Analyses
For The Lazy Superhero

PL Fencing Day

- Friday Dec 4 (unless it rains) @ 4:30pm
- Darden Courtyard; no experience necessary
- I provide gear
- I provide lesson

Final Exam Construction

- No Time Limit
- Closed Book, 2 sides of notes
- High level “take home message” questions

- Email me questions you think are relevant.
- Also, what do you want to hear about on the last day of class? Any topic!
- Grade predictions mailed out Fri or Sat
  - Hint: ...

Debugging Experiment

- Takes about one hour
- Identify fault (or not) in Java code
  - How difficult was the code to debug?
- Worth 2 points of extra credit on the final
  - ... but still anonymous.
- Research by Zak and Wes

- Monday – Mani's Day
  - The Latin days of the week in imperial Rome were named after the planets, which in turn were named after gods (see discussion at WEEK n.). **In most cases the Germanic names have substituted for the Roman god's name that of a comparable one from the Germanic pantheon.** In the case of Monday (as also of Sunday), the name of the planet (as the moon was considered in the classical period) and the god were the same. [OED]
  - Monday: Old English Mōnandæg (pronounced [mon.nan.dæg] or [mon.nan.dæj'), meaning "Moon's Day". This is likely based on a translation of the Latin name Dies Lunae (cf. Romance language versions of the name, e.g., French Lundi, Spanish, Lunes, Romanian Luni, Italian Lunedì). **In North Germanic mythology, the moon is personified as a god; Máni.** [Wikipedia]
- Tuesday – Tyr's Day
  - ... The sky-god Tiw was originally also a god of war, like the Roman Mars. [OED]
  - Tyr (pronounced /ˈtɪər/;[1] Old Norse: T˝r [ty:r]) is the god of single combat, victory and heroic glory in Norse mythology, portrayed as a one-handed man. Corresponding names in other Germanic languages are Gothic Teiws , Old English Tīw and Old High German Ziu, all from Proto-Germanic *Tîwaz. [Wikipedia]
- Wednesday – Odin's Day
  - The identification of Woden, the highest god of the Germanic pantheon, with the Roman Mercury is already suggested in Tacitus, although he does not give a Germanic name (Deorum maxime Mercurium colunt, 'they worship Mercury most of all the gods', Germania 9); common features between these gods include eloquence, swiftness, range of travel, and guardianship of the dead. ... The name of Woden (Old English W{omac}den, Old Saxon W{omac}den, Old High German Wuotan, Old Icelandic Ó{edh}inn) derives < the Germanic base of WOOD adj. + a nasal suffix. [OED]
  - Wotan, the High German variant of Wōden, the Continental West Germanic god corresponding to Norse Odin. [Wikipedia]
- Thursday – Thor's Day
  - The contemporary name comes from the Old English Þunresdæg, "Thunor's Day"[1][2][3] (with loss of -n-, first in northern dialects, from influence of Old Norse Þorsdagr, meaning "Thor's Day"). Thunor and Thor are derived from the Proto-Germanic god Thunaraz, god of thunder. [Wikipedia]
- Friday – Frig's Day
  - The name of Frig (Old English Fr{imac}g, Old High German Frija, Old Icelandic Frigg, in Icelandic mythology the consort of Odin) is attested in Old English only in the name of the day of the week ... [OED]
  - The name Friday comes from the Old English frīgedæg, meaning the day of Frige the Old English form of Frigg, a West Germanic translation of Latin dies Veneris, "day (of the planet) Venus." However, in most Germanic languages the day is named after freyja—such as Frīatag in Old High German, Freitag in Modern German, Freyjudagr in Old Norse, Vrijdag in Dutch, Fredag in Swedish, Norwegian, and Danish—but Freyja and Frigg are frequently identified with each other. [Wikipedia]
- Sunday – Sun's Day
  - Sunday: Old English Sunnandæg (pronounced [sun.nan.dæg] or [sun.nan.dæj), meaning "Sun's Day". ... In both West Germanic and North Germanic mythology the sun is personified as a goddess; Sunna/Sól. [Wikipedia]
- General:
  - Saturday: the only day of the week to retain its Roman origin in English, named after the Roman god Saturn associated with the Titan Cronus, father of Zeus and many Olympians.

Random Interpretation

Sumit Gulwani & George Necula

Probabilistically Sound Program Analysis!

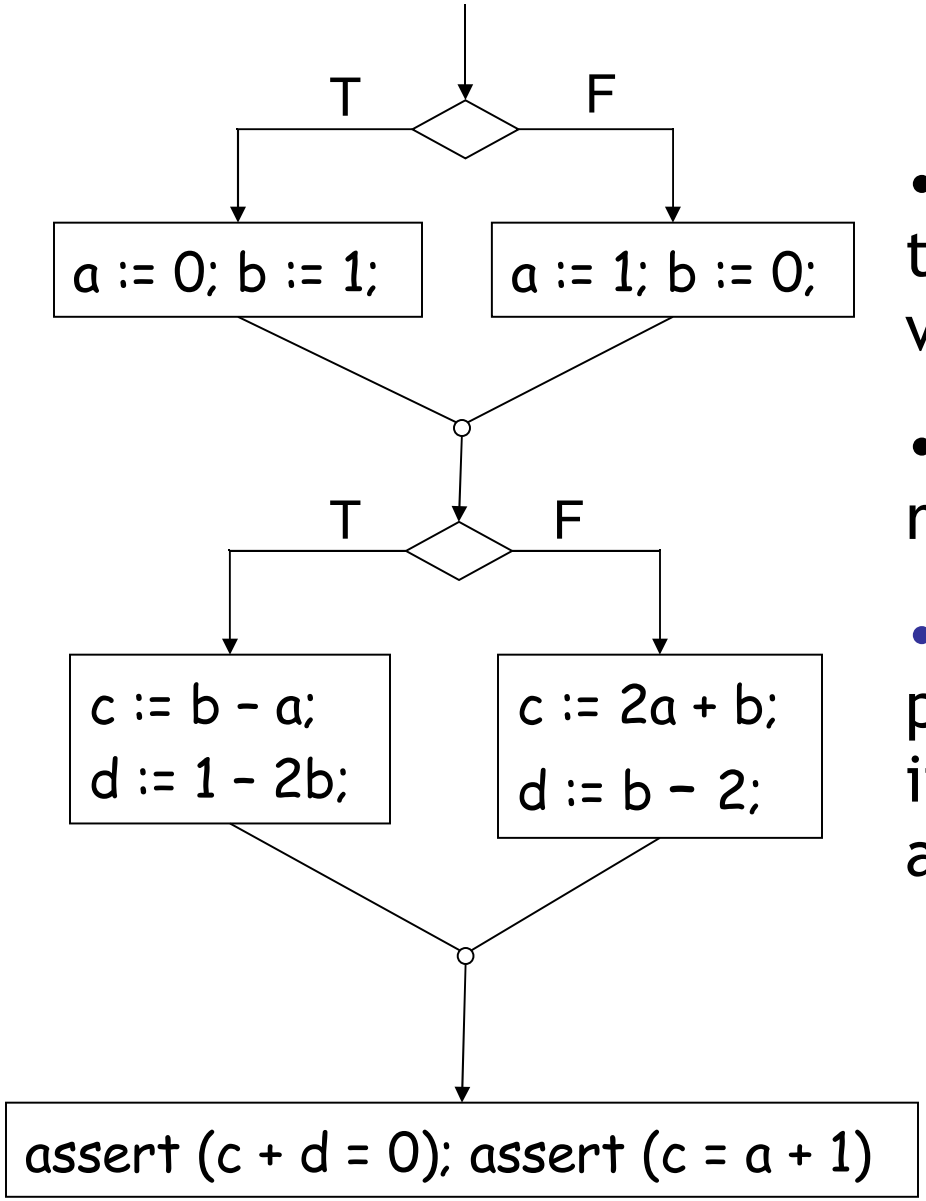
- Sound program analysis is hard (Rice's Theorem)
- PL researchers usually pay in terms of
  - Loss of completeness or precision
  - Complicated algorithms
  - Long running times

- Can we pay in terms of **soundness** instead?
  - Basically, *soundness* = *correctness*
  - Judgments are unsound with low probability
  - We can predict and control the probability of error
  - Can gain simplicity and efficiency

Discovering Affine Equalities

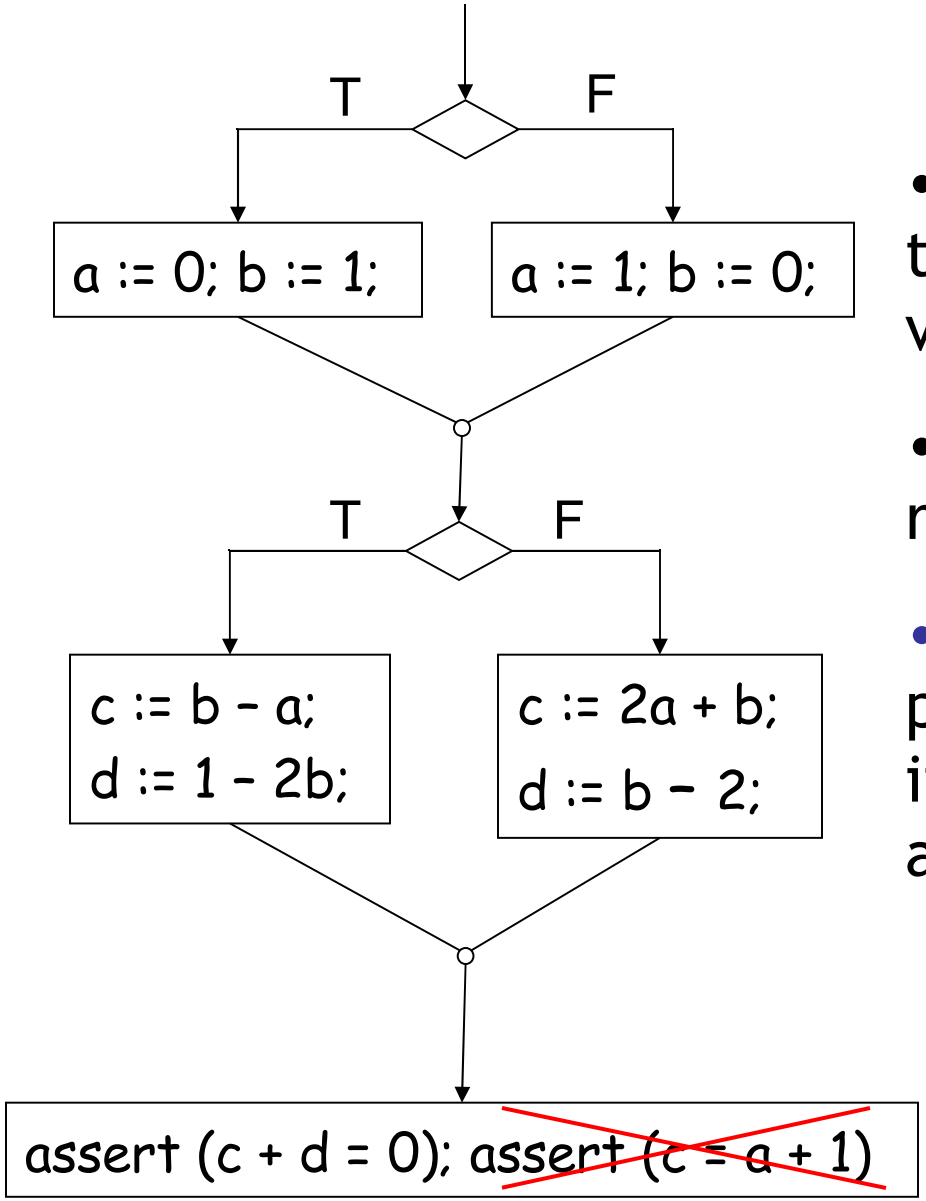
- Given a program (control-flow graph) …
- Discover equalities of the form $2y + 3z = 7$
  - Compiler Optimizations
  - Loop Invariants
  - Translation Validation
- There exist polynomial time deterministic algorithms [Karr 76]
  - involving expensive operations - $O(n^4)$
- We present a randomized algorithm
  - as complete as the deterministic algorithms
  - but faster - $O(n^2)$
  - and simpler (almost as simple as an interpreter)

