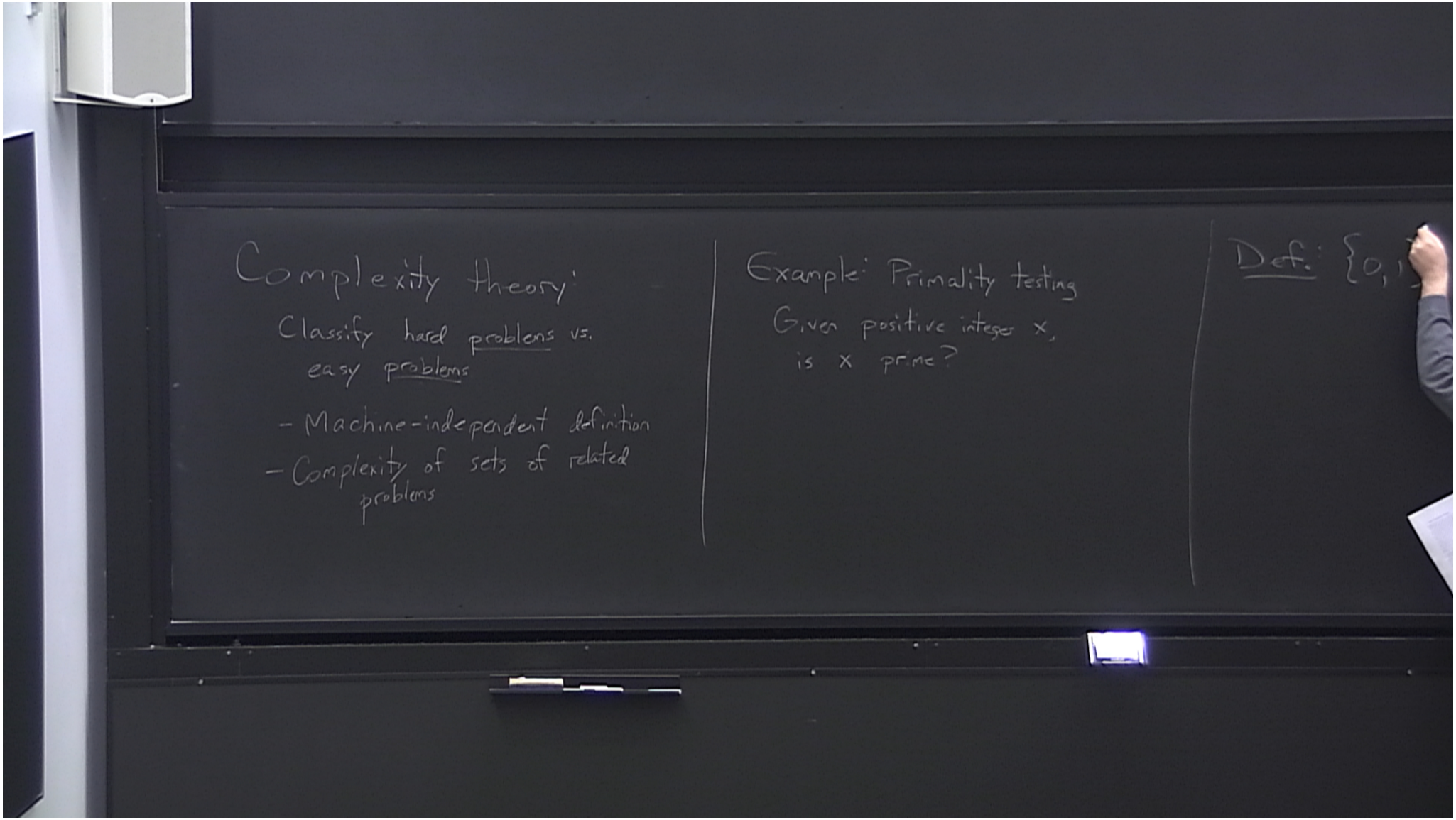
