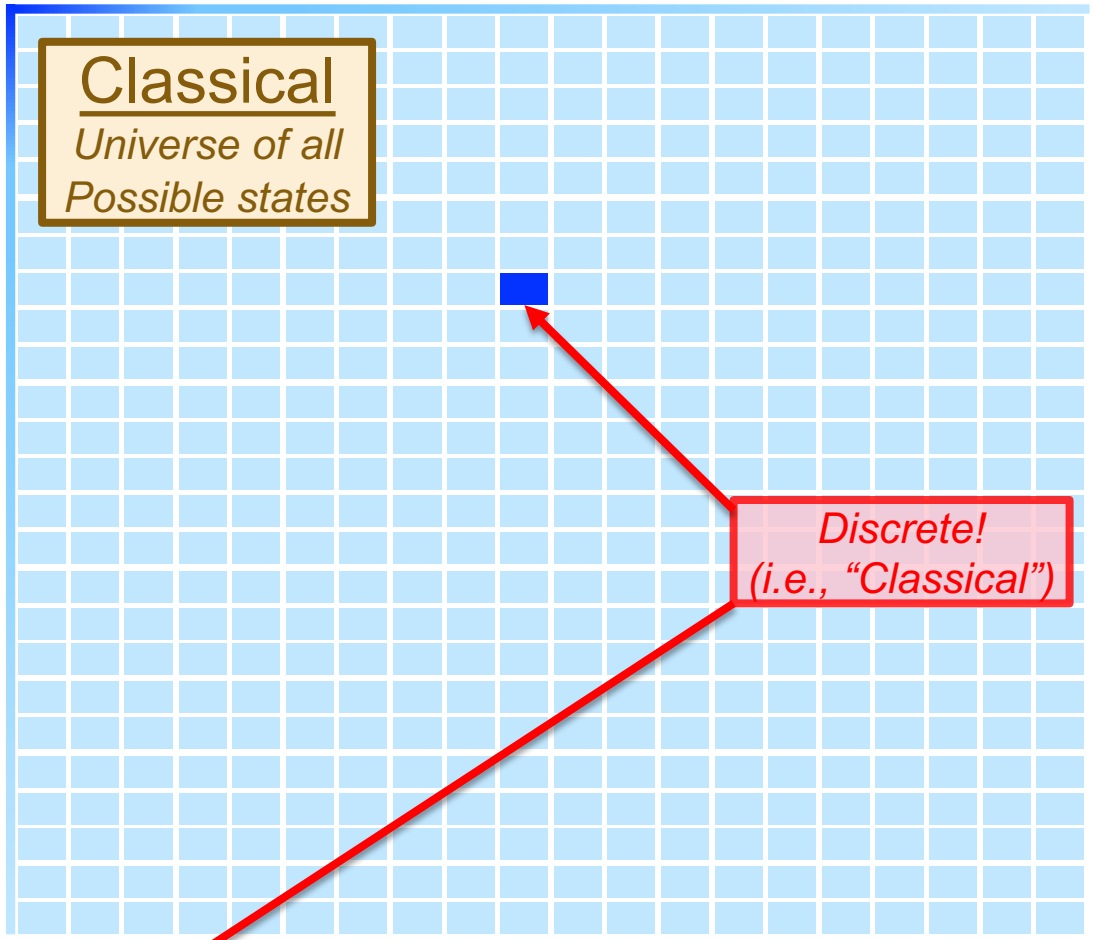
- Quantum: **Read** number of states:

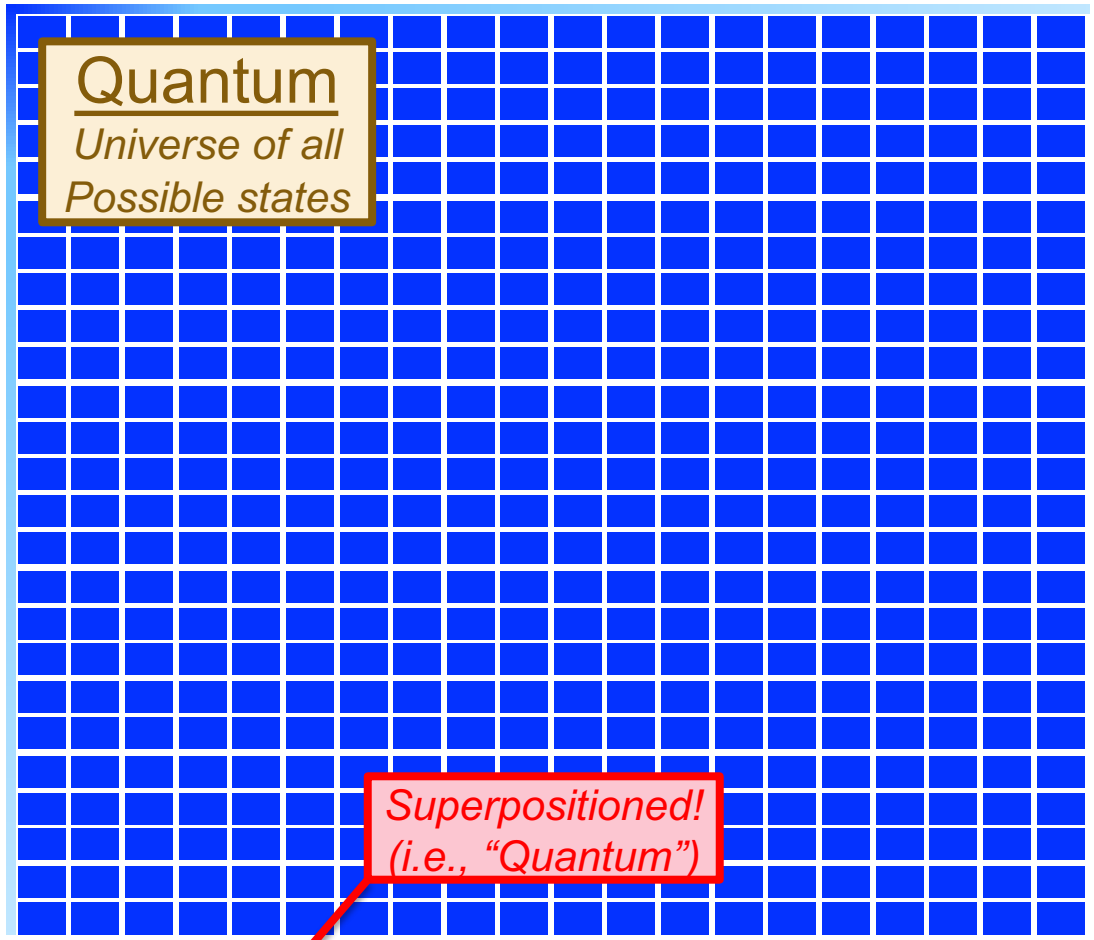
(2^32)

0 through 4,294,967,295

Both are binary!
(Radix == 2)

Number Of States At One Time






```
uint32_t value = ...;
// How many states at one time?
```

```
q_uint32_t value = ...;
// How many states at one time?
```

Memory Requirements

Memory requirements are ***fundamentally different*** between Classical and Quantum Systems

- Number of data objects to represent all possible 32-bit binary integers:

Classical:

- `(2^32)` data objects *(32-bit)*
- Memory required:
  - `(2^32) * sizeof(uint32_t)`
  - `~= 43.36` Gigabyte

Quantum:

- `(1)` data object *(32-qubit)*
- Memory required:
  - `1 * sizeof(q_uint32_t)`
  - `1 q_uint32_t`

Quantum Supremacy

Quantum Supremacy: That point at which a Quantum computer outperforms a Classical computer

- *Example:*
  - A single **50-qubit number** equates to ~= *9 Petabyte* memory on Classical computer

Classical:

- `(2^50)` data objects (50-bit)
- Memory required:
  - `(2^50) * sizeof(uint50_t)`
  - `~= 9` Petabyte

Quantum:

- `1` data object (50-qubit)
- Memory required:
  - `1 * sizeof(q_uint50_t)`
  - `1 q_uint50_t`

"50-qubit" Quantum Supremacy assumes a very narrow class of problems *(that can be solved within the 50-bit value space)*

A More Practical Quantum Supremacy

- Quantum offers *(fundamental)* benefits:

1. **Scale**
   (profoundly exponential!) increase in scale for **state** that can be "feasibly" **considered and manipulated**

   Mechanism: Quantum ***interference***

2. **Algorithmic Approach**
   Quantum algorithms offer **different advantages** over Classical algorithms, **for *some* classes of problems**

   Must leverage quantum interference by:
   1. ***discover algorithm***
   2. ***define data structure***

Some problems are inherently **inefficient in Classical**, but **efficient in Quantum**; and ***vice-versa***!

How Big Can Your Classical Program Get?

- *Serious question:* **How Big** Can Your *Executable* Get?
  - 100MB? 1GB? 1TB? 1PB? *(…Exabyte, Zettabyte, Yottabyte…?)*
  - How long would it take to load?
  - Paging helps, but will the cache always be cold?
- *Serious question:* **How Big** Can Your *Data Set* Get?
  - Due to size, must it be physically partitioned / distributed?
  - How to synchronize / coordinate transactional updates of large datasets?