Example 1



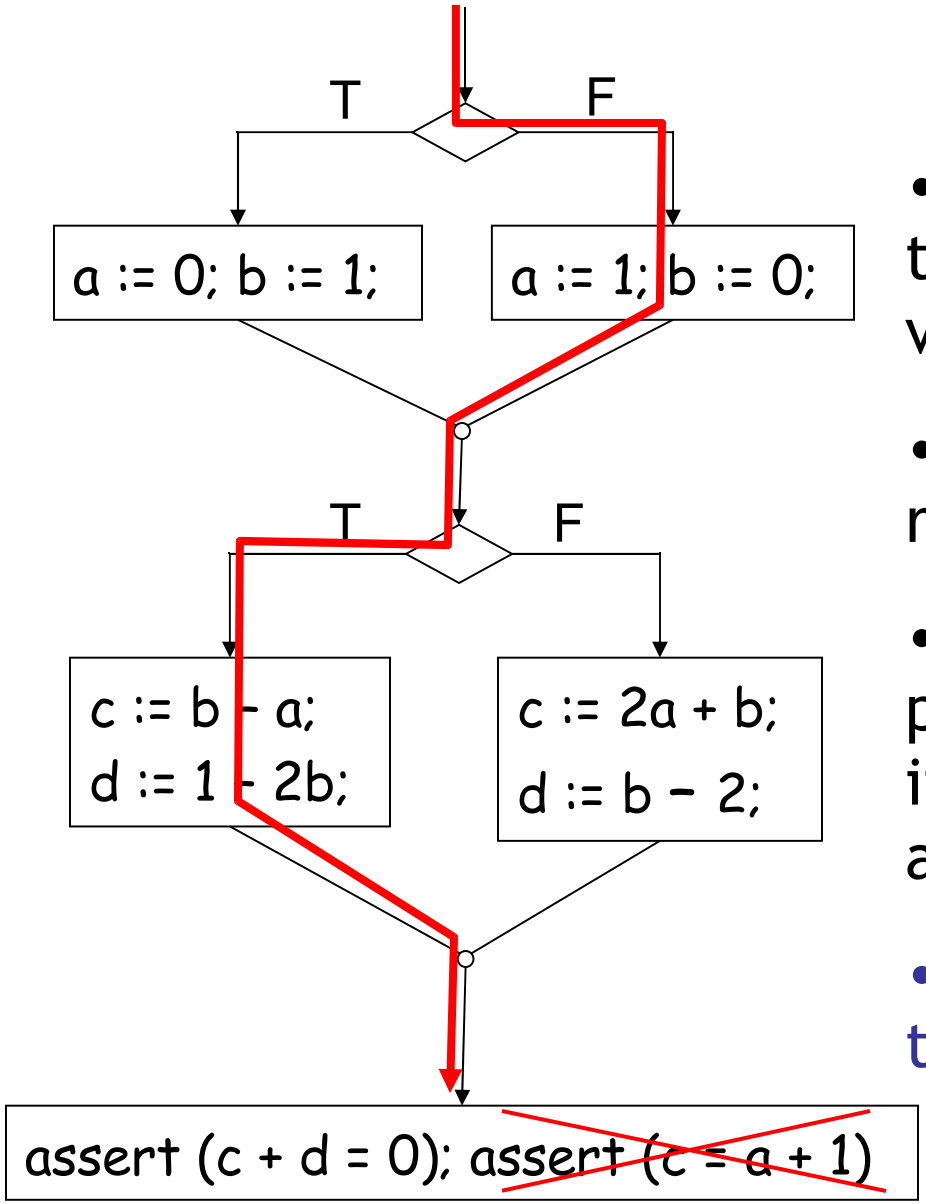
- Random testing will have to exercise all the 4 paths to verify the assertions
- Our algorithm is similar to random testing
- However, we execute the program once, in a way that it captures the “effect” of all the paths

Example 1



- Random testing will have to exercise all the 4 paths to verify the assertions
- Our algorithm is similar to random testing
- However, we execute the program once, in a way that it captures the “effect” of all the paths

Example 1

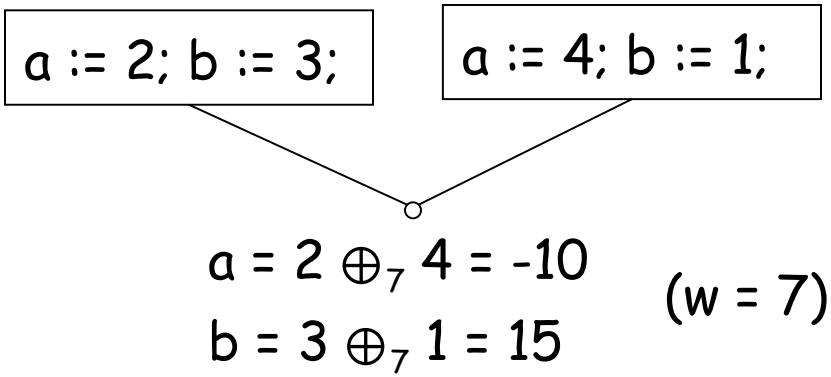


- Random testing will have to exercise all the 4 paths to verify the assertions
- Our algorithm is similar to random testing
- However, we execute the program once, in a way that it captures the "effect" of all the paths
- Exponential work, linear time! *(P=NP?)*

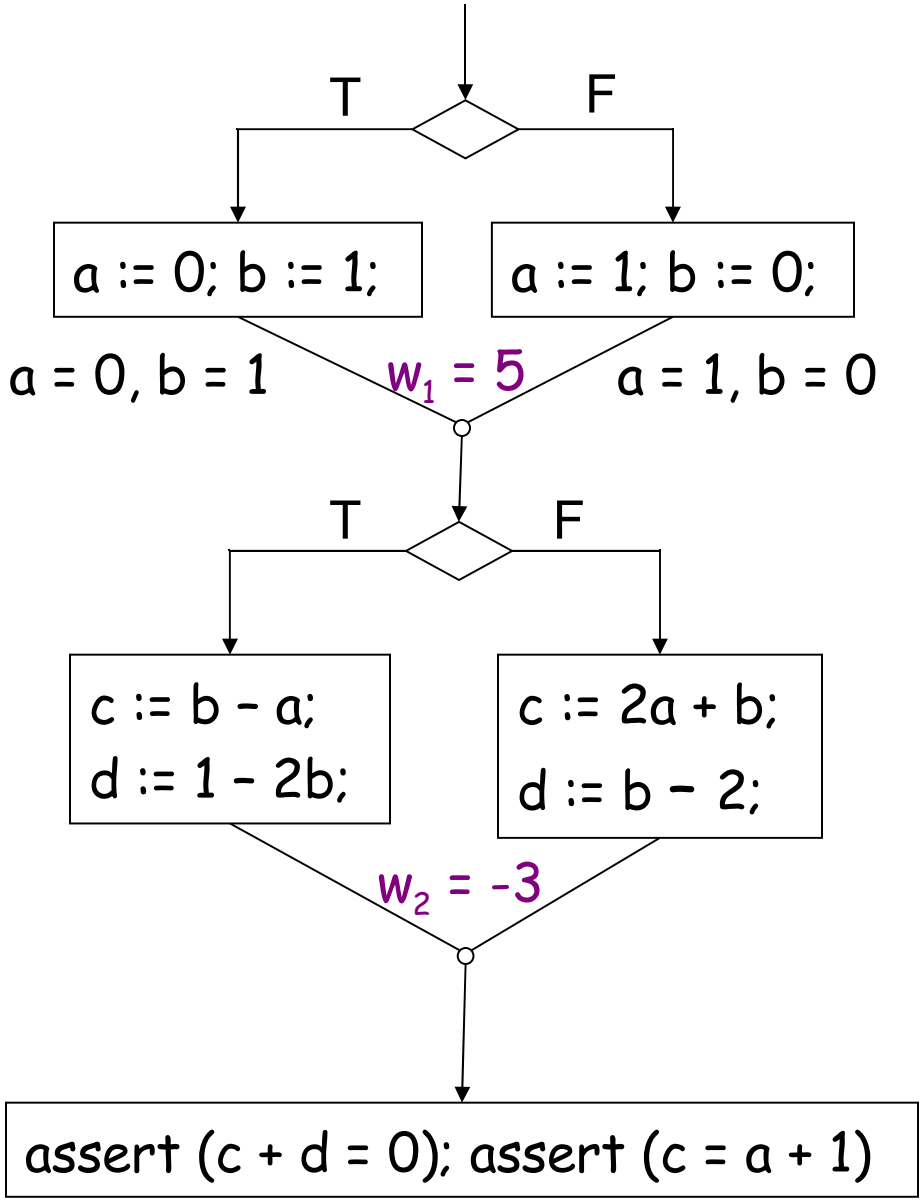
Idea #1: Affine Join Operation

- Execute both the branches
- Combine the values of the variables at joins using the affine join operation $\oplus_w$ for some randomly chosen w