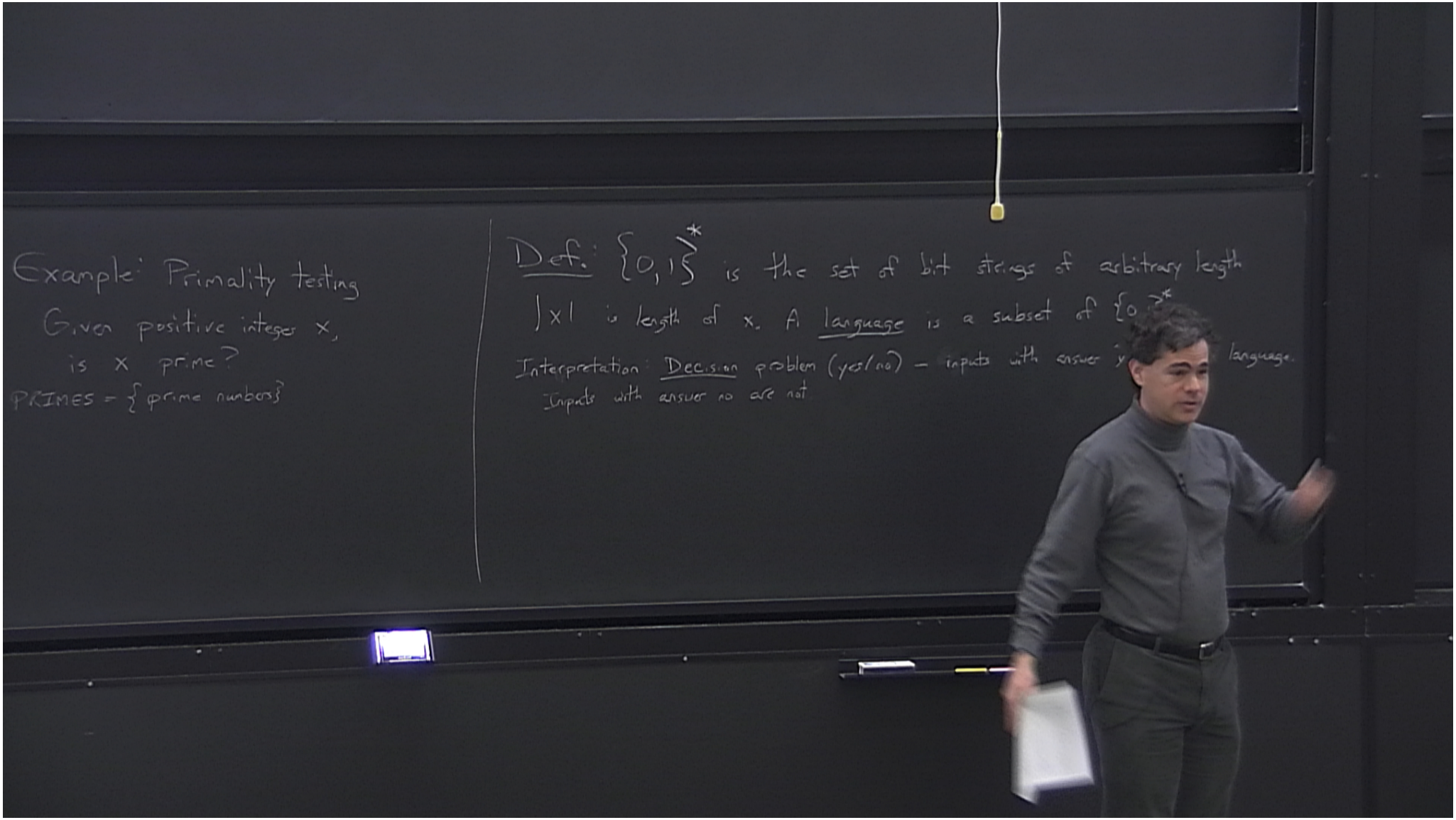
Title: Quantum Information Review-5