*Quantum computers can by-pass **von Neumann architecture** limitations*

Observation

- **Data Centers** are **specified by limiting factor**: [Watts/energy] *(limited by cooling);* and ***not*** by artifact *("storage capacity")*
- **Computers** could be **specified in terms of limiting factor**: [computed results] *(limited by throughput);* and not by artifact *("retired instruction count")*

*"Retired Instruction Count" is (largely) **irrelevant** on a quantum computer*

Quantum Computing Issues

*Is **PRO**!*
*Is **CON**!*

1

(Tremendously) Higher Complexity

Hardware:

- Hard to build, tricky tolerances
- Exhibit phenomenon we do not understand
- Primitive ISAs, APIs

Software:

- Data structures, algorithms *(exponentially complex over Classical)*
- High state coupling *(is the "opposite" of Functional Programming)*
- Need research into algorithms, idioms, APIs, confidence intervals

2

Non-Observable State

- **New techniques** are required: "*Use*" state without "*Inspecting*" state
- **Forever Unknowns**: Based on **Probabilities**, **non-quantified external influences**

Observation

Quantum data structures offer
higher computational density
(over classical and functional approaches),
at the **cost of significantly higher complexity**

Quantum: Is Multi-Dimensional

- Quantum isn't quantum: It's **Multi-Dimensional**
  - *"This surprising behavior is quantum !"* ← ***wrong!***
  - *"This surprising behavior is multi-dimensional !"* ← ***correct!***

Conceptually (and literally):

- Quantum computing **offloads processing to "other dimensions"** outside the bounds of "Time" in this dimension

Quantum is **NOT infinitely "fast"**:

1. **Must sequence** operations for **entanglement**, and **wavefunction collapse** *(similar to overhead from designing an inter-dimensional concurrency interface)*
2. Quantum **error correction** increases calculation times *(to combat decoherence / dephasing)*
3. Many quantum **algorithms must iterate "many times"** *(to increase probability of "correct")*
4. **Quantum data structures** in a Classical system **merely approximate "lazy compute"** *(time-shift computation latencies)*

Time

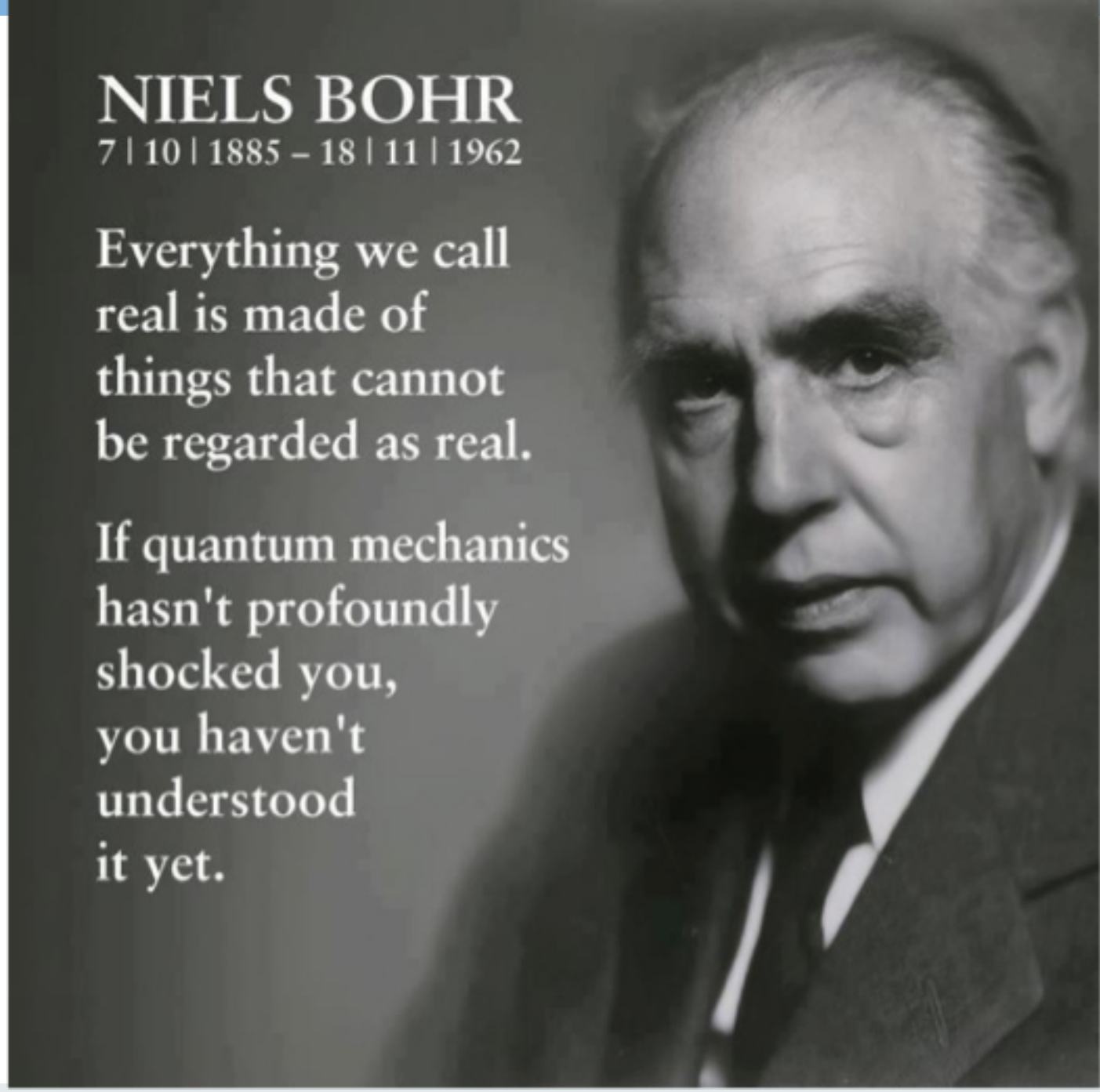
Time Is A Technology

C++ now

“If quantum mechanics hasn’t profoundly shocked you, you haven’t understood it yet.”

- Niels Bohr

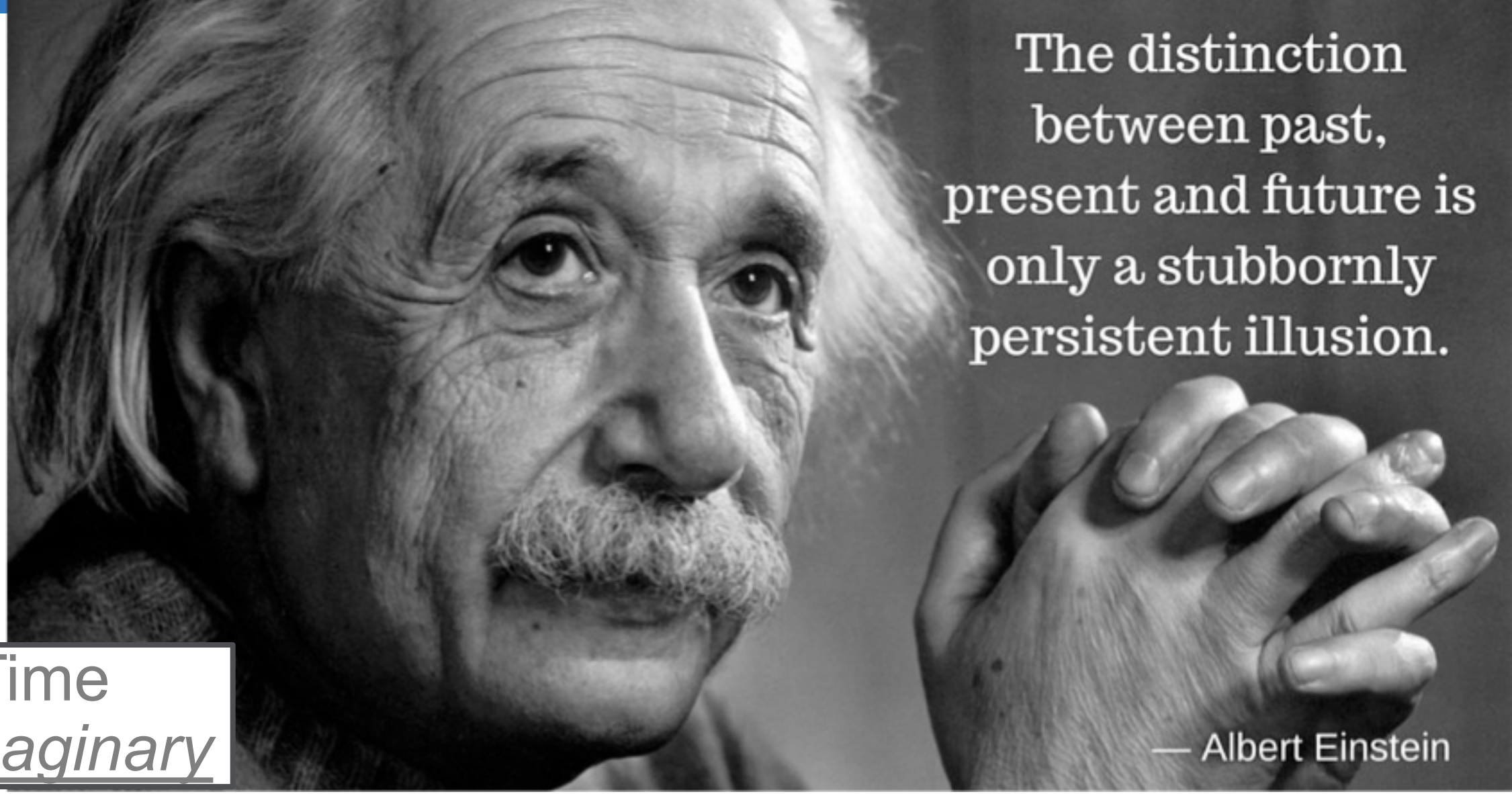
7-Oct-1885 – 18-Nov-1962



Time

- To intuitively work with Quantum Computing, must accept reality:

Time
does *not* exist



Time
is *imaginary*

Timeline discontinuities *can* and *do* occur

(witnesses commonly label them as “miracles” and “magic”)

Time Is A Technology

Time
is *imaginary*

- Time is a **technology** by which we **serialize correlation of effects**

Question: If Time is a Technology, what is *implied*?