$$v_1 \oplus_w v_2 = w \times v_1 + (1-w) \times v_2$$

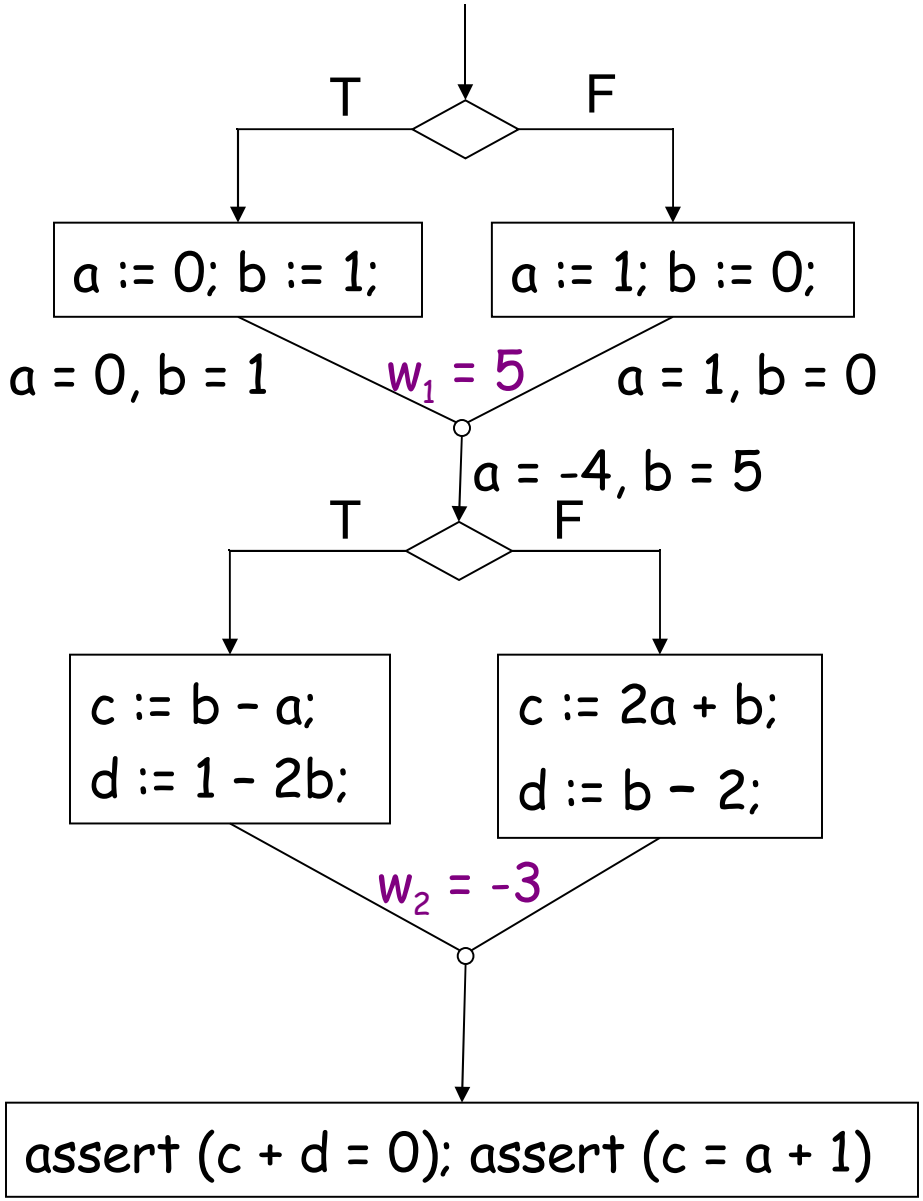


Example 1



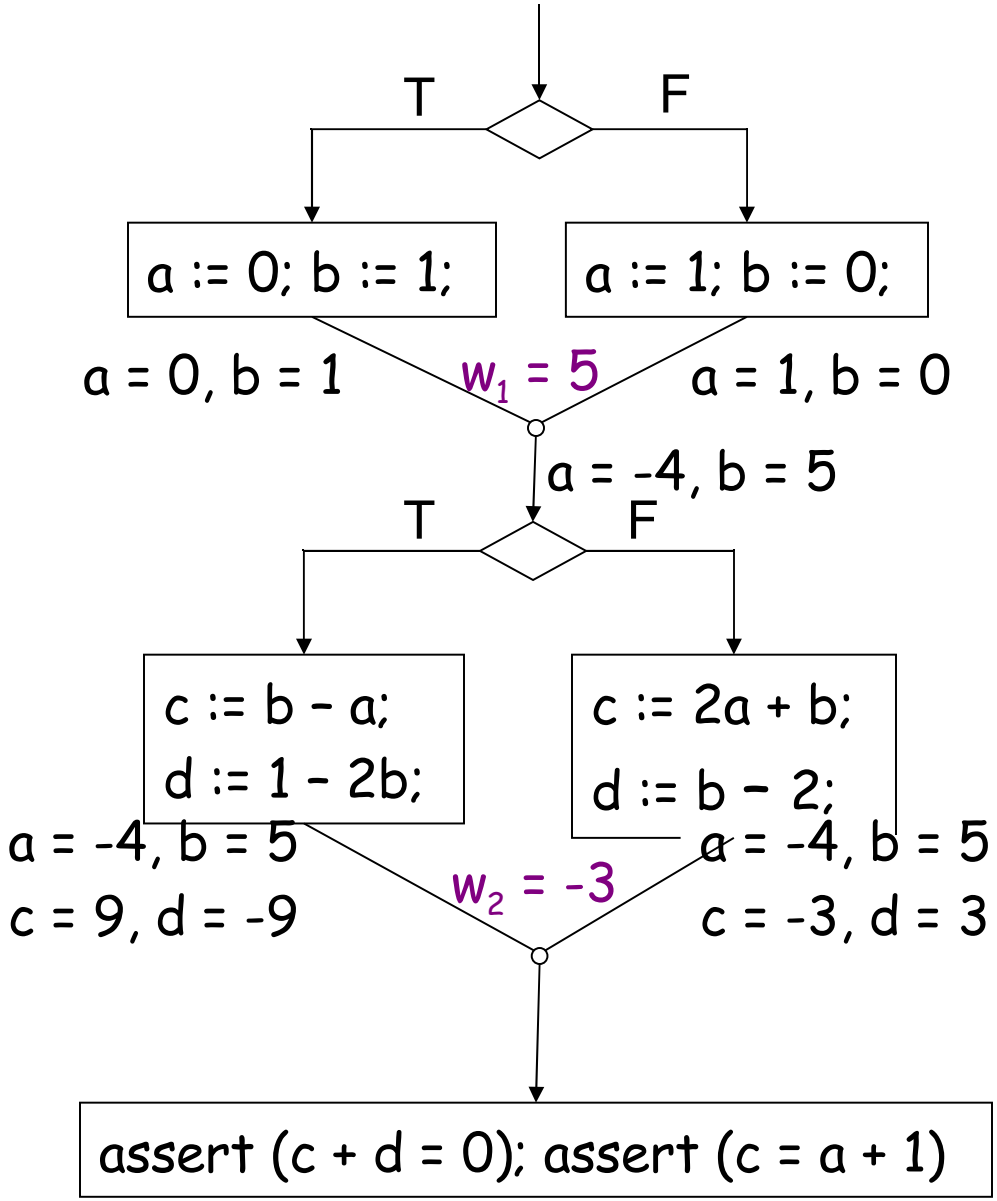
- Choose a random weight for each join independently.
- All choices of random weights verify the first assertion
- *Almost* all choices contradict the second assertion.

Example 1



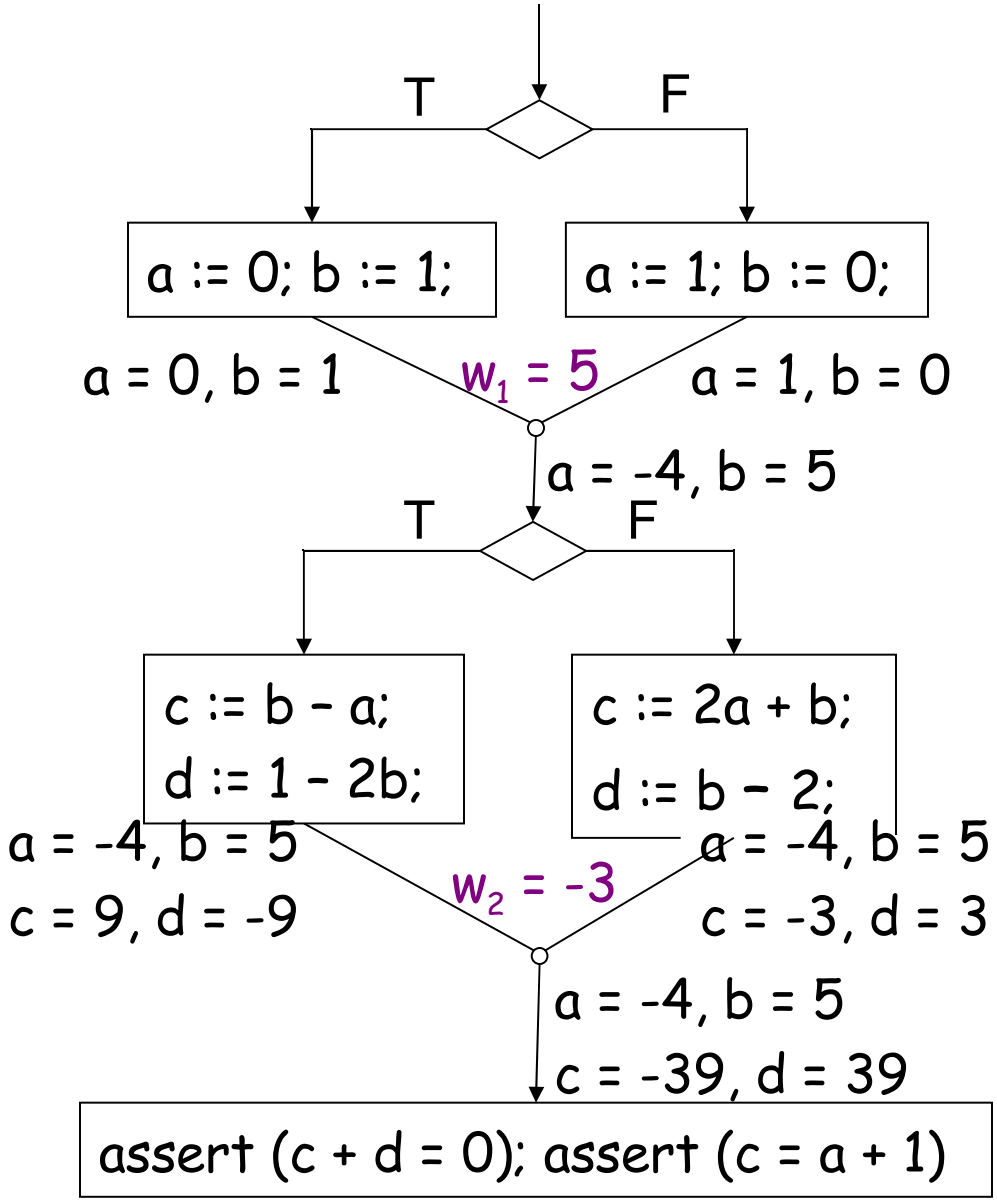
- Choose a random weight for each join independently.
- All choices of random weights verify the first assertion
- *Almost* all choices contradict the second assertion.

Example 1



- Choose a random weight for each join independently.
- All choices of random weights verify the first assertion
- *Almost* all choices contradict the second assertion.

Example 1

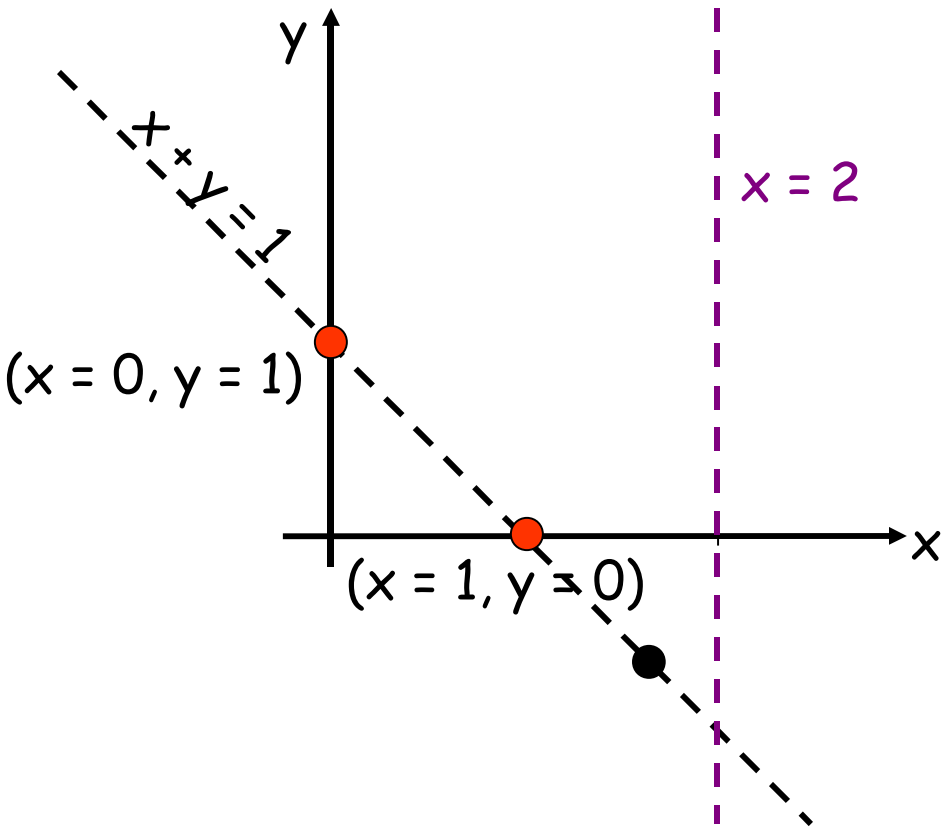


- Choose a random weight for each join independently.
- All choices of random weights verify the first assertion
- *Almost* all choices contradict the second assertion.

