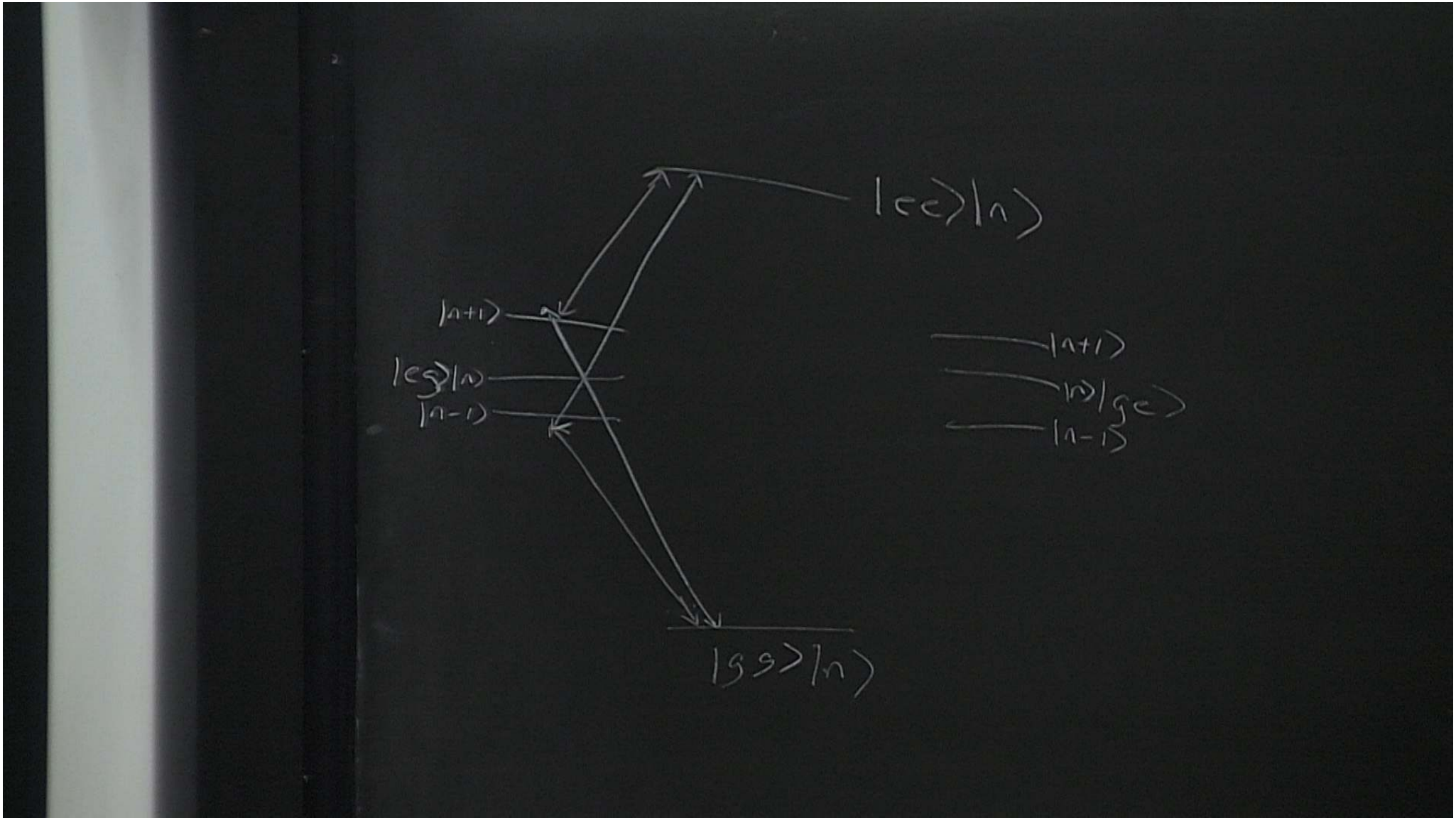
Title: PSI 2018/2019 - Quantum Information Review - Lecture 6