- Like All Technologies:
  1. Time can be ***manipulated***
  2. Time can be ***by-passed***

Theory of Relativity!
*(We **know** Time is **not** absolute!)*

The Quantum Universe exists *outside* of time
- It does not use Time

- Humans *do* think in terms of time
  - Consciousness navigating Quantum Entanglement leads to a ***perception*** of time *(this is a separate talk)*

- Software Engineers are familiar with *"Time As A Technology"*

Using Time As A Technology

- All things have: **begin** **middle** **end**

Among all things:
Time is a **technology for sequencing**
based on [***begin***, ***middle***, ***end***]

Time is a **technology** to **enable synchronization**
- across *clocks*
- across *threads / processes*
- across object *lifecycles*
- across logical *sequences*

Example:
Critical section to control ***sequenced access*** *to (mutating) state*

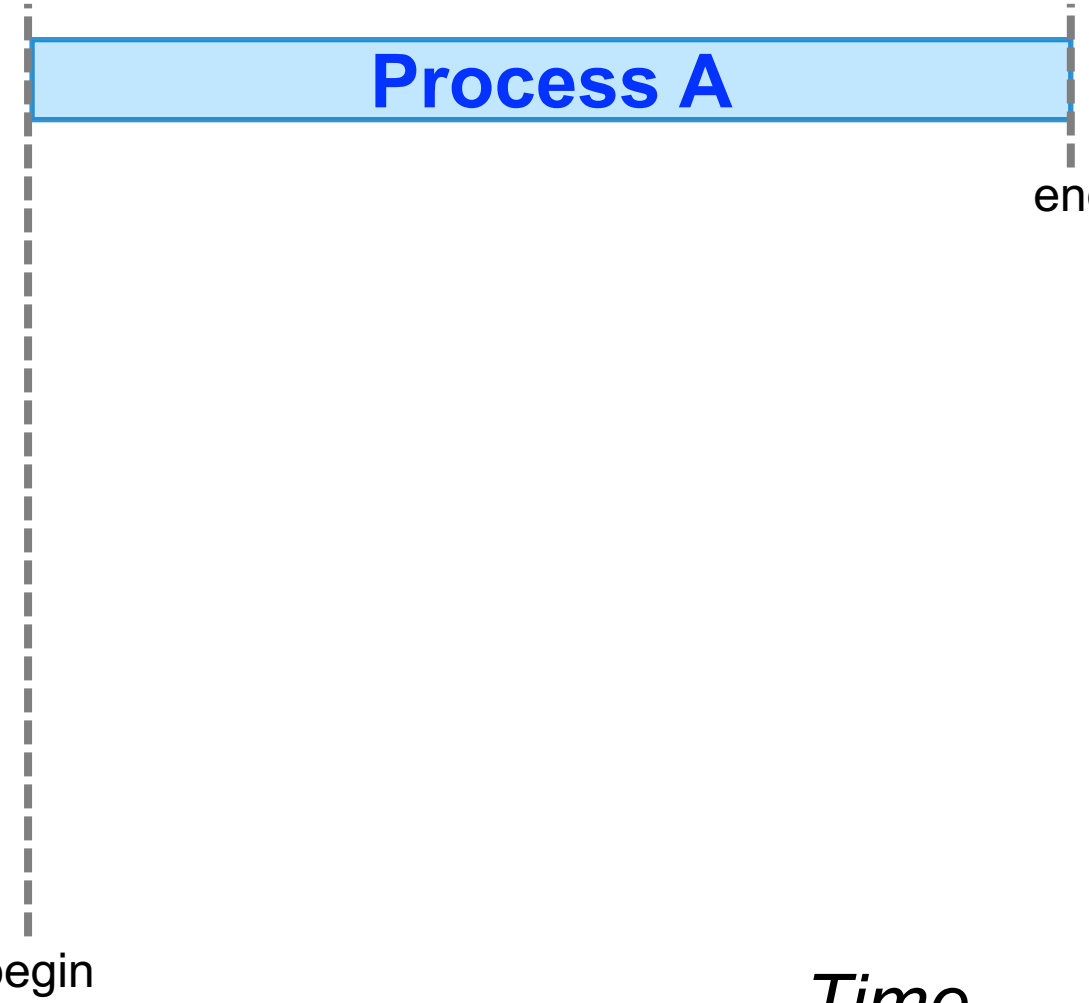
- We **model this in C++**:
  - The C++ Object Model: *[ctor…dtor]*
  - Is why the most powerful idiom in C++ is: ***RAII***
  - Many other languages do not have a *dtor*, so can be difficult to model Time properly

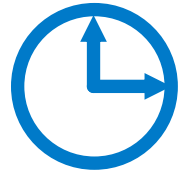
Is how quantum data structure ***works****!*

Modeling the **quantum lifecycle** is ***CRITICAL***:
1. **Setup** *(Entangle)*
2. **Teardown** *(Wavefunction-Collapse)*