Geometric Interpretation of the Affine Join operation

● : State before the join

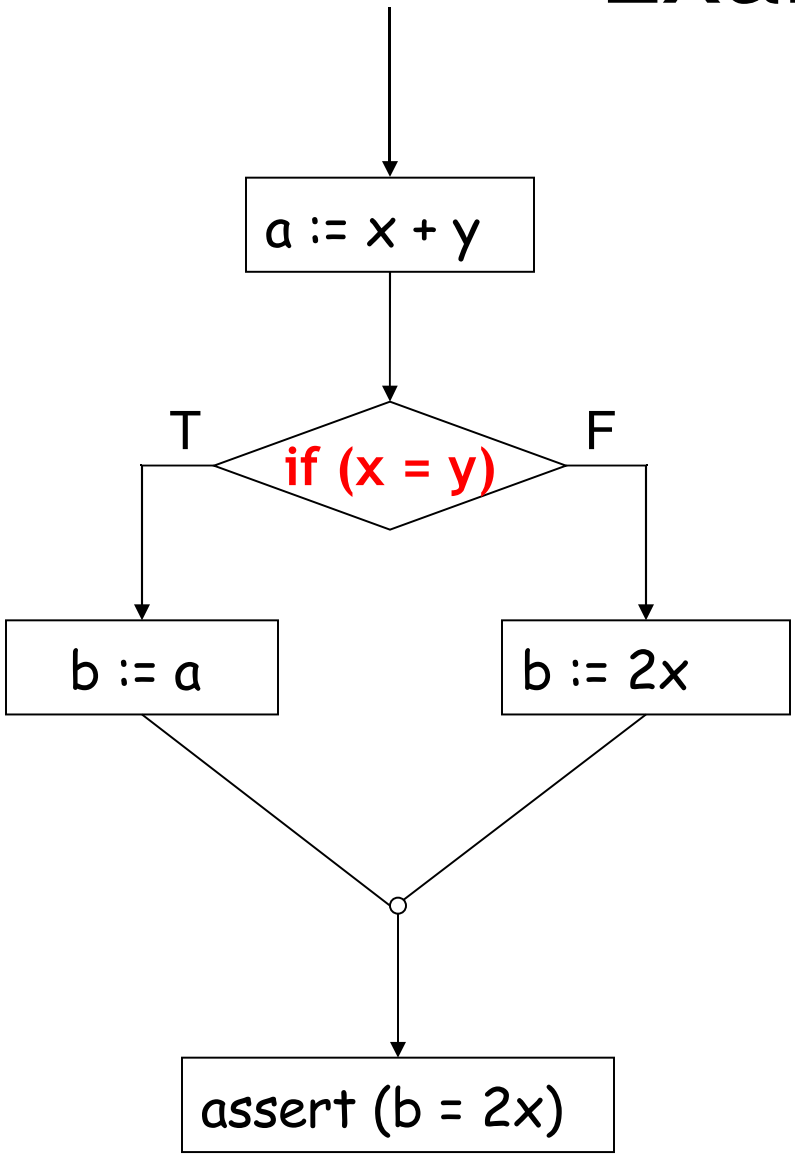
● : State after the join



● satisfies all the affine relationships that are satisfied by both ●
(e.g. x + y = 1, z = 0)

Given any relationship that is *not* satisfied by any of ● (e.g. x=2), ● also does not satisfy it <u>with high probability</u>

Example 2

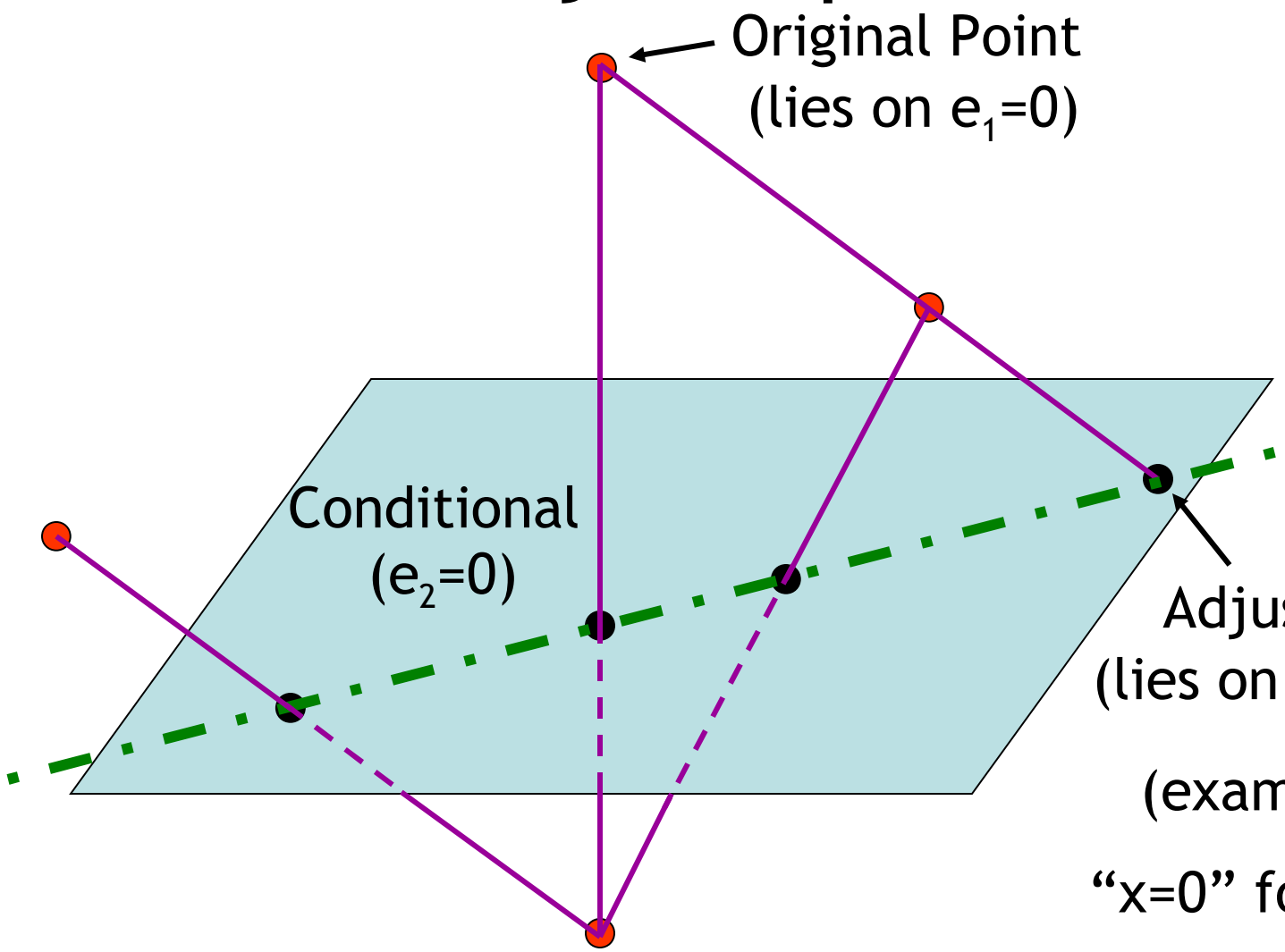


- Idea #1 is not enough
- We need to make use of the conditional x=y on the true branch

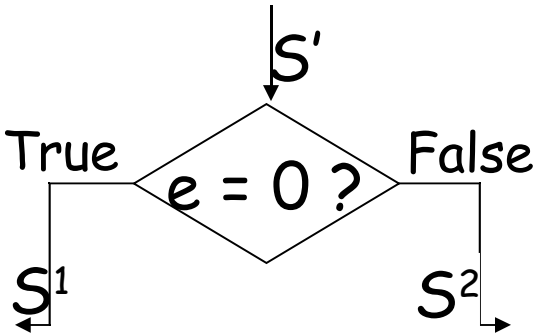
Idea #2: Adjust Operation

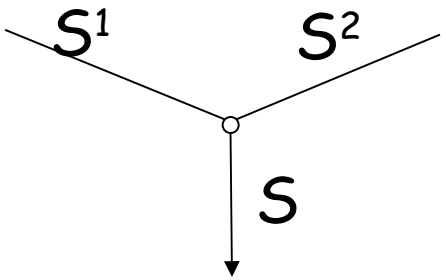
- Execute multiple runs of the program in parallel
- "Sample" = Collection of states at each program point
- "Adjust" the sample before a conditional (by taking affine joins of the states in the sample) such that
  - Adjustment preserves original relationships
  - Adjustment satisfies the equality in the conditional
- Use adjusted sample on the true branch

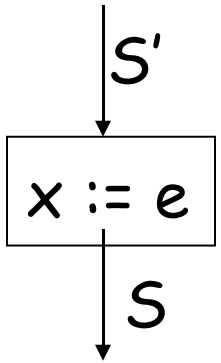
Geometric Interpretation of the Adjust Operation



The Randomized Interpreter R





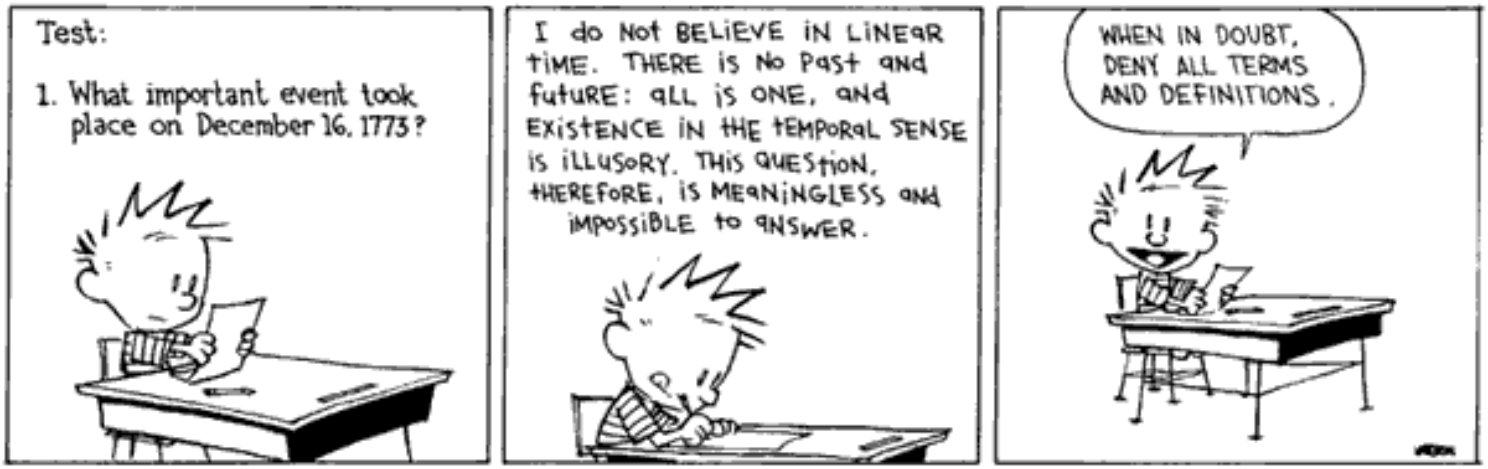


$S^1 = Adjust(S', e)$

$S^2 = S'$

$S_i = S^1{}_i \oplus_{wi} S^2{}_i$

$S_i = S'_i[x \leftarrow e]$



Completeness and soundness of R

- We compare the randomized interpreter R with a suitable actual interpreter A
  - Actual Interpreter A would be too slow (etc.) to use in real life!

- R mimics A with high probability
  - R is as complete as A
  - R is sound *with high probability*

Soundness Theorem

- If $A \not\Rightarrow g = 0$, then *with high probability* $R \not\models g = 0$
- Error probability $\leq (2d)^b \left(\frac{j+1}{d}\right)^r$
  - b: number of branches
  - j: number of joins
  - d: size of the field
  - r: number of points in the sample
- If j = b = 10, r = 15, $d \approx 2^{32}$, then error probability $\leq \frac{1}{2^{98}}$

Conclusions, Wessy Summary

- Randomization can help achieve simplicity and efficiency at the expense of making soundness probabilistic
- Has been extended to handle uninterpreted function symbols, interprocedural analyses, randomized decision procedures for theorem proving, combined abstract interpreters, …
- May help with complicated security analyses
- Go to grad school!

Q: TV (070 / 842)

- This 1993-1998 Emmy Award-winning CBS series featured Jane Seymour as Dr. Mike. It was originally targeted at the 18-to-40 demographic but ended up viewed predominantly by women aged 40 or older.

Q: Music (123 / 842)

- Name the song: *"He was a famous trumpet man from old Chicago way. He had a boogie style that no one else could play. He was the top man at his craft, but then his number came up and he was gone with the draft. He's in the army now a-blowin' reveille."*

Q: Music (205 / 842)

- This 19th century Russian composer created the piano suite **Pictures At An Exhibition**. It includes a "Tuileries" theme depicting children playing in a garden near the Louvre.