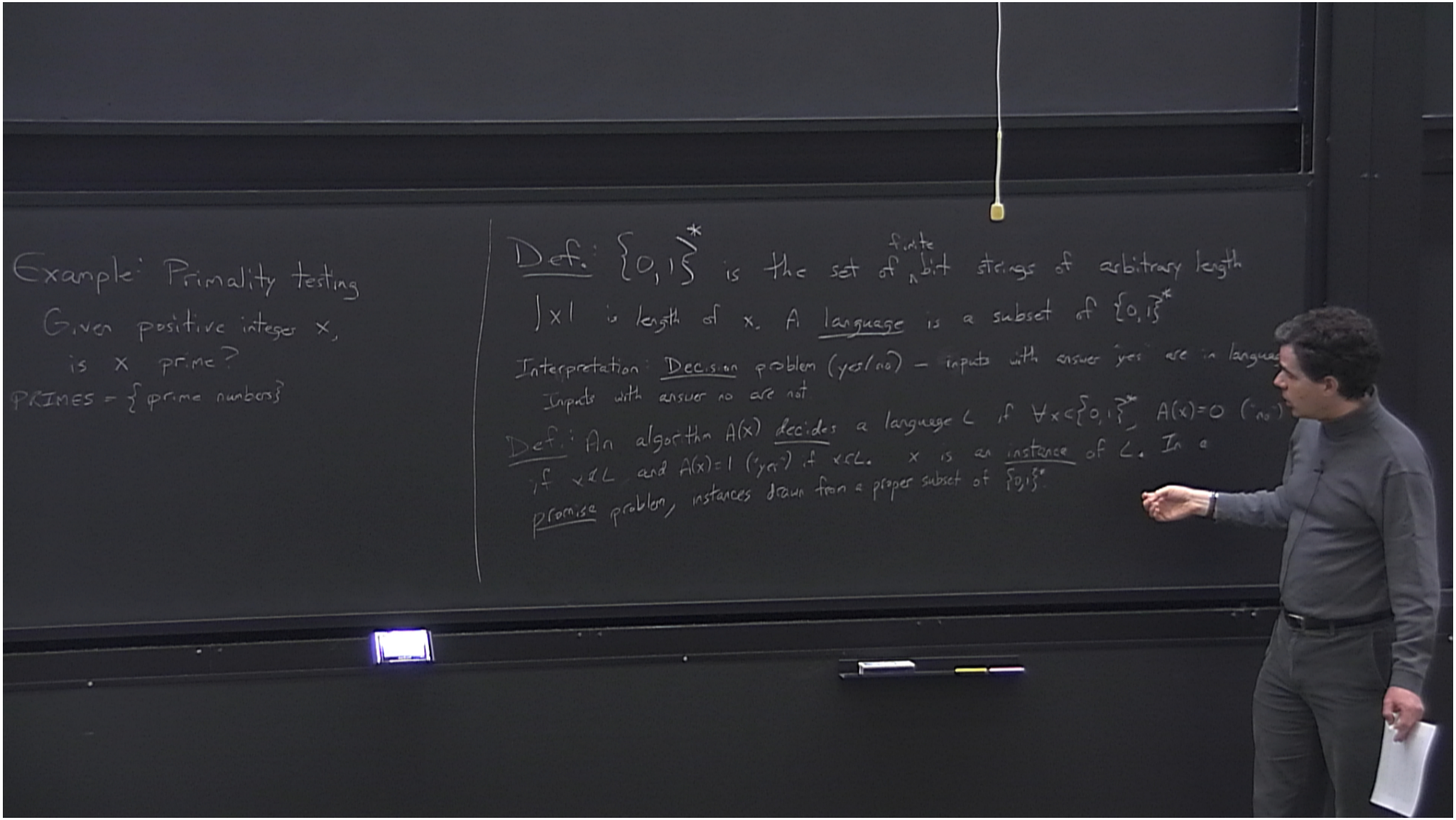
Date: Feb 23, 2015 11:30 AM

URL: http://pirsa.org/15020067

Abstract:


Complexity theory:
Classify hard problems vs. easy problems
- Machine-independent definition
- Complexity of sets of related problems
Example: Primality testing
Given positive integer x, is x prime?
Def: {0,


Example: Primality testing
Given positive integer x,
is x prime?
PRIMES = {prime numbers}
Def: {0,1}* is the set of bit strings of arbitrary length
|x| is length of x. A language is a subset of {0,1}*
Interpretation: Decision problem (yes/no) – inputs with answer yes
Inputs with answer no are not


Example: Primality testing
Given positive integer x,
is x prime?
PRIMES = {prime numbers}
Def: $\{0,1\}^*$ is the set of finite bit strings of arbitrary length
$|x|$ is length of x. A language is a subset of $\{0,1\}^*$
Interpretation: Decision problem (yes/no) – inputs with answer yes are in language
inputs with answer no are not
Def: An algorithm A(x) decides a language L if $\forall x \in \{0,1\}^*$, A(x)=0 ("no")
if $x \notin L$ and A(x)=1 ("yes") if $x \in L$. x is an instance of L. In a
promise problem, instances drawn from a proper subset of $\{0,1\}^*$.

Example: Primality testing

Given positive integer x,

is x prime?

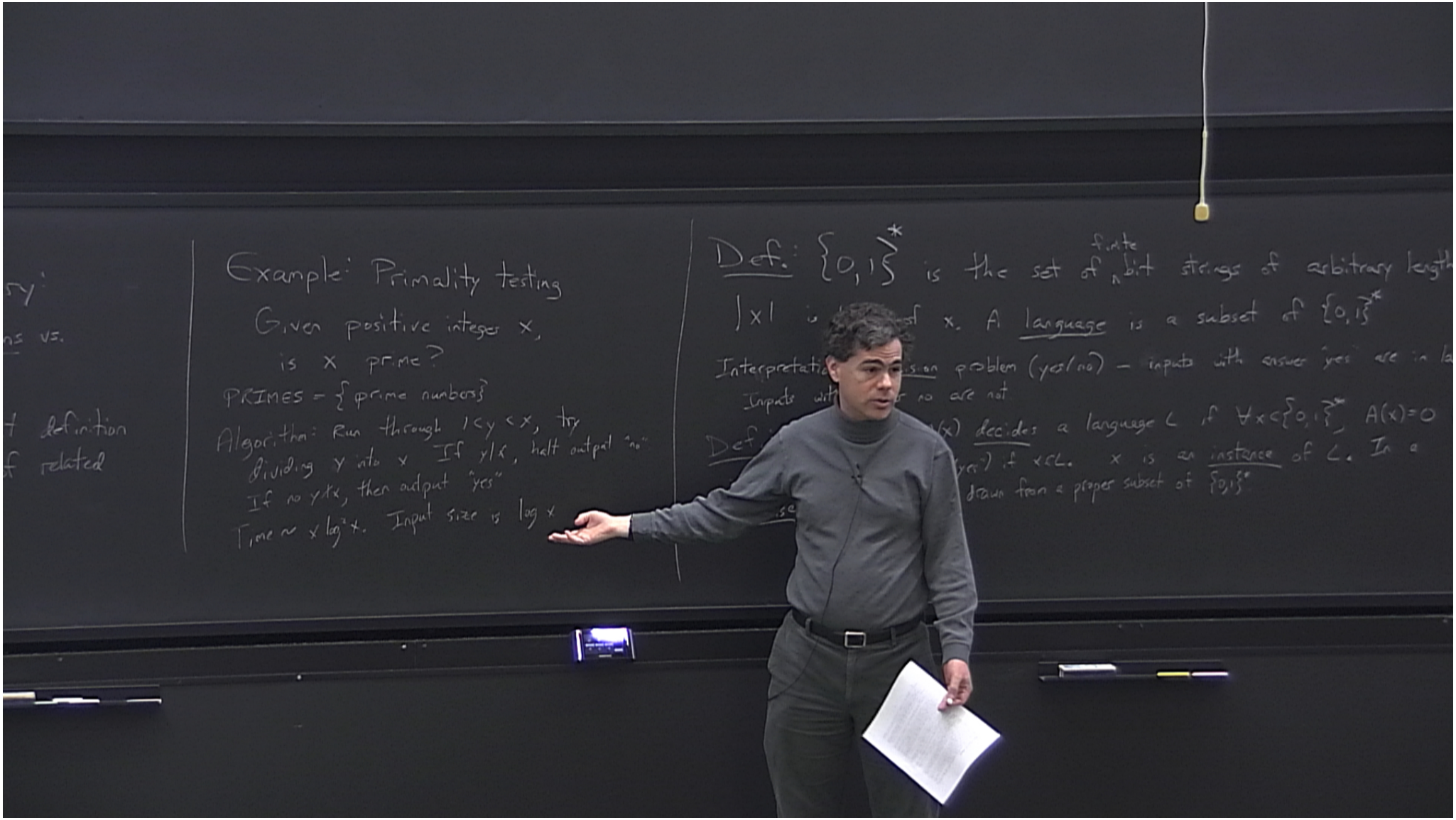
PRIMES = {prime numbers}

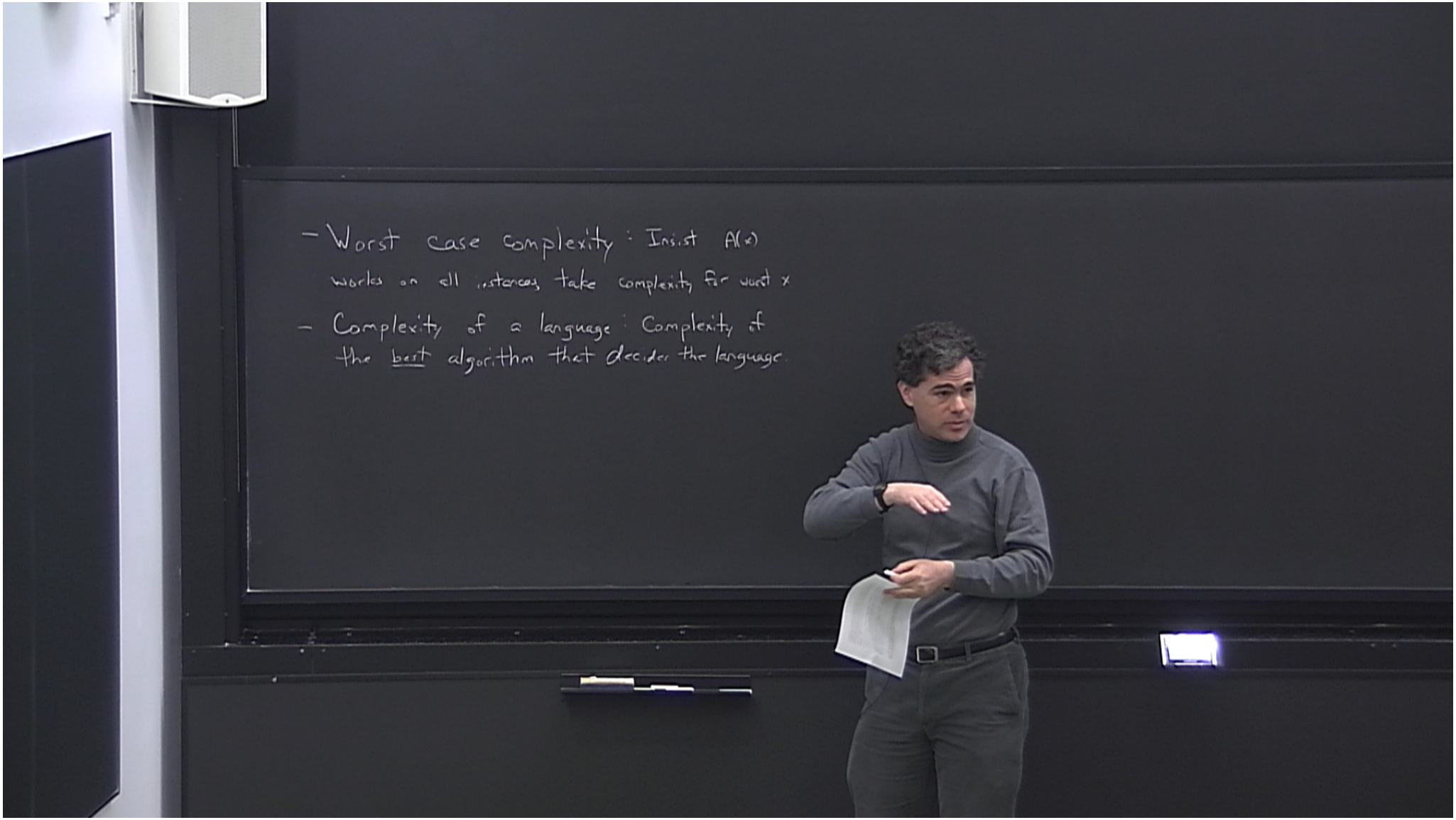
Def: $\{0,1\}^*$ is the set of finite bit strings of arbitrary length