Time As A Technology: Duration

- Software Engineers are familiar with *"Time As A Technology"*

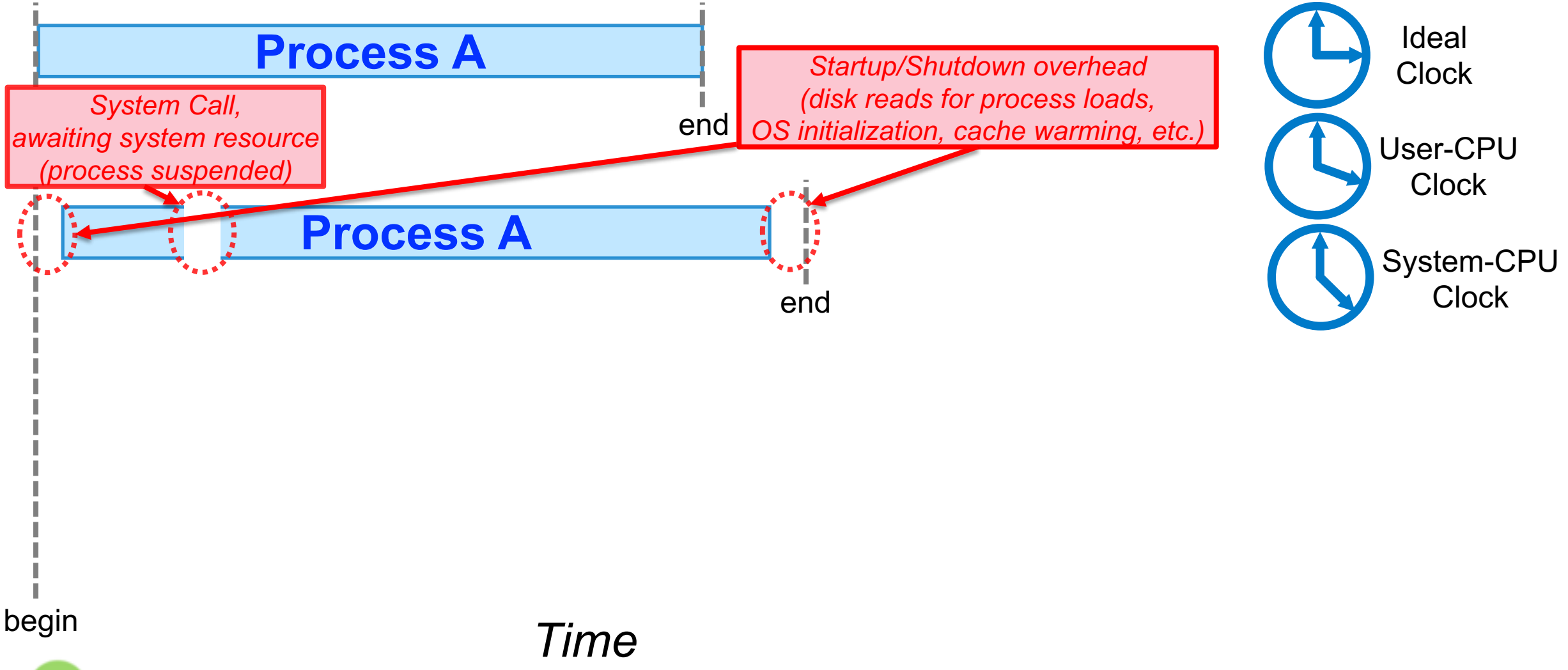




Ideal Clock

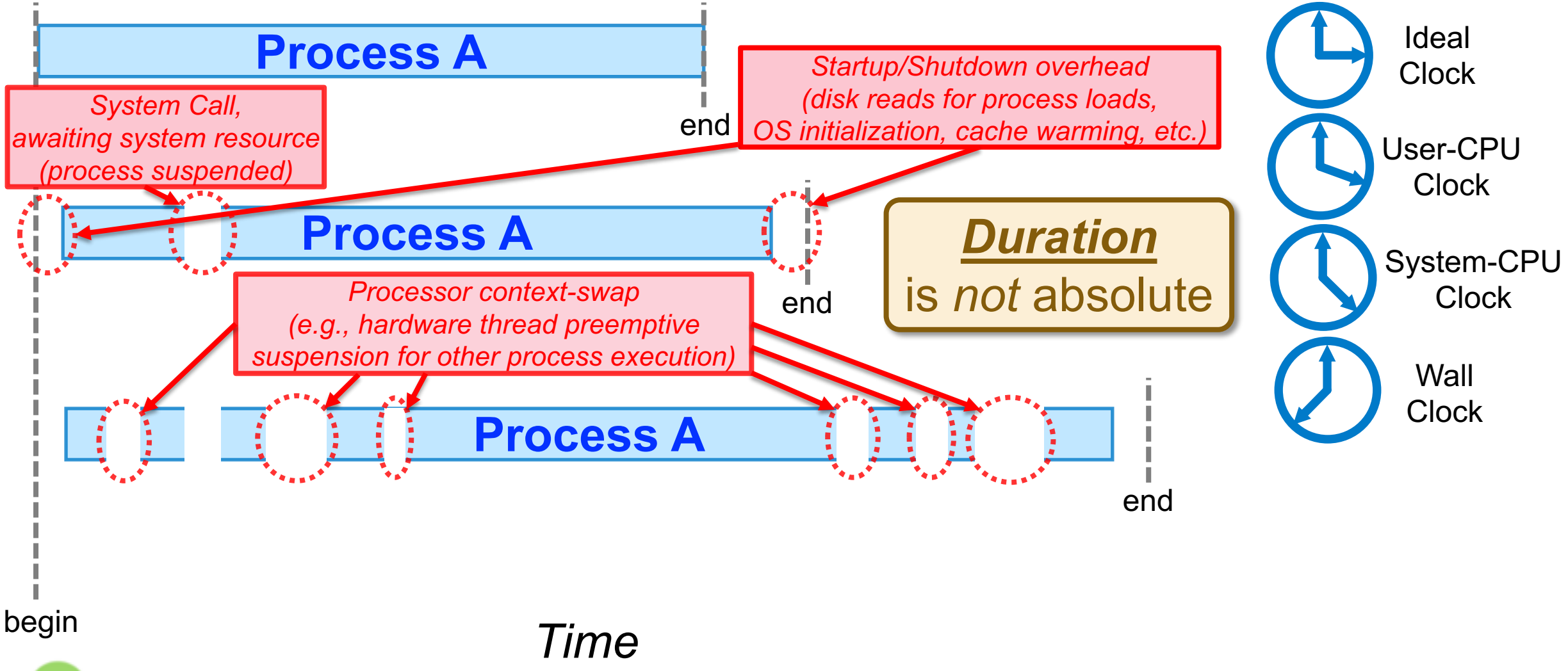
Time As A Technology: Duration

- Software Engineers are familiar with *"Time As A Technology"*



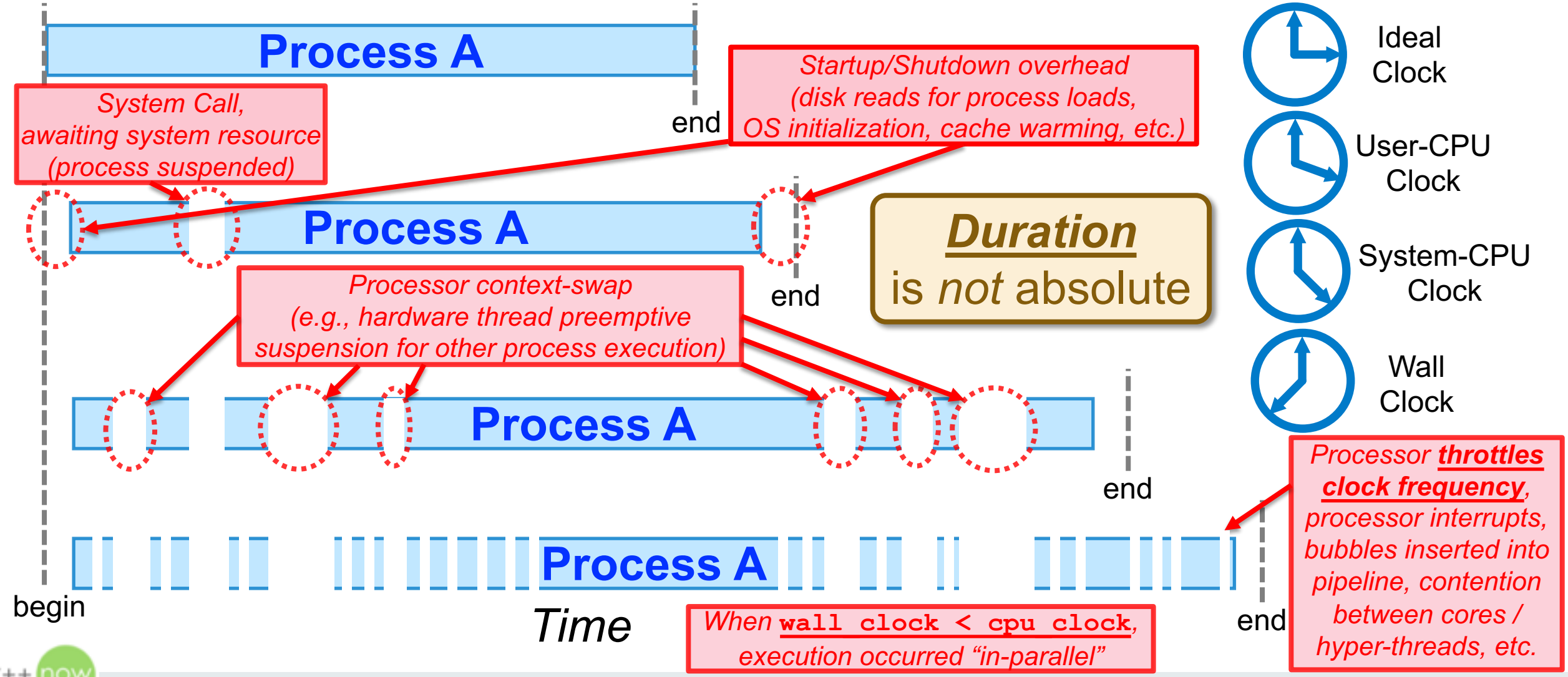
Time As A Technology: Duration

- Software Engineers are familiar with *"Time As A Technology"*



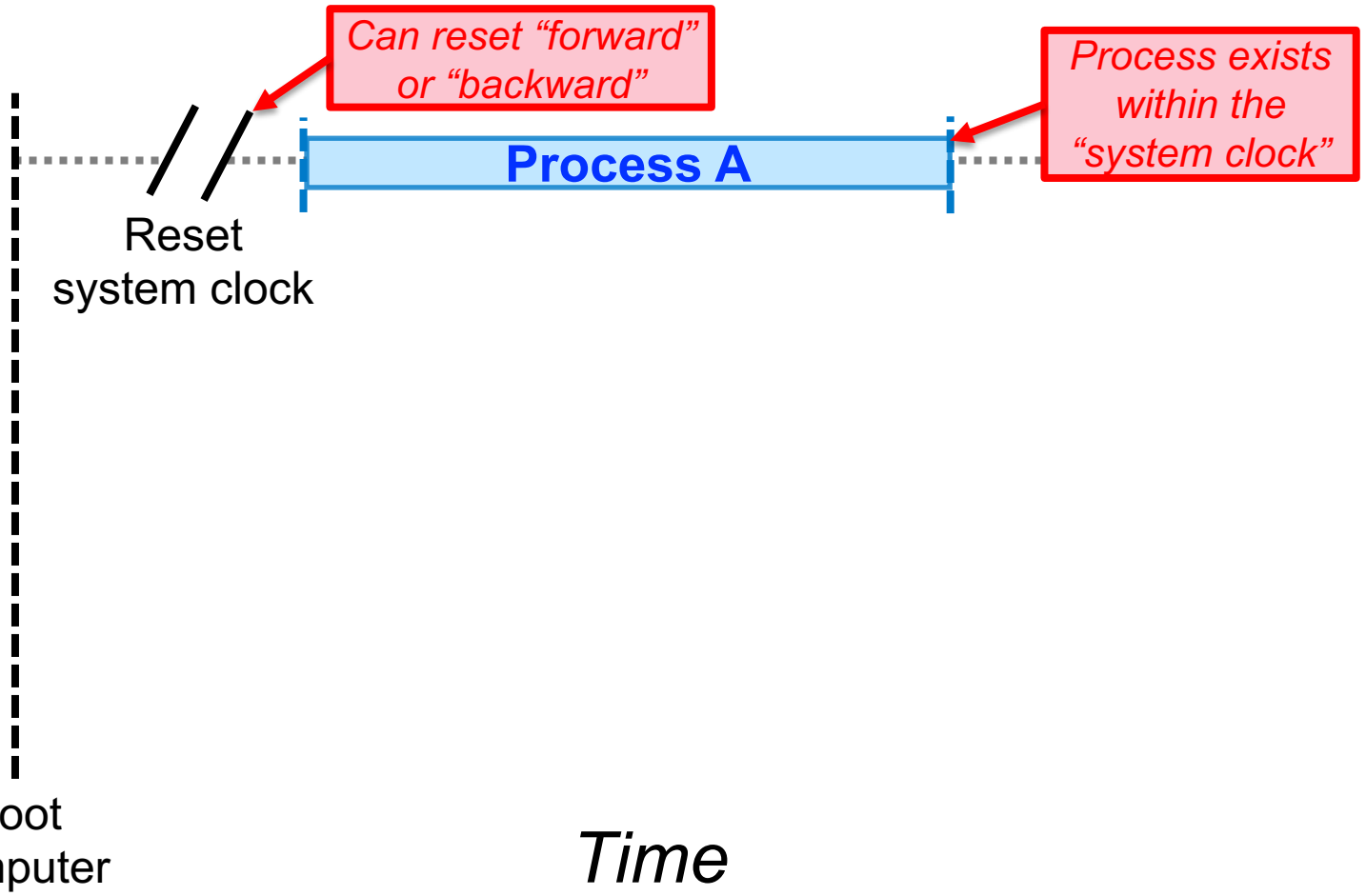