Q: Movies (267 / 842)

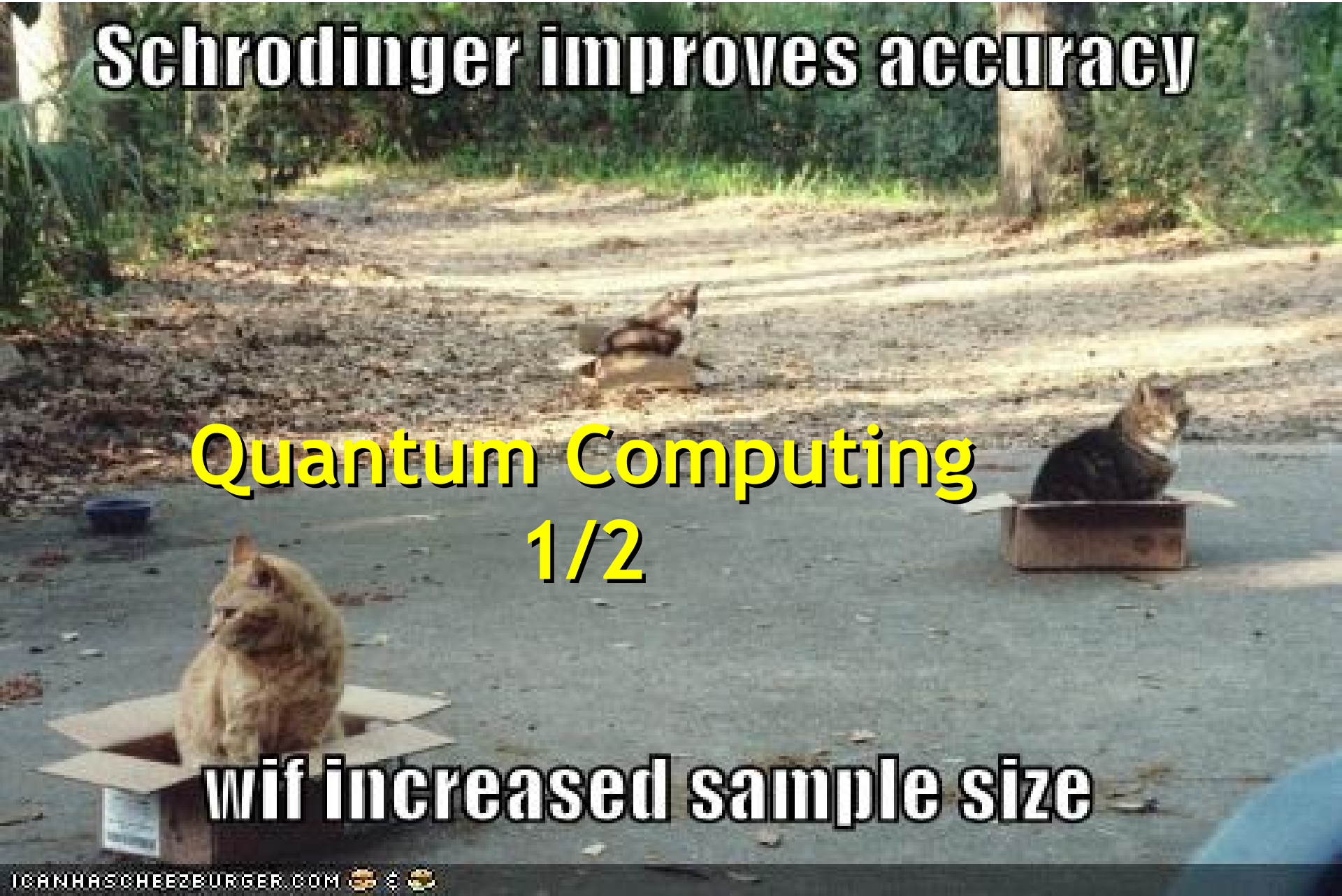
- Name the movie described below, its heroine and its star. This 1979 Ridley Scott movie began the first major American film series with a female action hero.

Q: Movies (322 / 842)

- In this 1984 movie one of the villains is the Stay Puft Marshmallow Man.

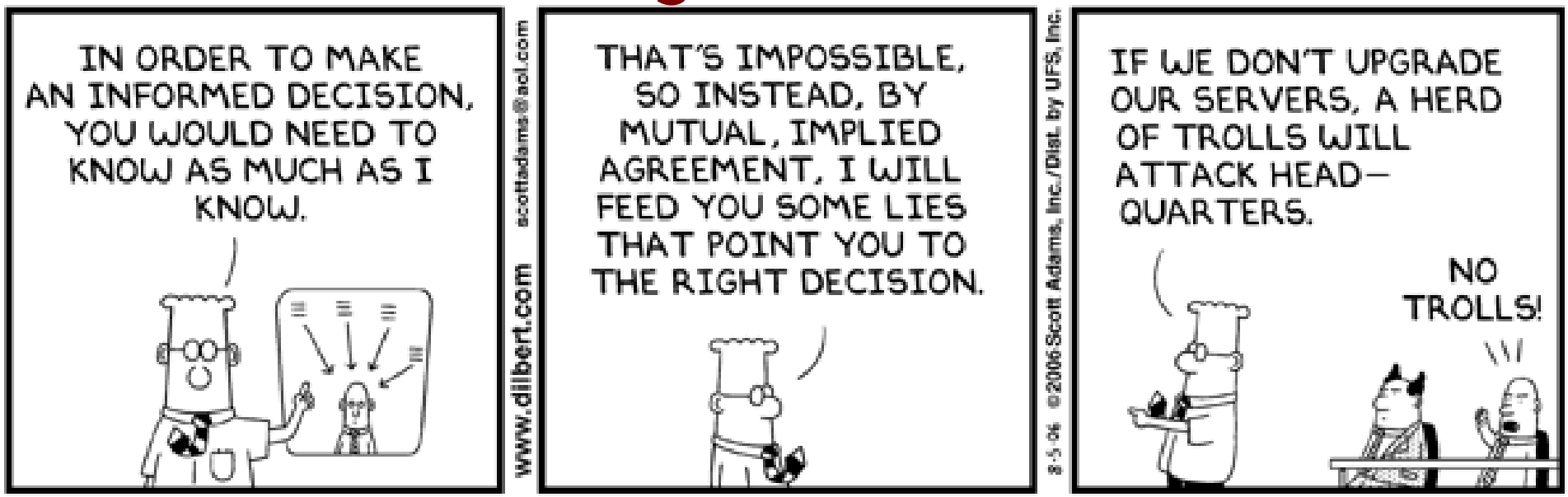
Q: Movies (374 / 842)

- In this 1989 family comedy the children of Rick Moranis spend most of the day playing with bugs in the yard.


Schrodinger improves accuracy
Quantum Computing
1/2
wif increased sample size
ICANHASCHEEZBURGER.COM

Lies

- This lecture will gloss over most details.



© Scott Adams, Inc./Dist. by UFS, Inc.

- You actually know more than I know, I just don't have time to get into it.
- **Math Class: Intro To Quantum Computing**

One-Slide Summary

- A **quantum computer** manipulates **quantum bits**; such qubits can represent a **superposition** of possible states.
- Quantum computers are **probabilistic**. **Grover's Algorithm** (for linear search in sub-linear time) and **Shor's Algorithm** (for factoring integers in polylog time) are common quantum algorithms.
- When you use a quantum computer to “try everything in parallel” you get back a **random answer**.

Theory and Practice

- Quantum Computers do not yet exist
- But we can still talk about them in theory
- Serious experimental work remains to be done!

Quantum Computers

- A **quantum computer** uses quantum effects, such as superposition or entanglement, to perform operations on data.
  - Quantum Computers ***do not exist*** (yet).
- Classical Bit: 0 *or* 1
- Quantum Bit or **Qubit**: 0 *or* 1 *or* quantum superposition of 0 and 1
  - 3 Bits = exactly one state (= one number 1-8)
  - 3 Qubits = up to 8 states simultaneously!

Quantum Digital Logic Design

- Qubits are manipulated by **quantum gates**
  - Just as AND, OR and NOT gates manipulate bits
  - There is a notion of a complete set of gates
    - AND, OR and NOT for classical
    - Hadamard, Phase Rotation and Controlled Not for Q
- Quantum gates are unitary matrices
  - Each quantum gate is **reversible**
  - This is not obvious
- One builds quantum circuits out of quantum gates

Quantum Storage

- A **quantum register** (qregister) holds qubits
  - A qregister is described by a wavefunction where the coefficients are *complex* numbers whose amplitudes squared are the probabilities to measure the qubits in each state ...
  - Complex numbers means the phases can constructively and destructively **interfere**
- Recording or simulating the state of a qregister requires an **exponential** number of classical complex numbers
  - 3-qubit qregister == 8 complex numbers
  - 300-qubit qregister == 10^{90} > |universe|

Quantum Computation

- Initialize n-qubit qregister to starting values
- Each step: n qubits go through quantum gates
  - (i.e., are multiplied by unitary matrices)
- Read qregister via **quantum measurement**
  - Yields a ***random*** n-bit string
  - And ***destroys*** the stored state as well
- This sounds **bad**
  - But: run whole program many times (i.e., sampling)
  - Probability distribution of output string is skewed in favor of the right answer

Probabilistic

- Quantum Computers are **probabilistic**
  - Repeated runs of qcomputer plus majority polling of the outputs yields the right answer with high probability
- They are *not* deterministic
  - Officially, they can get the "wrong" answer, just like that probabilistic program analysis from last week
    - Exceptions: Deutsch-Jozsa, etc.
  - In practice this is not a problem: rerun until prob of error is less than prob of earth exploding, etc.

Database Search Problem

- **Database Search Problem:**
  - To Solve: guess random answers and check them
  - There are ***n*** possible answers
  - Each guess takes the same time to check
- Example Uses:
  - Searching for an entry in an unsorted array
  - Guessing your friend's password
  - Attacking symmetric ciphers (AES, 3DES)
- Classical Running Time: **O(n)** linear search
  - Average Case: (n+1)/2 guesses to find answer

Enter Lov Grover in 1996

- **Grover's Algorithm**
  - Quantum algorithm for solving the database search problem (i.e., for doing linear search)
  - Takes **$O(\sqrt{N})$** time (yes, sqrt(N) time!)
  - Uses **$O(\log_2 N)$** qubits of storage
  - Probability of measuring wrong answer: **$O(1/N)$**
  - Algorithm is optimal
- I will not explain Grover's Algorithm.
- Instead, we will cover Shor's Algorithm.

Let's Break RSA

- Break RSA == find factors of large integer ***N***
- No known polynomial classical algorithms
  - General Number Field Sieve: ~ $O(2^b)$
  - ***b*** is number of bits in ***N*** (e.g., *b*=512)
- Could use Grover
  - Linear search for factors in **$O(\sqrt{2^b}) = O(2^{b/2})$** time
  - Still too slow!
- New Proposed Quantum Approach
  - Try all factors in parallel, pick the right one!
  - Does this work?

Itty Bitty Living Space

- Despite popular rumors, quantum computers *cannot easily* "try everything in parallel"
- Sure, you can try everything in parallel
- But when you measure the outcome, you get something **random**!
- In this case, you'd get a random (non-)divisor
- Which is not what we want!
  - Using the power of quantum computing to look for a needle in a haystack is not efficient!
- So we must exploit the structure of the problem

Needles In Haystacks

- Quantum computing can give you a random answer back
  - So the trick is to make sure that even a random answer will help you
  - By making sure that all answers have some important property that contributes to what you really want to know
- Look: if you think about quantum computing in terms of “parallel universes” (and whether you do or don’t is up to you), there’s no feasible way to detect a single universe that’s different from all the rest. Such a lone voice in the wilderness would be drowned out by the vast number of suburb-dwelling, Dockers-wearing conformist universes. What one can hope to detect, however, is a joint property of all the parallel universes together — a property that can only be revealed by a computation to which all the universes contribute. [Scott Aaronson]

Digression: Periodic Sequences

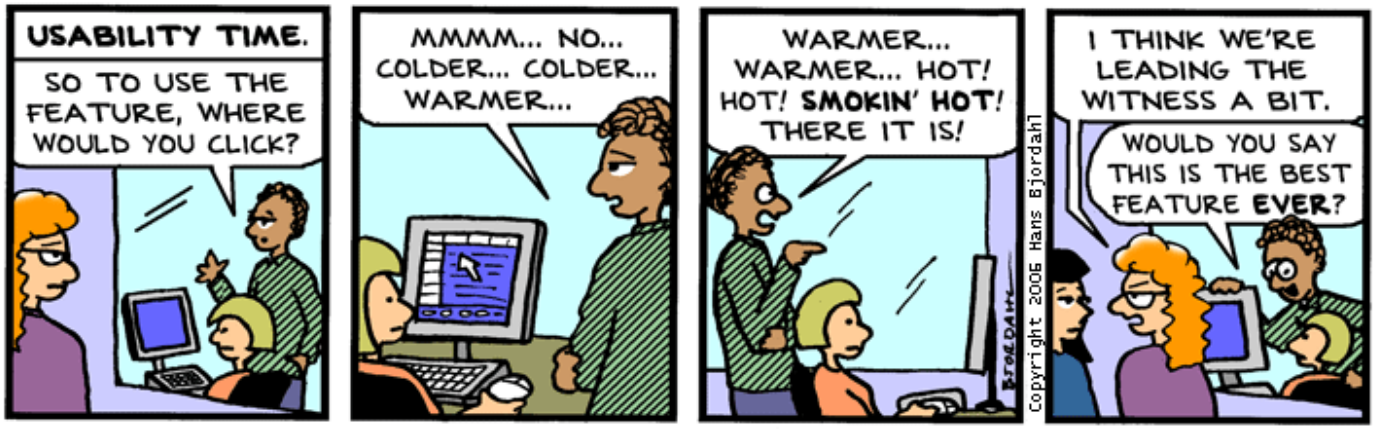
- Powers of Two: 2, 4, 8, 16, 32, 64, 128, 256, ...
  - Not a periodic sequence.
- Powers of Two Mod 15: 2, 4, 8, 1, 2, 4, 8, 1, ...
  - **Period** is 4
- Powers of Two Mod 21: 2, 4, 8, 16, 11, 1, 2, 4, ...
  - **Period** is 6
- Let N be the product of two primes p * q
  - Series: x mod N, x^2 mod N, x^3 mod N, x^4 mod N, ...
  - Has period that evenly divides (p-1)(q-1) [Euler ~1760]

Quantum World Series

- Thus ***if*** we can find the period of
  - x mod N, x^2 mod N, x^3 mod N, x^4 mod N, ...
- ***Then*** we learn something about the prime factors of N!
  - In particular, we learn a divisor of (p-1)(q-1)
  - Not as good as learning p and q themselves ...
  - But if we run it a few times and get several random divisors of (p-1)(q-1)
  - Then we can put them together to get (p-1)(q-1)
  - And then use some math tricks recover p and q

Reduction To Nowhere?