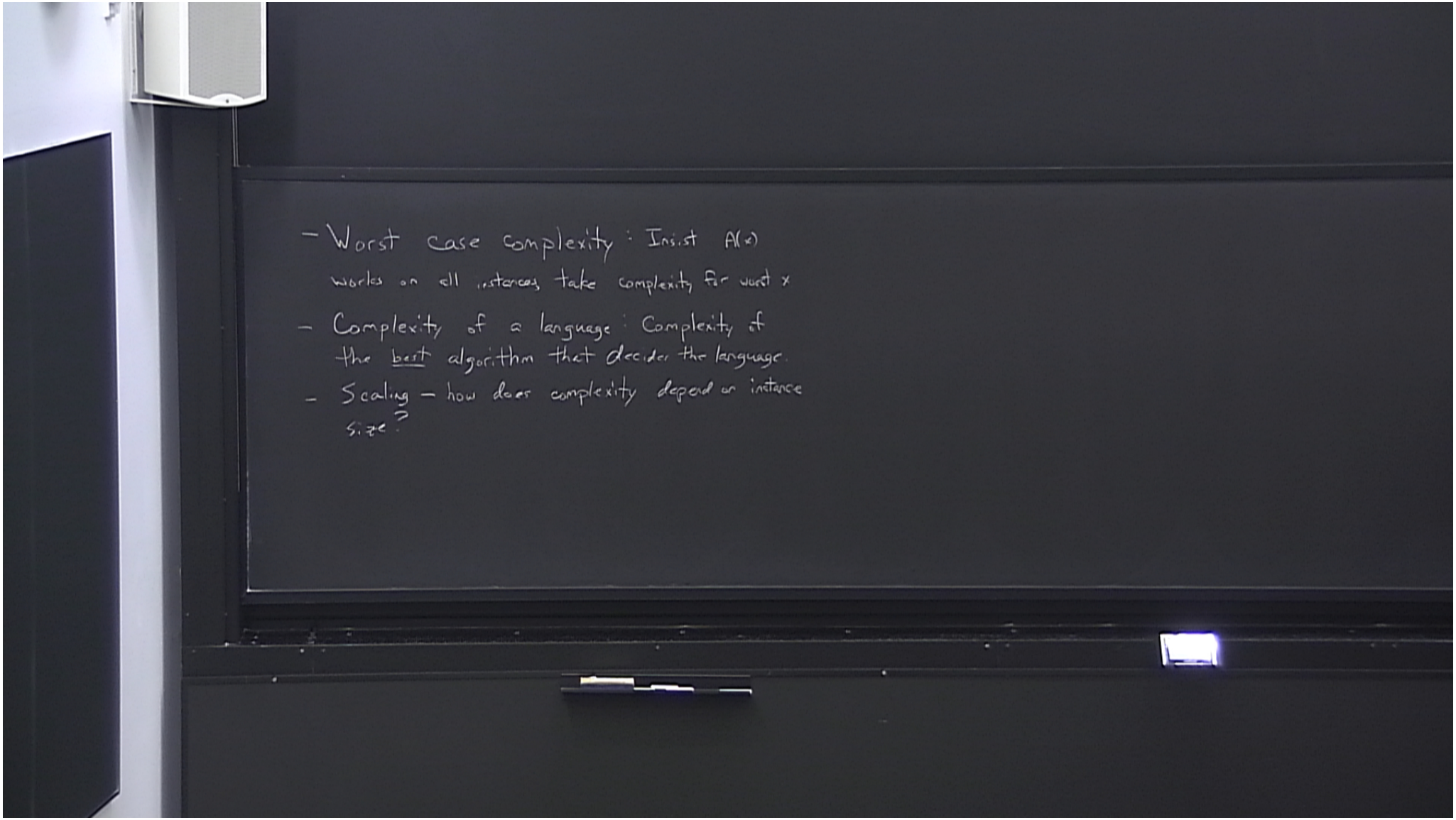
$|x|$ is length of x. A **language** is a subset of $\{0,1\}^*$