Time As A Technology: Duration

- Software Engineers are familiar with *"Time As A Technology"*



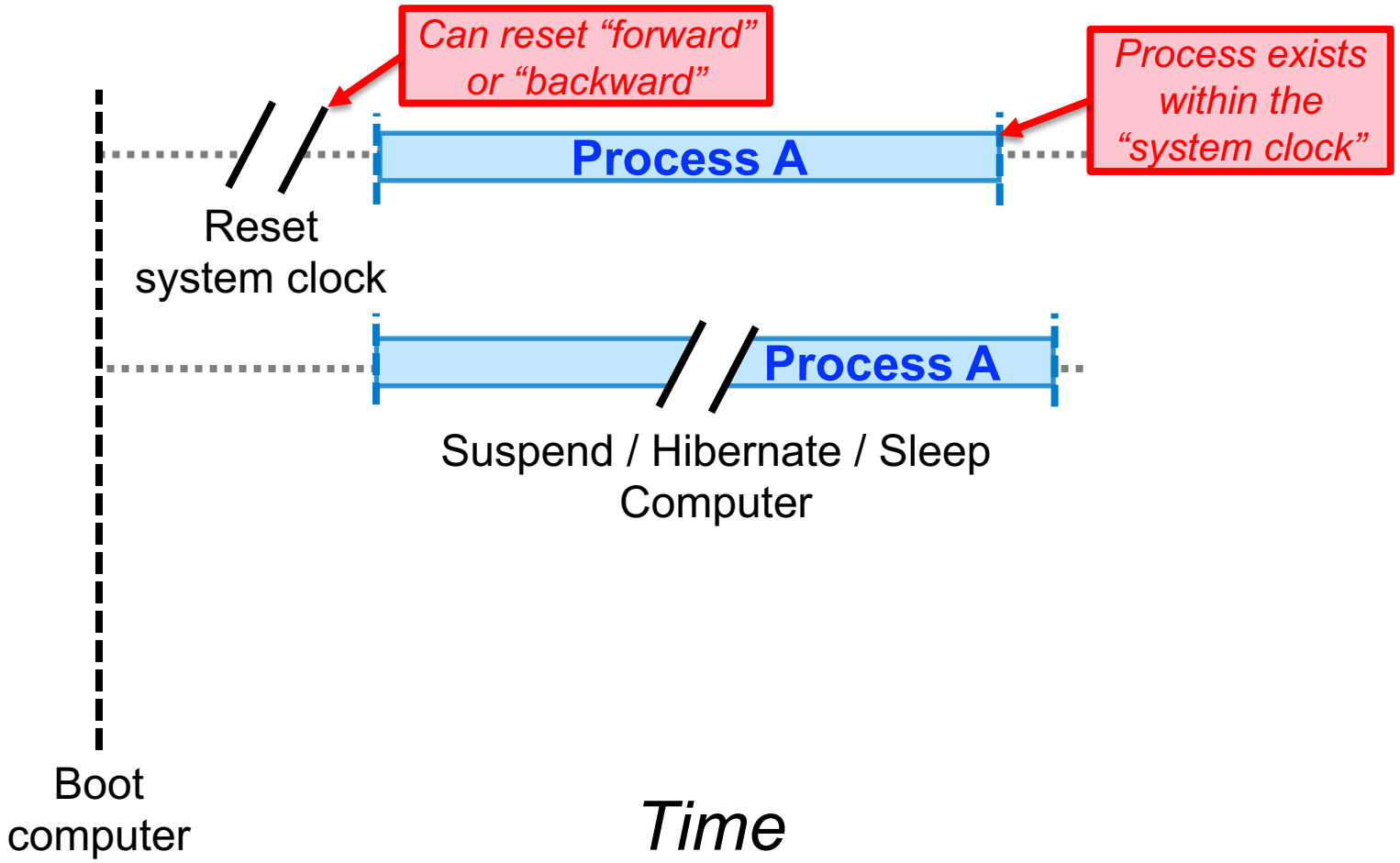
Time As A Technology: Absolute Measure

- Time has no "absolute measure"
  - Recall: Einstein's Theory of *Special Relativity*, *General Relativity*



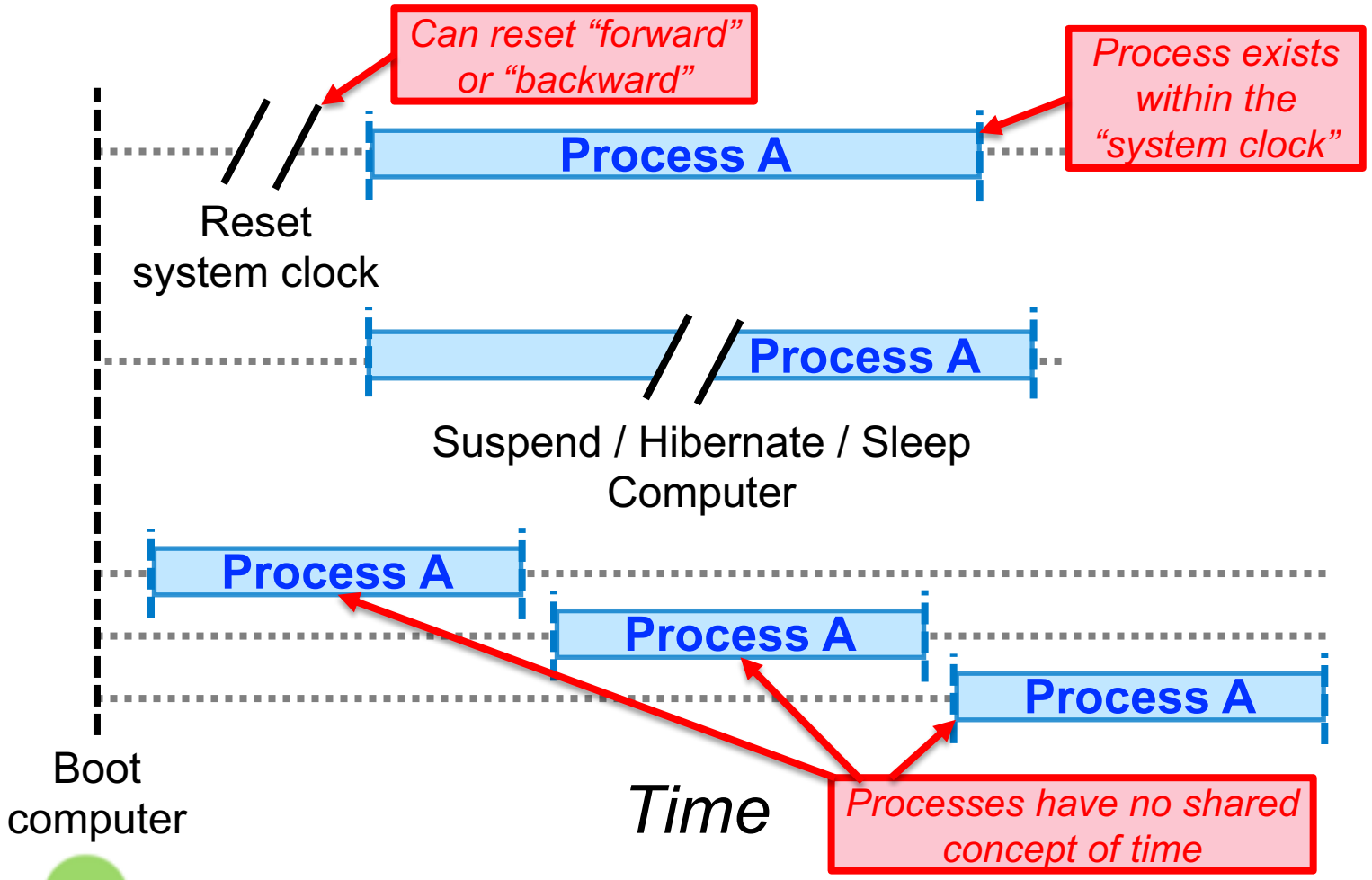
Time As A Technology: Absolute Measure

- Time has no "absolute measure"
  - Recall: Einstein's Theory of *Special Relativity*, *General Relativity*