- The sequence
  - x mod N, x^2 mod N, x^3 mod N, x^4 mod N, ...
- Will *eventually* start repeating, but the number of steps before a repeat could be O(N)
  - And N was already huge!
  - Which is why this is not a good *classical* algorithm



Captain Quantum

- So let's make a superposition over all of the numbers in our sequence:
  - $x \bmod N$, $x^2 \bmod N$, $x^3 \bmod N$, $x^4 \bmod N$, ...
- Then perform some quantum operation over all of them to reveal the period
- We're no longer looking for needles in haystacks
- The period is a **global property** of all of the numbers in the sequence taken together

The Key

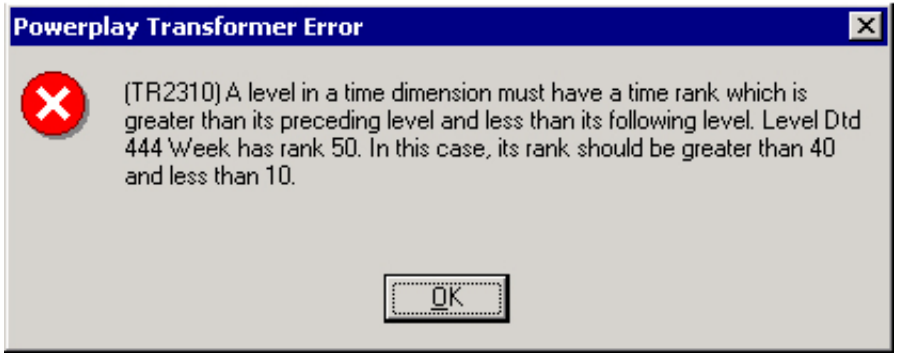
- The key to quantum algorithms is to make a bunch of parallel worlds that all have something (part of the right answer) **in common**.

Shor's Algorithm

- Let's assume we've made our superposition
  - x mod N, x^2 mod N, x^3 mod N, x^4 mod N, ...
- So, given a superposition of elements in a periodic sequence, how do we extract the period?
- We use the **Quantum Fourier Transform**
  - The heart of **Shor's Algorithm** (1994)
- Reasoning by analogy time!

Sleepless In Seattle

- Let's say your body functions on a 26-hour cycle
- Then if you had no appointments and good window blinds, you might wake up at 8am one day, 10am the next day, noon the next day, etc.
- If you were on a 27-hour cycle, it would be 8am, 11am, 2am, etc.
- So, if I see you waking up at 8am, can I tell what kind of cycle you're on?

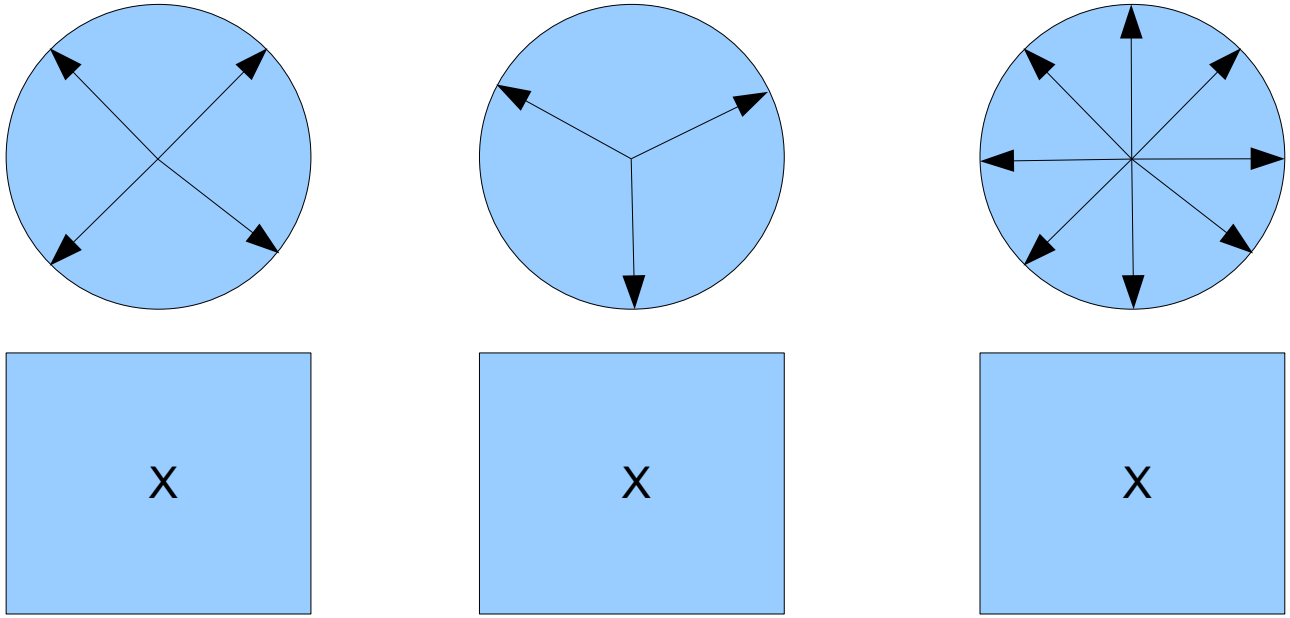


Groundhog Day

- Let's imagine that your bedroom has many clocks in it
  - Each clock has an hour hand (no minute hand)
  - One clock has 26 hours per day
  - One clock has 27 hours per day
  - One clock has 3 hours per day, etc.
  - Each hour is still 60 minutes on all clocks
- Each clock has its own posterboard with a thumbtack in it – mounted right below the clock
  - When you wake up, you move each thumbtack in the direction of its clock's hour hand

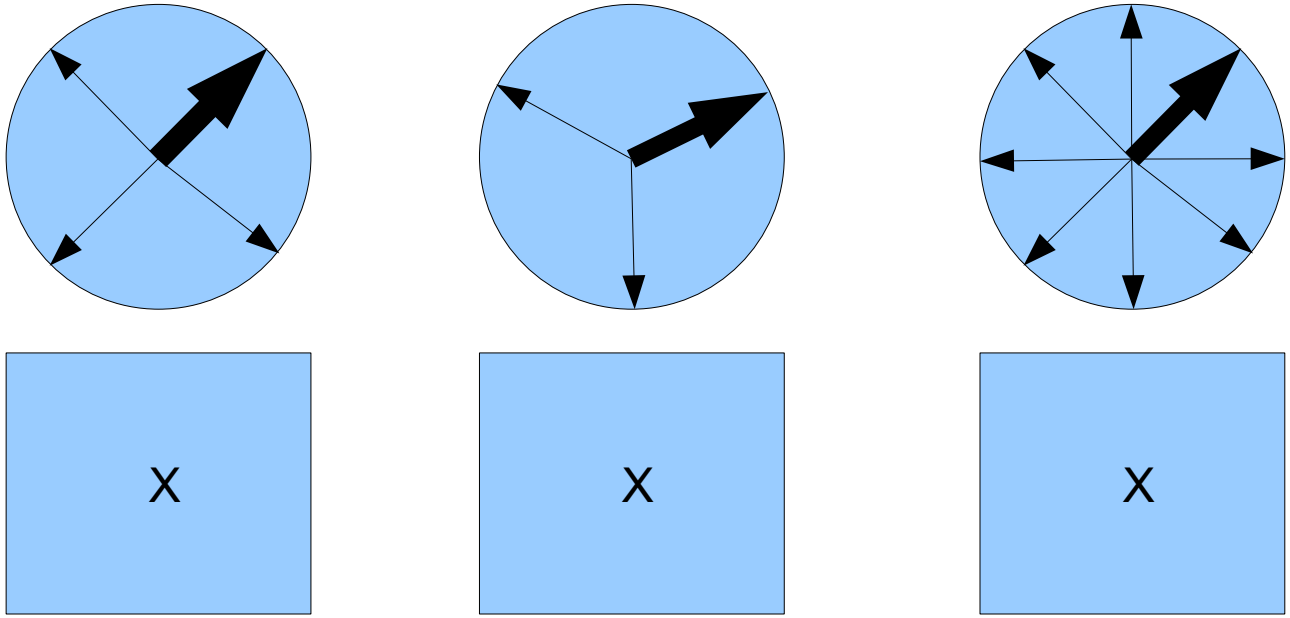
Bedroom Of Doom!

- Three of your clocks: 4-hour, 3-hour, 8-hour:



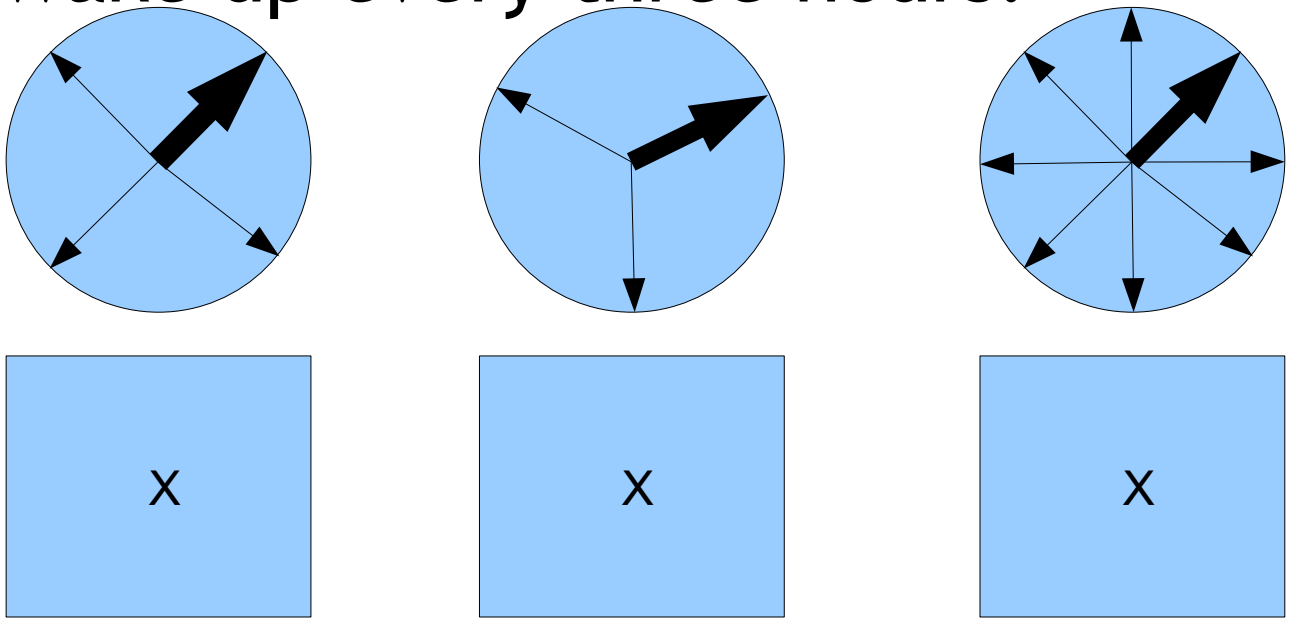
Bedroom Of Doom! (1pm)

- Let's say the current time is 1pm on all clocks.