Interpretation: **Decision** problem (yes/no) — inputs with answer yes are in language. Inputs with answer no are not

Def: An algorithm A(x) **decides** a language L if $\forall x \in \{0,1\}^*$, A(x)=0 ("no") if $x \notin L$ and A(x)=1 ("yes") if $x \in L$. x is an **instance** of L. In a **promise** problem, instances drawn from a proper subset of $\{0,1\}^*$.


Example: Primality testing
Given positive integer x,
is x prime?
PRIMES = {prime numbers}
Algorithm: Run through 1<y<x, try
dividing y into x. If y|x, halt output "no"
If no y|x, then output "yes"
Time ~ x log² x. Input size is log x
Def: {0,1}* is the set of finite bit strings of arbitrary length
|x| is ... of x. A language is a subset of {0,1}*
Interpretation: ... on problem (yes/no) — inputs with answer yes are in ...
inputs with ... no are not
... decides a language L if ∀x∈{0,1}*, A(x)=0
... if x∈L. x is an instance of L. In a
... drawn from a proper subset of {0,1}*


- Worst case complexity: Insist A(x)
works on all instances, take complexity for worst x
- Complexity of a language: Complexity of
the best algorithm that decides the language.


- Worst case complexity: Insist A(x)
works on all instances, take complexity for worst x
- Complexity of a language: Complexity of
the best algorithm that decides the language.
- Scaling – how does complexity depend on instance
size?

size?

Ignore constant factors

Different models of computation may differ by polynomials $O(g(|x|)) \rightarrow O(poly(g(|x|)))$

$f(|x|)$ for some polynomial f

Languages in P are "easy". Polynomial-time algorithm is "efficient"

Strong Church-Turing thesis: The complexity of languages differs by only polynomial factors in physically reasonable models of computation.

QC seem to violate this!

Quantum Church-Turing thesis: Any physically reasonable model of computation can be simulated in polynomial time on a quantum computer

$f(|x|)$ for some polynomial f

Languages in P are "easy". Polynomial-time algorithm is "efficient"

the same set of languages

plexity
factors
tion.

Quantum Church-Turing thesis: Any physically reasonable model of computation can be simulated in polynomial time on a quantum computer

Def: BQP is the set of languages L for which $\exists$ quantum algorithm $A_L(x)$ s.t.

1) $A_L(x)$ runs in time $\text{poly}(|x|)$,
2) If $x \in L$, then $A_L(x) = \text{yes}$ w/ prob $\geq 2/3$,
3) If $x \notin L$, then $A_L(x) = \text{no}$ w/ prob $\geq 2/3$