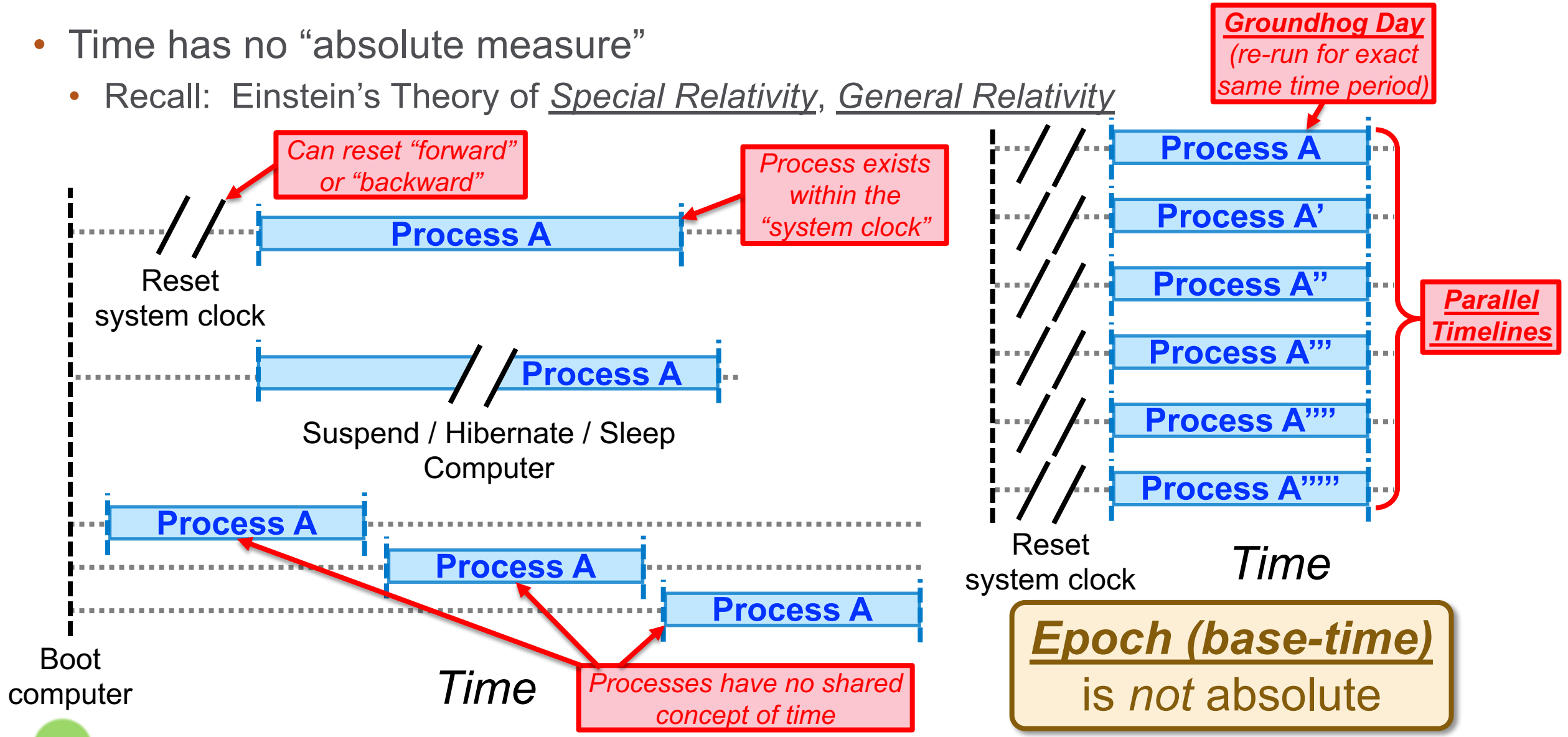
Time As A Technology: Absolute Measure

- Time has no "absolute measure"
  - Recall: Einstein's Theory of *Special Relativity*, *General Relativity*



Time As A Technology: Absolute Measure

- Time has no "absolute measure"
  - Recall: Einstein's Theory of *Special Relativity*, *General Relativity*



Time Metaphor: A Movie Reel

- Movie Reel: Composed of **ordered sequence** of image-frames
- Image-frames are ***static*** *(i.e., the whole reel exists)*

Time Sequence: The ***illusion*** resulting from sequenced inspection of individual image-frames

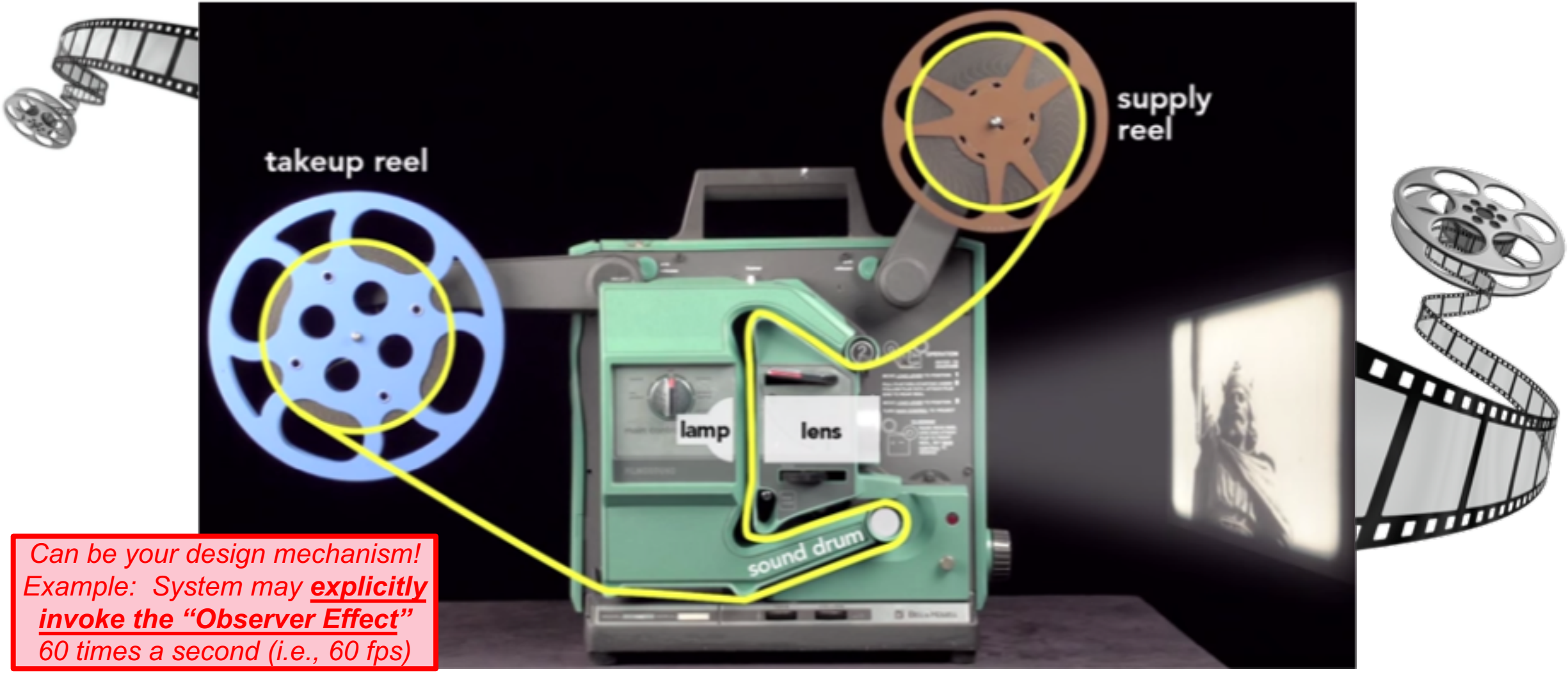
Also possible: ***Random-access*** *frame inspection!*

- Not Fatalistic:
  - Movie exists as a collection of **probabilities**
  - Plot / Story-Arc is established as your **consciousness** navigates probabilities and **collapses waveforms** based on biases / preferences (i.e., ***"The Observer Effect"***)

We can pretend "Time" *exists*, and is *static*; but it does *not*, and is *not*

Time is a ***Technology***

Simple Technology Manipulating Time



Can be your design mechanism!
*Example: System may **explicitly invoke the "Observer Effect"** 60 times a second (i.e., 60 fps)*