Bedroom Of Doom! (1pm)

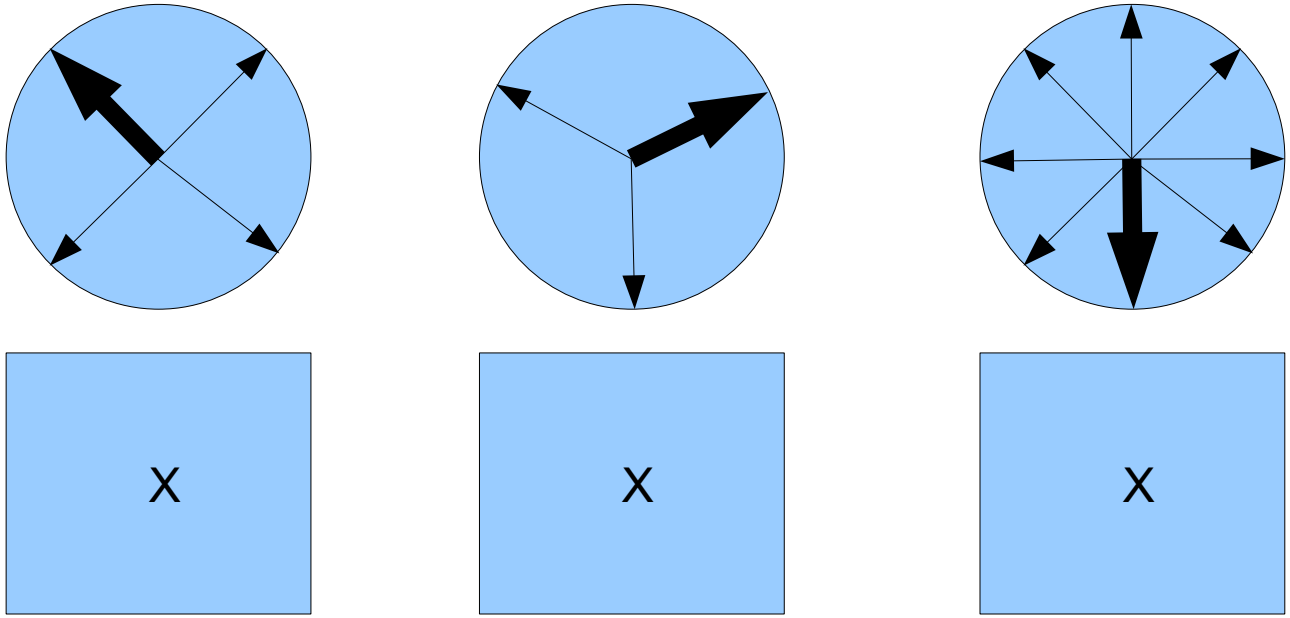
- Let's say you're on a 3-hour day, so you wake up every three hours.



- So when next you wake up, it'll be three hours later ...

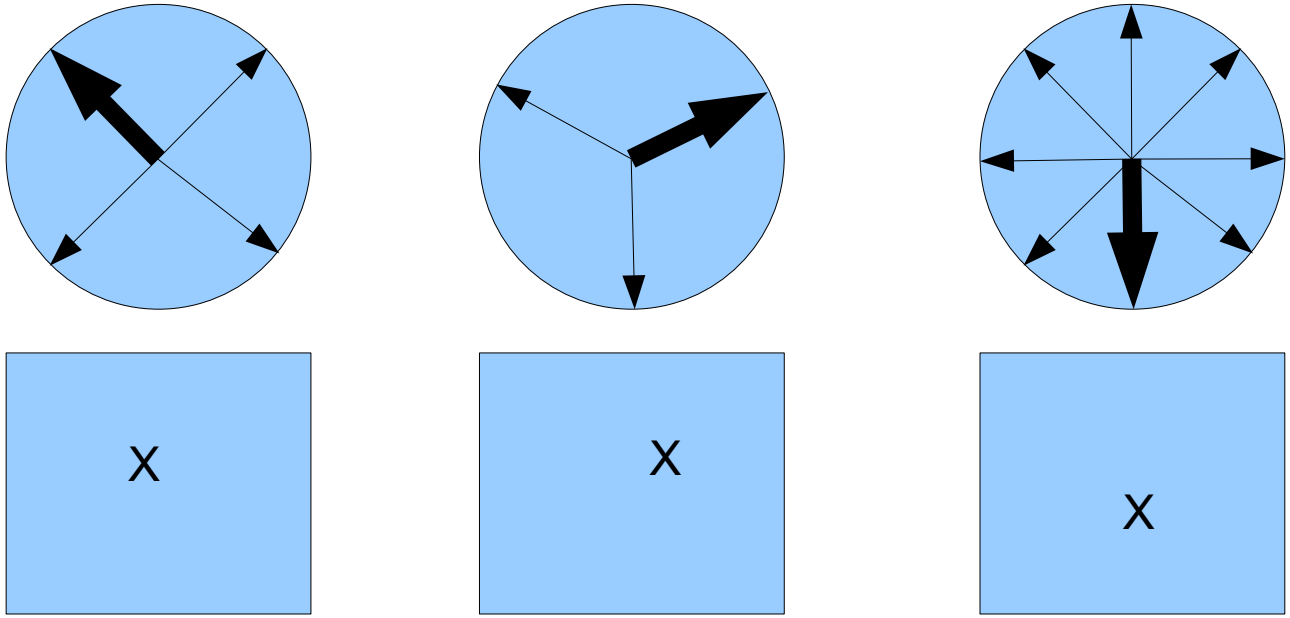
Bedroom Of Doom! (4pm)

- So you adjust the clocks



Bedroom Of Doom! (4pm)

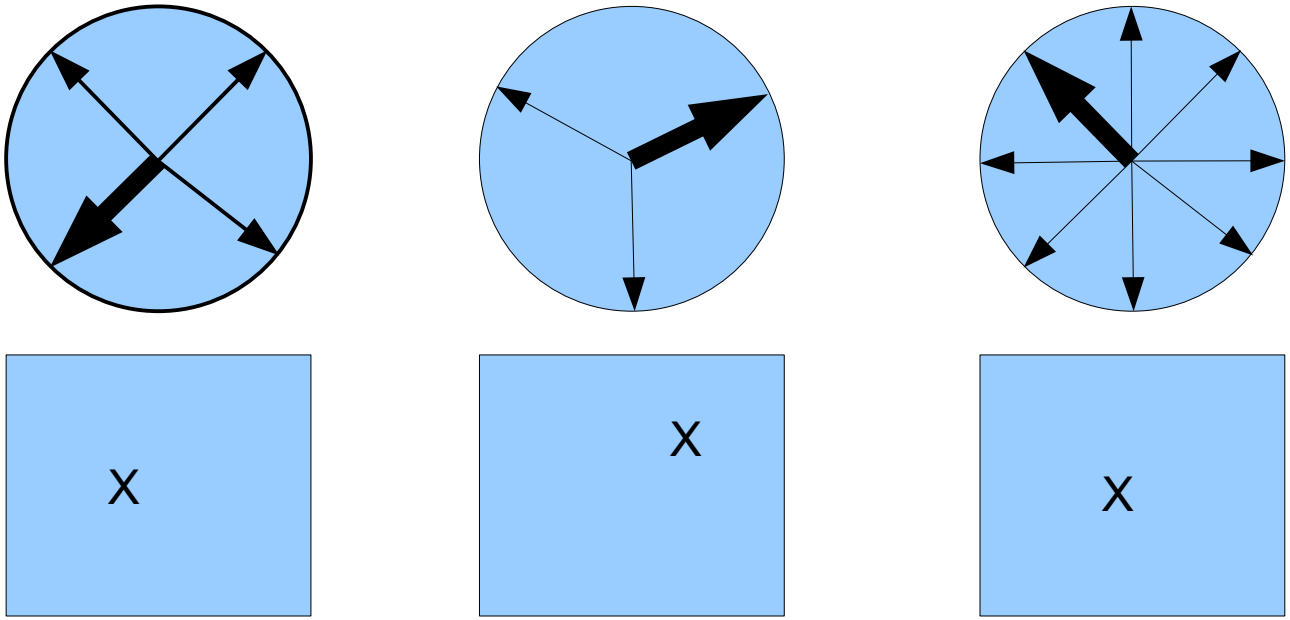
- So you adjust the clocks



- And move the thumbtacks ...

Bedroom Of Doom! (7pm)

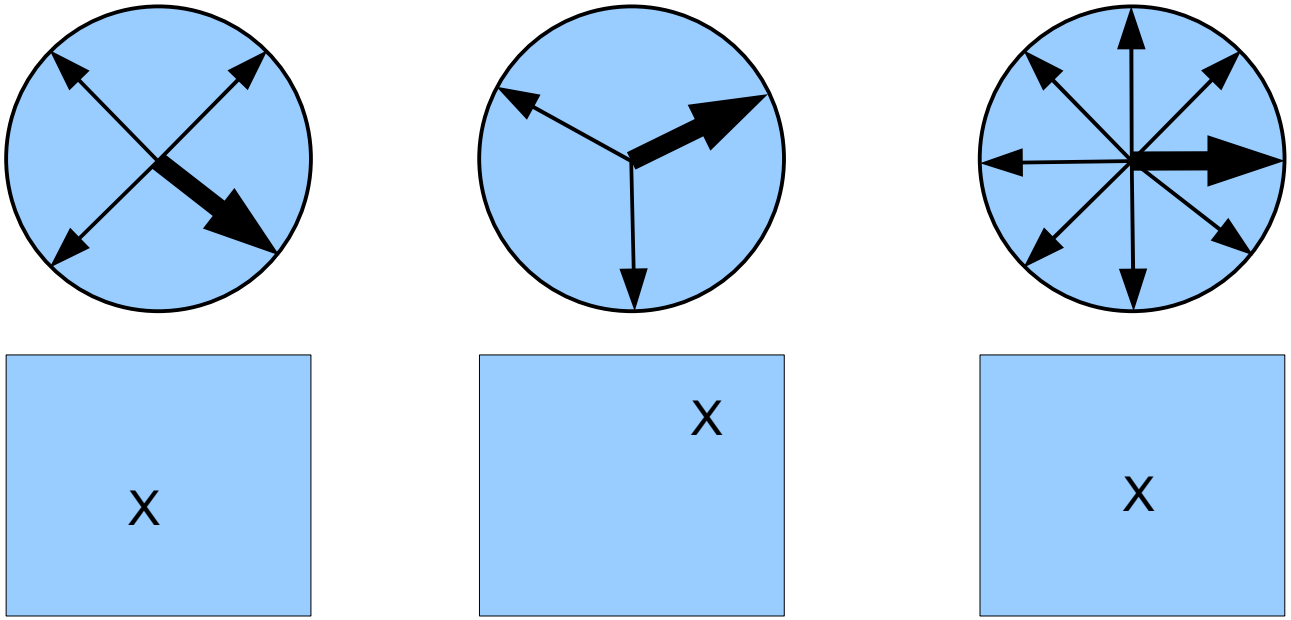
- Wakey Wakey! So you adjust the clocks



- And move the thumbtacks ...