…Happens to look *a lot* like a ***Turing Machine***…

Time Technology: Stop Motion Carousel

- Is static *(3D printed)* Carousel
- When Rotated *(at **specific phase**):* Presents **illusion** of time-lapsed motion

4-Mation carousel 1: Jumping Frogs – a 3D Zeotrope

https://www.youtube.com/watch?v=d_-j9uuaDOQ

Time As A Technology

Time
is *imaginary*

A real **mathematical basis exists** *(supported by* ***physics observations****)*
to demonstrate **actions in the future can have an effect**
on the probability patterns that exist in the present

Effect Precedes Cause
(is the basis for
many movie plots
and Star Trek episodes)

- *Recall:* All Technology can be ***manipulated*** and ***bypassed***
  - **Mass** is an *effect*, not a cause *(is not constant)*
  - **Time** is an *effect*, not a cause *(is not constant)*

Fred Alan Wolf (Physicist): **Information from the possible futures** comes to the present in an "**offer wave**", and which future we navigate depends on which choices we make

The Observer Effect: Our ***intent influences probability amplitudes***
(is how we navigate offer waves)

The Quantum Data Structure

Based On Interference (Entanglement)

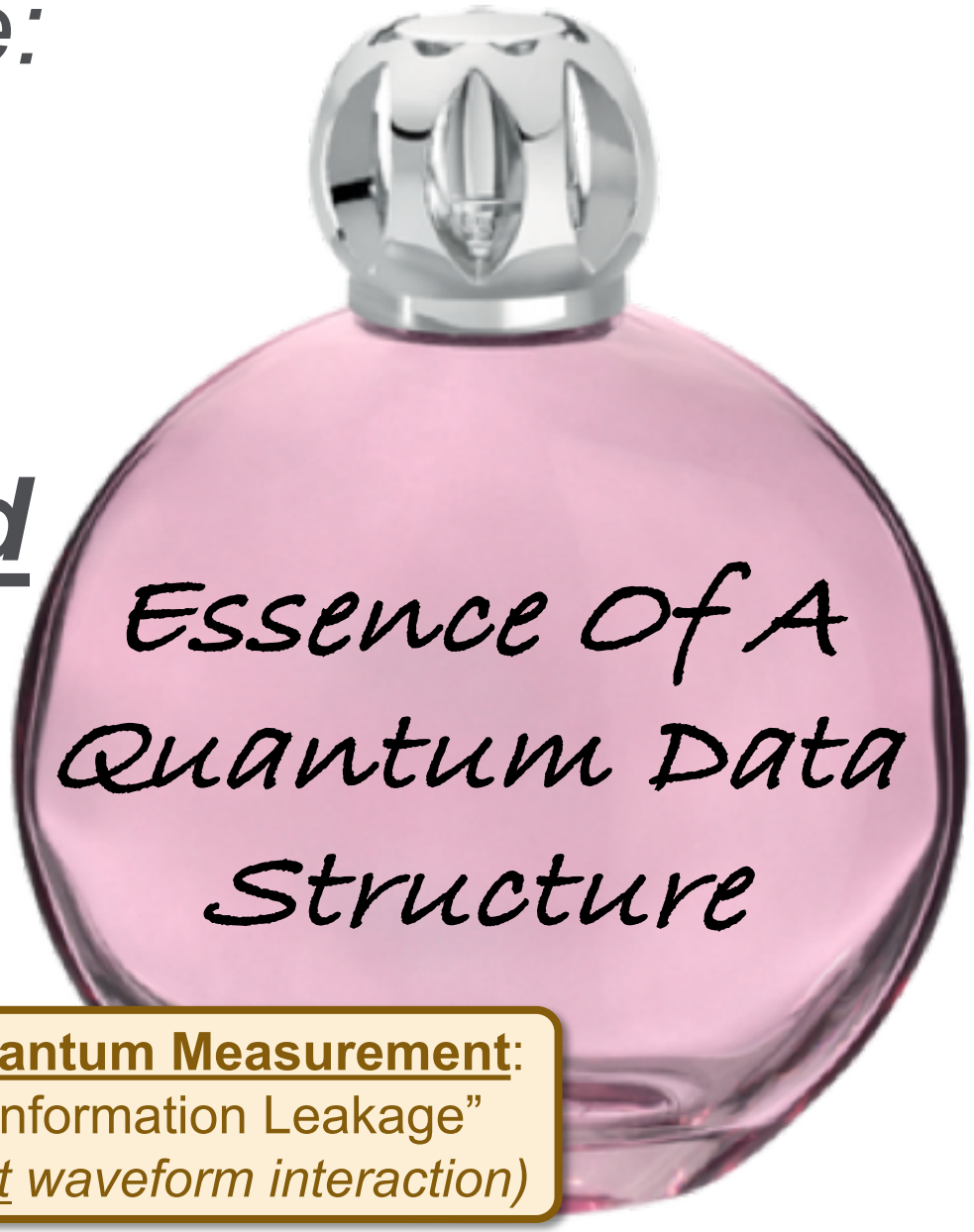
C++ now

Nature has a very strange rule:

An Object

Behaves *Differently*

When *Not Being Measured*



A ***Quantum* Object** is one that exists with a **portion of its lifecycle** that is ***not measured***.

True for:
1. ***Quantum Data Objects*** *(in Classical or Quantum Systems)*
2. ***The Natural World*** *(the Observed Quantum Universe)*

Quantum Measurement:
"Information Leakage"
(not waveform interaction)

Attributes of a *(Quantum)* Data Structure

- A quantum data structure **in a Classical system**…

1. **Has state**
   - State is required to represent entanglement through interference
   - State is always of discrete value *(for whole counts of quantum)*

2. **Transitions through quantum jumps**
   - Transactional atomic transitions to new quantum values

3. **Cannot inspect intermediate state**
   - *Pattern:* Usable, but Unknowable *(until final wavefunction collapse)*

4. **Undermines Type Safety**
   - May be arbitrarily coupled across objects of different types

5. **Has High Coupling / High Complexity**
   - Interference is achieved through entanglement with zero or more data objects
   - *Note:* A single object may be coupled to itself *(independent of "time");* or coupled to itself with respect to specific "time"

Different memory requirements *between Classical and Quantum data structure*

Type safety is largely undermined, ***type punning is expected***
(examples):
1. *Entangling a* `Date` *instance with a* `bool` *instance*
2. *Entanglement of 'qubits' (same 'type', different contexts)*
3. *Entanglement between external data object and sub-object data member*

*Possible: Compiler or API could provide some degree of "**Type-safety**" for entanglement of higher-order types*

Using a *(Quantum)* Data Structure

- **API requirements** for using a quantum data structure:

API **enables transactional atomic mutation** for quantum jumps
- *Note:* Each **mutation incurs penalties** *(for complexity, contention)*
  - This **is "cheap" in Quantum** systems *(hardware)*
  - This **can be "expensive" in Classical** systems *(hardware)*

Quantum Emulation *in Classical Systems can be expensive (slow, and demand high resources)*

(Helpful): API…
- Enables **distinction between "Eigenstate"** *(pinned)* and **"Eigenvalue"** *(superpositioned)*
- Provides clear **"source / sink" lifecycle** for quantum objects

- **Algorithms** for using a quantum data structure:

Wavefunction collapse is defined by **stateless functional algorithms**
(which mutate the entangled operands)
- Is similar to executing matrix operations:
  - *Matrix multiplication:* implies **stateless functional operation**
  - *Matrix operands:* implies *(entangled)* **operands** / **data objects**

Existing: *(Quantum)* Data Structure Patterns

- *(Existing)* **Idioms well-suited** to quantum data structures:

Familiar and Common Patterns!

Lazy Compute: Operands are "established", but not evaluated until "observed / requested" *(at which point operands are "pinned" and discarded)*

Factory: Client receives *(quantum)* object created-and-entangled by authority *(object implementation remains forever opaque)*

Future / Promise: Client receives *(quantum)* object representing a "handle" to a "promised" future state *(where observation triggers wavefunction collapse)*

PIMPL / Dynamic Ctor: Object internally changes type as it progresses with its "context" lifecycle *(each "observation" results in behavior specific to its latest accumulated payload)*

New: *(Quantum)* Data Structure Patterns

- **New Idioms** *(example):*

Likely requires exploration, research

"Functionally Quantum": Each object "observation" results in a *(logically)* independent value, but leaves the object unchanged *"entangled", (examples):*

- `RandomNumber` yields atomic value upon each "observation", but remains "entangled" with the *generator*
- `ClockTime` yields atomic value upon each "observation", but remains entangled with the *system clock*

"Sticky Quantum": Object maintains *(sticky)* "pinned" state, but remains "entangled", *(example):*

- `Sprite` is "pinned" upon each observation *(e.g., for each observed frame, its position and status is "pinned")*, but remains "entangled" *(so it can subsequently be observed for other frames-in-time)*

Similar to "double-buffering"

"Quantum" Idioms are
Consistent With Existing Practice:
These examples are "not very new"
(existing software already exhibits these patterns to some degree)

*Possibly useful for "**Progressive Heuristics**": Each observation generates "best-fit" based on continuously-updated history data (similar to CPU branch-predictors)*

Summary: Key Observations

Something Old, Something New

C++ now

Key Observations of a Quantum Data Structure

- *Notable* **SAME**:

1. Quantum Data Structure is *(very!)* **similar to existing practice** *(idioms, patterns)*
   - *Fundamental Aspect:* Defining Rules for Quantization

- *Notable* **DIFFERENCE**:

2. **Quantum Sequencing** occurs **through the data structure**
   - *Versus:* Classical sequencing occurs through the algorithm

- *Notable* **CONCERN**:

3. **Coupling** / **Complexity** is *(significantly)* **expanded**
   - *On quantum hardware:* Is closer to circuit design *(not Classical software design)*

Quantum *(vs. Classical)* results in a *(greatly!)* ***increased* importance** for the **Data Structure**, and corresponding ***decreased* importance** for the **Algorithm**

*Classical "algorithm" responsibilities are now encoded "within" the **data structure** (through quantum interference)*

Similar To Existing Practice

For Software Engineers,
Migration to Quantum is **No Big Deal™**

- Software Engineers are *(?surprisingly?)* "well-suited" to adopt quantum data structures:
  - Consistent with current **practice**, **techniques**, **idioms**, **terms**, **tools**
  - Already comfortable **defining rules for quantization**
  - Already perform **discrete processing** of quantized data
  - Already familiar with **sequencing issues** in concurrent environments
  - Already use a **highly precise notation** to describe systems
  - Already familiar using "**Time-as-a-Technology**"

…Including ***understanding limitations*** *from decisions made*

Example terms borrowed from quantum physics:
- ***Eigenstate****: "pinned" value*
- ***Eigenvalue****: "superpositioned" value*

Possible Issues:
- Some **enhancements are needed** *(terms, idioms, API convention, confidence intervals)*
- Some concerns about **increased complexity density**

Quantum Is How It Works

The Art Of Software:
Defining Rules for Quantization
How values are observed ("measured")

- **Quantum** is **how everything works**
  (the natural world, all computational hardware)
- **Software** is **already quantum**
  (everything deals with rules for quantizing values)
- **Quantization** is already **Current Practice**
  (everything software engineers do is to control quantization)

"Quantum": Ranges are composed of discrete values
(not continuous values)

A *Really Big*™ Caveat:
Programming today's quantum computers *(hardware)* is ***VERY DIFFERENT***
(…next slide…)

What It's Like: Quantum Software On Quantum Hardware

Quantum Software: Different On *Quantum Hardware*

A *Really Big*™ Caveat:

Programming today's quantum computers *(hardware)* is ***VERY DIFFERENT***

Is like authoring "**`constexpr` lock-free algorithms**" ← *Tedious, tricky, hard-to-debug*

…where software is closer to circuit design, not so much software design

Why So Hard? *(Software Programming of Quantum Hardware)*

Because: Wavefunction Collapse demands the **data structure itself** establishes **fixed sequencing** *(through entanglement / quantum interference)*

1. **Cannot rely upon "ordered sequencing"** through imperative logic *(sequencing is in the data structure, not the algorithm)*
2. **Wavefunction Collapse** *(literally)* **takes place "outside the bounds of Time"** *(effect can precede cause)*

quantum jump

- This is *similar to*: **Circuit Design** ← *i.e., "**Fixed Runtime**"*
- This is *different from*: **Imperative algorithm execution** based on **runtime state** *(value)* classification ← *i.e., "**Runtime Conditional Processing**"*

Sequencing In The Data Structure (*Not Algorithm*)

Classical Sequencing: Performed by ***algorithm***
Quantum Sequencing: Performed by ***data structure***

*Through conditional branch based on **runtime value classification**; and limitations on **instruction (re-)ordering***

Classical *(Historical):*

- **Software** manages sequencing issues through **imperative execution**
- **Hardware** manages sequencing issues through **dependency references**

*Through **circuit design***

Software for quantum hardware:

1. **Cannot "inspect" value** for runtime classification *("Usable, but Unknowable")*
2. **Cannot conditionally process** based on runtime classification *(state inspection is destructive)*
3. **Cannot rely upon imperative sequencing** *("outside the bounds of Time", "effect precedes cause")*
4. **Is Non-Deterministic** *(has "unknowns" due to probabilities, and unquantified external influences)*

(Using Classical ordering) **Classical** system **can *emulate* quantum sequencing**
(possible, but computationally and memory-expensive)

Quantum Sequencing

- Logically Opposite:

Functional Processing
- Through *First Principles*, value is **computed through functional evaluation** of First Principle State *(which remains unchanged)*

*Functional: Based on **NO side-effect***

Quantum Processing
- Through *quantum interference*, value is **computed through wavefunction collapse** *(which causes mutation)*

*Quantum: Based on **side-effect***

The **Quantum Data Structure** is *(logically)* a "Classical Object on steroids" that **seizes control of all sequencing** for **self-mutating side-effect** upon that object

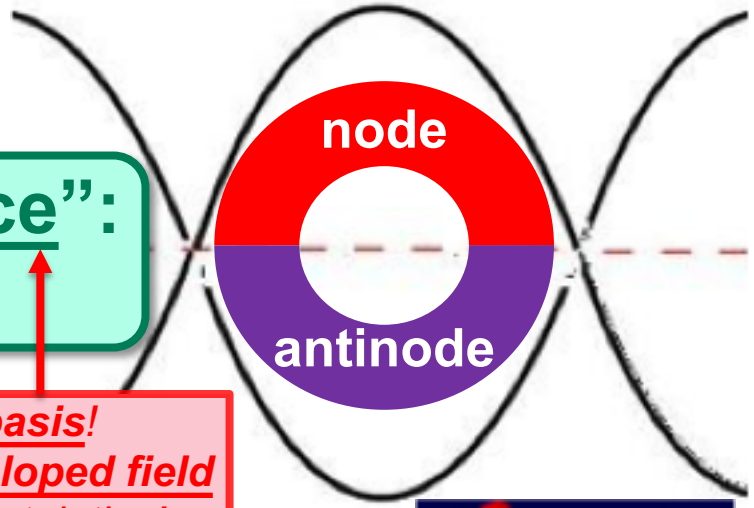
*Quantum Sequencing is **through the data structure** (not algorithm)*

Final Thoughts

Managing Multi-Dimensional Complexity

C++ now

Welcome To The "Real" World

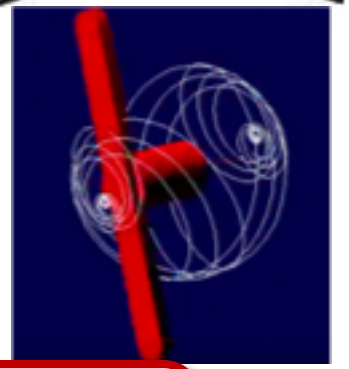


Quantum Science can be seen as "fringe science":
Operates in areas that are *(very)* difficult to test

*Has **solid basis**!*
*But, is an **undeveloped field** with many uncertainties!*

- **You Are Prepared!**

Software Engineers already do "quantum thinking"
(more so than any other engineers)



Most Significant Change embracing quantum data structures seems to be:

1. **Increased complexity** *(higher state density, and higher computational density)*
2. Increased **priority for some idioms** *(e.g., "lazy-compute")*
3. **New / Unfamiliar idioms** *(e.g., "Usable" but "Unknowable" – somewhat present in classical concurrent systems; but addressed by developers with specialized skills)*

Expanding Our Quantum Thinking

*C++ approximates "**precise executable notation**" (system description that executes)*

- **"Quantum Thinking"** in software seems to **require consideration of**:

1. *(Highly)* **Precise Notation** *(to describe quantum data structures, their current "state" and/or "status")*
   - C++ is particularly good at this: pointers, references, object model

2. *(Highly)* **Precise Terminology** *(example):*
   - **Eigenstate**: a "pinned" value *(the spin is "up")*
   - **Eigenvalue**: a "superpositioned" value *(the spin is one of "up" or "down")*

Terms borrowed from quantum physics

*Note: Quantum Physics established its own (math-heavy) precise **notation**, **terminology***

3. **Sequencing** the quantum data object *(lifecycle, wavefunction collapse)*
   - ***Why:*** Because of complexities related to reliance upon: lazy-compute, complex state mutation model, concurrent resource access among coupled state *(inter-dimensional and independent of time)*
   - ***Useful tools:*** ctor…dtor, object model, C++ Language sequencing rules

Essential Capability

4. **Conventions** Within Your System / Domain
   1. What is the "state"? *(i.e., possible "eigenstates")*
   2. What is the "entanglement"? *(e.g., references within the object model)*
   3. What is the create / destroy lifecycle? *(i.e., management of object lifetime)*

*These are Nothing New! But (for quantum) are **Profoundly Essential**!*

We (*Software*) Have It Easy!

If you want to see "Hard", check out what the ***Quantum Hardware*** *people are doing!*

"Topological qubit"

A future talk

"Atomic Nuclei Spins"

Black Projects

?!

"Superconducting qubit"

"Quantum annealing"

https://en.wikipedia.org/wiki/List_of_companies_involved_in_quantum_computing_or_communication

C++ now

Thank you for coming!