Bedroom Of Doom! (10pm)

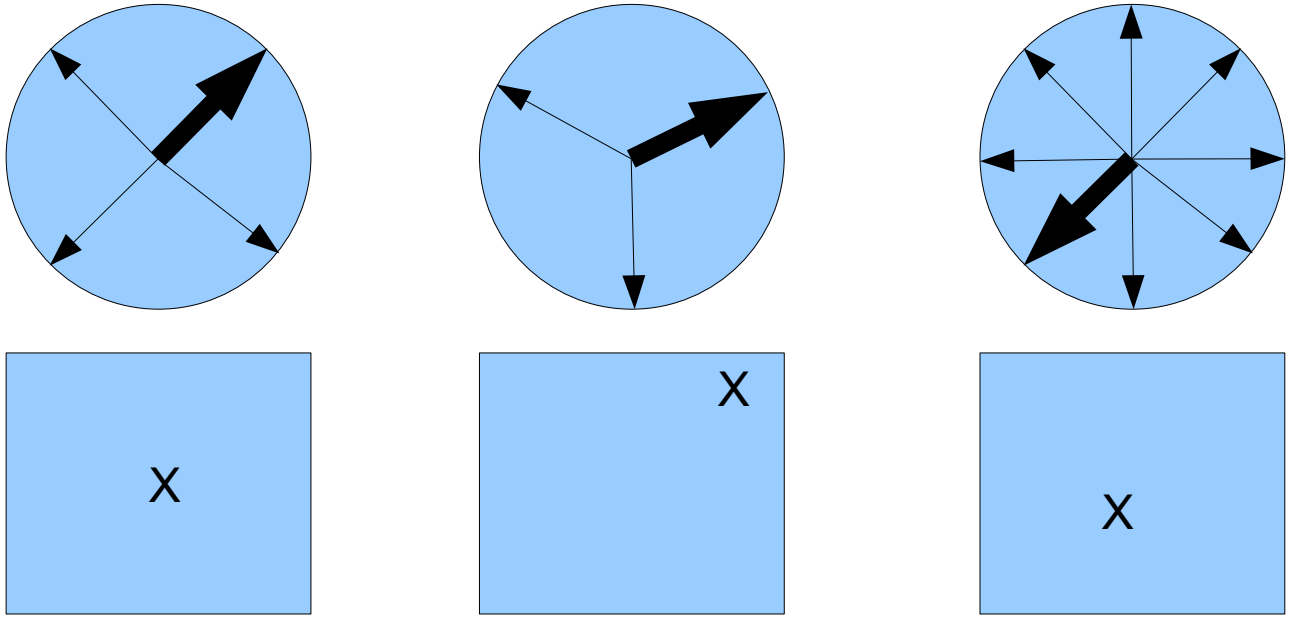
- Wakey Wakey! So you adjust the clocks



- And move the thumbtacks ...

Bedroom Of Doom! (1am)

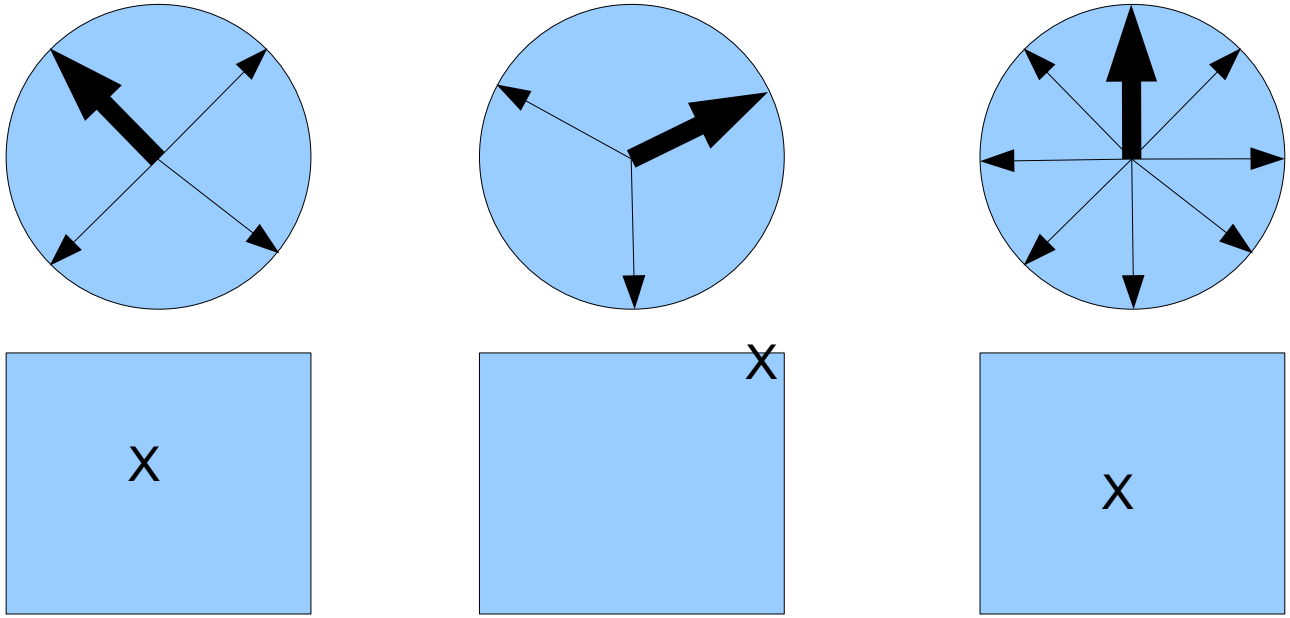
- Sigh! So you adjust the clocks



- And move the thumbtacks ...

Bedroom Of Doom! (4am)

- Sigh! So you adjust the clocks



- *How can you tell which clock matches your period?*

Periodic Motion

It's Just A Jump To The Left

- If you're on a 3-hour day, the 4-hour clock's thumbtack drifts around a little, but every few days it returns to the center
  - All of the movements cancel each other out!
- On the other hand, from the perspective of the 3-hour clock you've been waking up at the same time each “morning”
  - **So you keep moving that thumbtack in the same direction!**
- So just find which thumbtack is farthest from the center and you've found the period.

QFT, QED.

- The **Quantum Fourier Transform** is a linear (unitary) transformation that maps a vector of complex numbers to another vector of complex numbers
- Input vector has nonzero entries every time I wake up, zero entries everywhere else
- Output vector records thumbtack positions
- In the end: it's a linear transform mapping quantum state encoding a periodic sequence to a quantum state encoding the *period* of the sequence!

Interference

- In quantum-land, probabilities are always non-negative but **amplitudes** may be negative, positive or even complex.
- Thus amplitudes corresponding to different ways of getting a particular answer can **intefere destructively** and cancel each other out
- In Shor, all periods from all observations (i.e., all alternate universes) other than the true one cancel each other out. Only for the true period do contributions from all observations (i.e., all universes) point in the same direction.

Shor's Algorithm

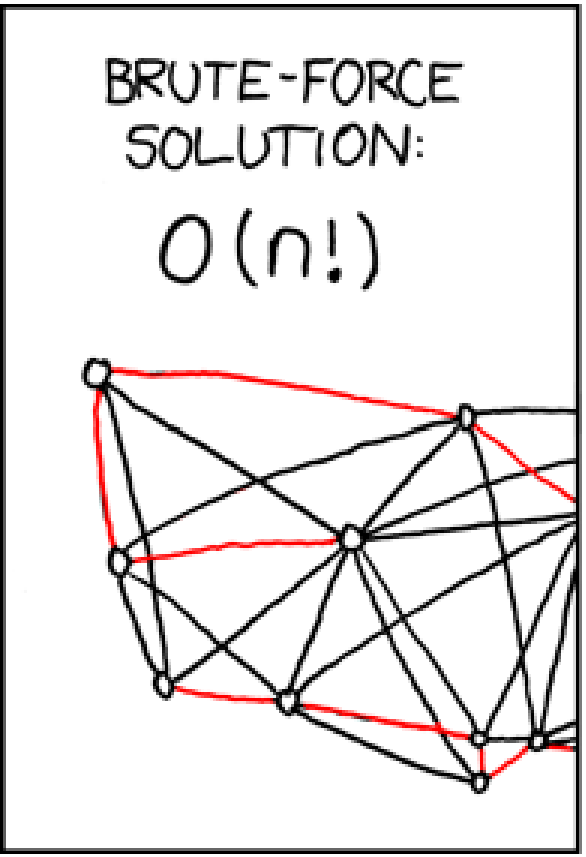
- On a quantum computer, **Shor's Algorithm** takes $O((\log N)^3)$ time to factor the integer N
  - Recall: best classical time ~$O(2^{\log N})$
- In 2001, a team at IBM implemented Shor's algorithm and factored 15 using 7 qubits
  - *Experimental realization of Shor's quantum factoring algorithm using nuclear magnetic resonance*
  - "We use seven spin-1/2 nuclei in a molecule as quantum bits, which can be manipulated with room temperature liquid-state nuclear magnetic resonance techniques."

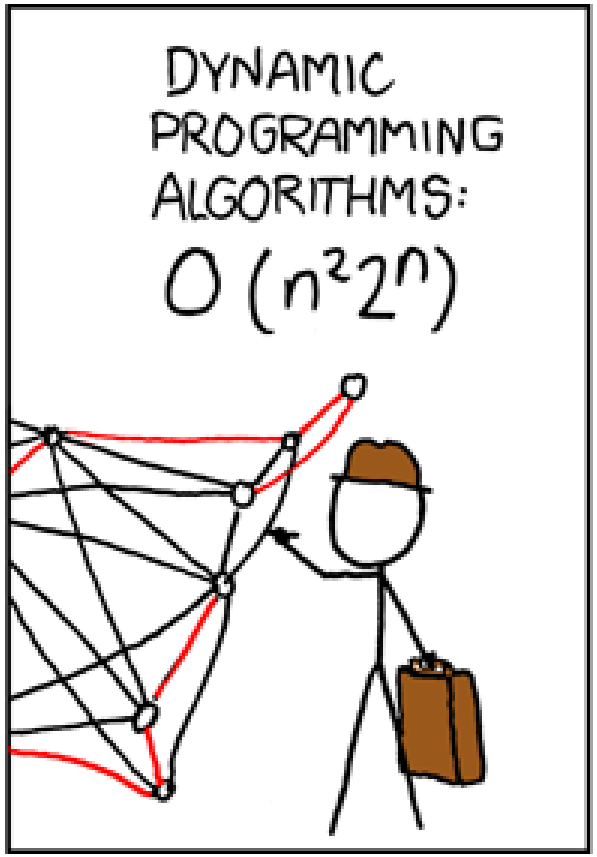
Did We Win?

- A normal Turing machine can simulate a quantum computer (slowly ...)
  - So we do not gain any expressive power
  - Quantum computers do not solve the halting problem
- But quantum computers sure seem faster!
- The class of problems that can be solved efficiently by quantum computers is called **BQP** (bounded error, quantum, polynomial time).

P = NP ?

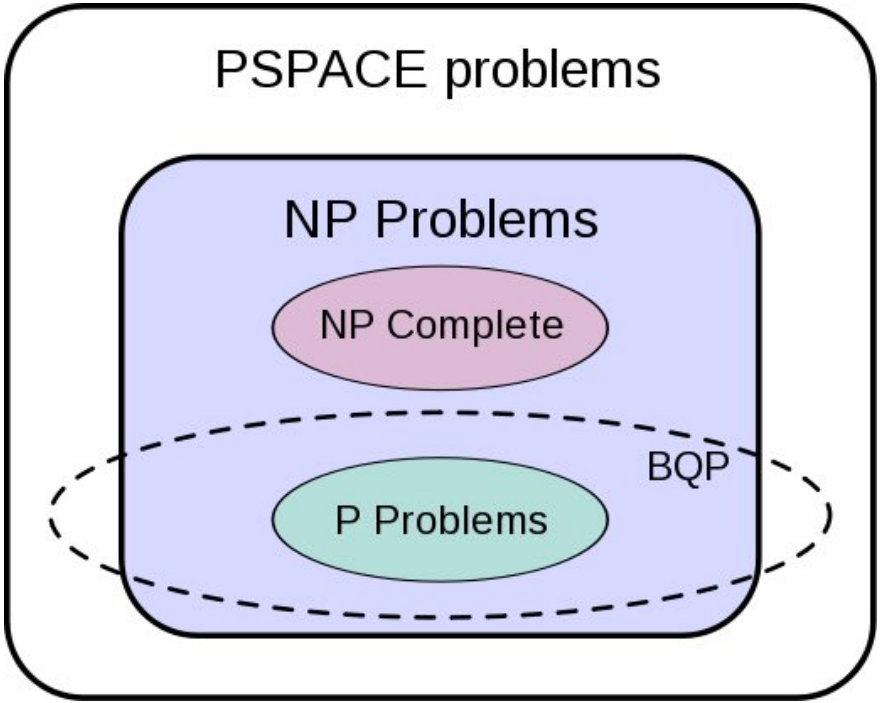
- **So**: “quantum computers can solve NP-complete problems in polynomial time” ?





P = NP ?

- **Misconception**: “quantum computers can solve NP-complete problems in polynomial time”
- BQP is *suspected* to be a superset of P and disjoint from NP (this is *unknown*)



What Is Quantum Good For?

- BQP contains Integer Factorization
  - Believed to be in NP but not in P
- BQP contains Discrete Log
  - Believed to be in NP but not in P
- BQP contains Quantum Database Search
  - Can give an N^2 speedup on any NP-complete problem (by searching through all the answers), but that's still exponential time
- And that's currently about it.

Homework

- Final Exam Soon ...