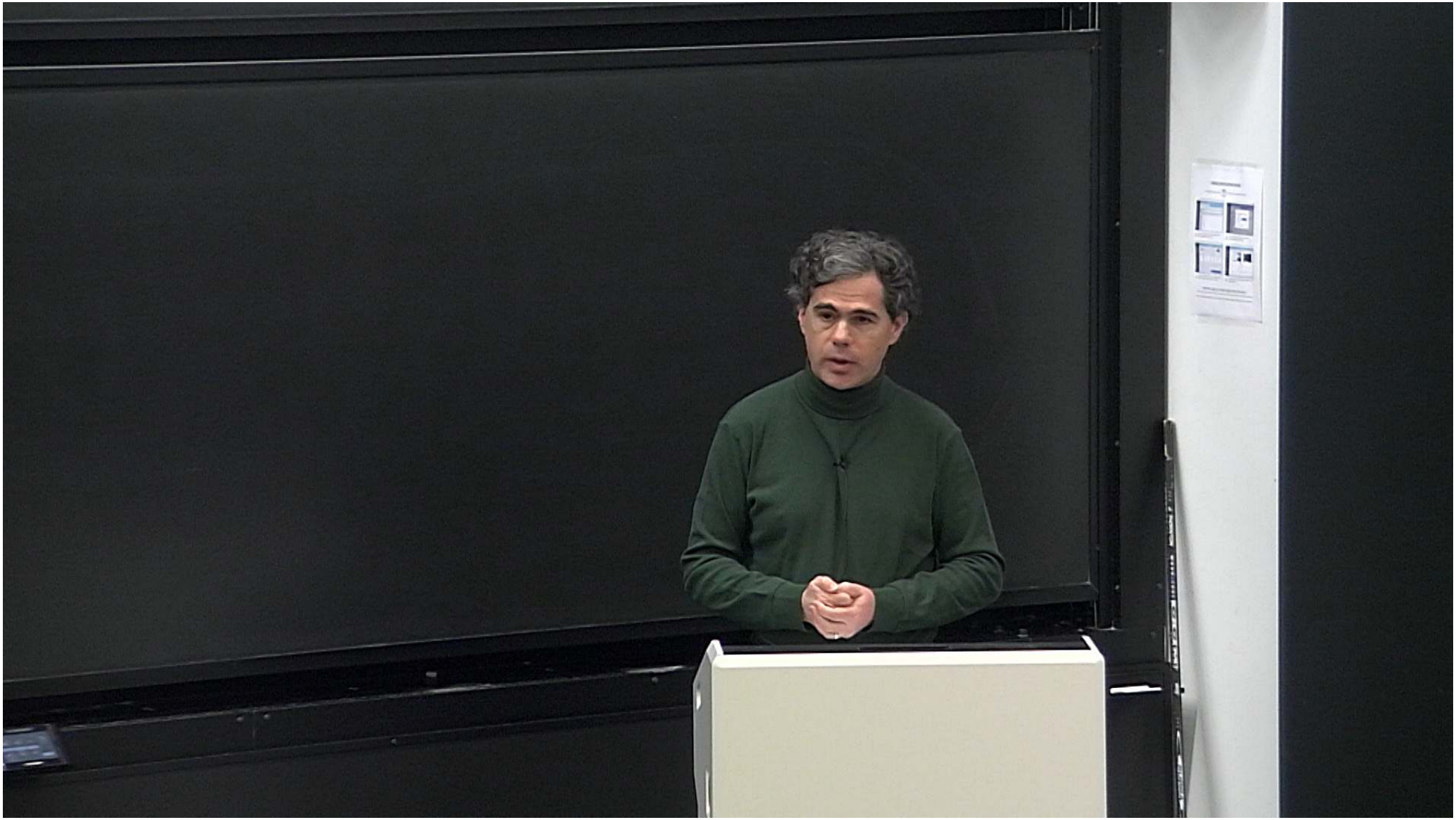
Speakers: Daniel Gottesman

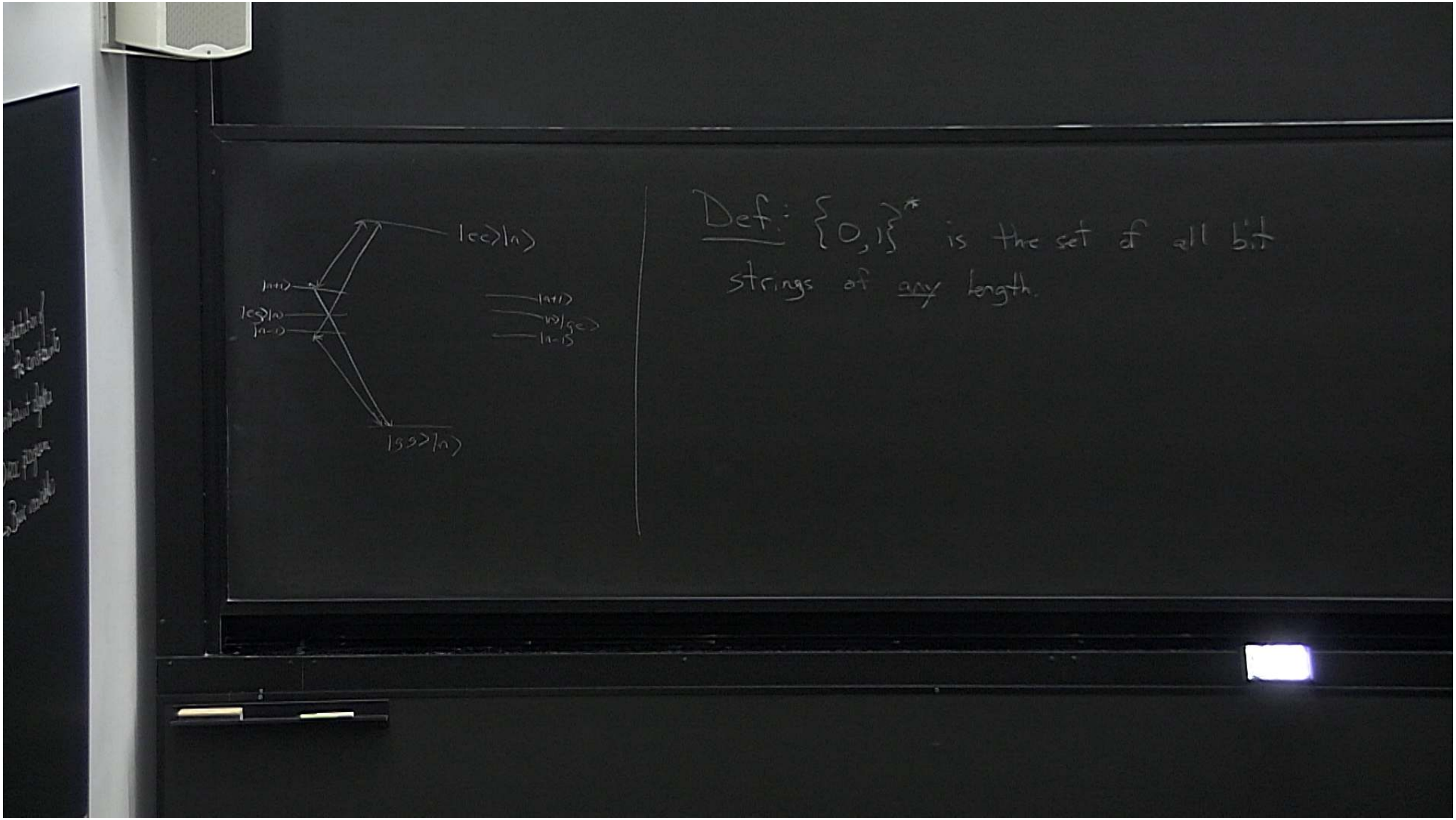
Collection: PSI 2018/2019 - Quantum Information Review (Gottesman)

Date: March 11, 2019 - 11:30 AM

URL: http://pirsa.org/19030043


|ee⟩|n⟩
|n+1⟩
|g⟩|n⟩
|n-1⟩
|n+1⟩
|n⟩|ge⟩
|n-1⟩
|gg⟩|n⟩





Def: $\{0,1\}^*$ is the set of all bit strings of ~~any~~ length.

Def: $\{0,1\}^*$ is the set of all bit strings of *any* length. The size $|x|$ of $x \in \{0,1\}^*$ is the length of the string. A *language* is a subset of $\{0,1\}^*$.

Interpretation: Bit strings as input to problem
Ones in the language are inputs with answer "yes"
Others give answer "no".

Example: PRIMES. Given x,
is x prime?

Def: An algorithm $A(x)$ *decides* language L if $\forall x \in \{0,1\}^*$, $A(x)=0$ if $x \notin L$, and $A(x)=1$ if $x \in L$.

x is an *instance* of L; "yes" instance if $x \in L$, "no" instance if $x \notin L$

In a *promise* problem, possible inputs are subset of $\{0,1\}^*$

Example: PRIMES. Given x, is x prime?

For $1 < y < x$, try dividing x by y. If integer, then output "no"; otherwise continue. At end, output "yes".

Def: An algorithm $A(x)$ *decides* language L if $\forall x \in \{0,1\}^*$, $A(x) = 0$ if $x \notin L$, and $A(x) = 1$ if $x \in L$.

x is an *instance* of L; "yes" instance if $x \in L$, "no" instance if $x \notin L$.

In a *promise* problem, possible inputs are subset of $\{0,1\}^*$

Complexity using "Big-O" notation:

Def: A function $f(n)$ is $O(g(n))$ if $\exists$ constants n_0, C s.t.

$$f(n) \leq C g(n) \quad \forall n > n_0$$

Def: A complexity class is a set of languages. P is the set of languages that can be decided in time $O(|x|^n)$ for some n. If $L \in P$, we say that there is an efficient algorithm for L; or that L is "easy"

Some problems are uncomputable. undecidable.

E.g.: Halting problem.

Church-Turing thesis: All physically reasonable models of computation have the same set of computable functions.

Church-Turing thesis: All physically reasonable models of computation have the same set of computable functions.

Strong Church-Turing thesis: All physically reasonable models of computation differ in complexity by at most polynomial factors

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

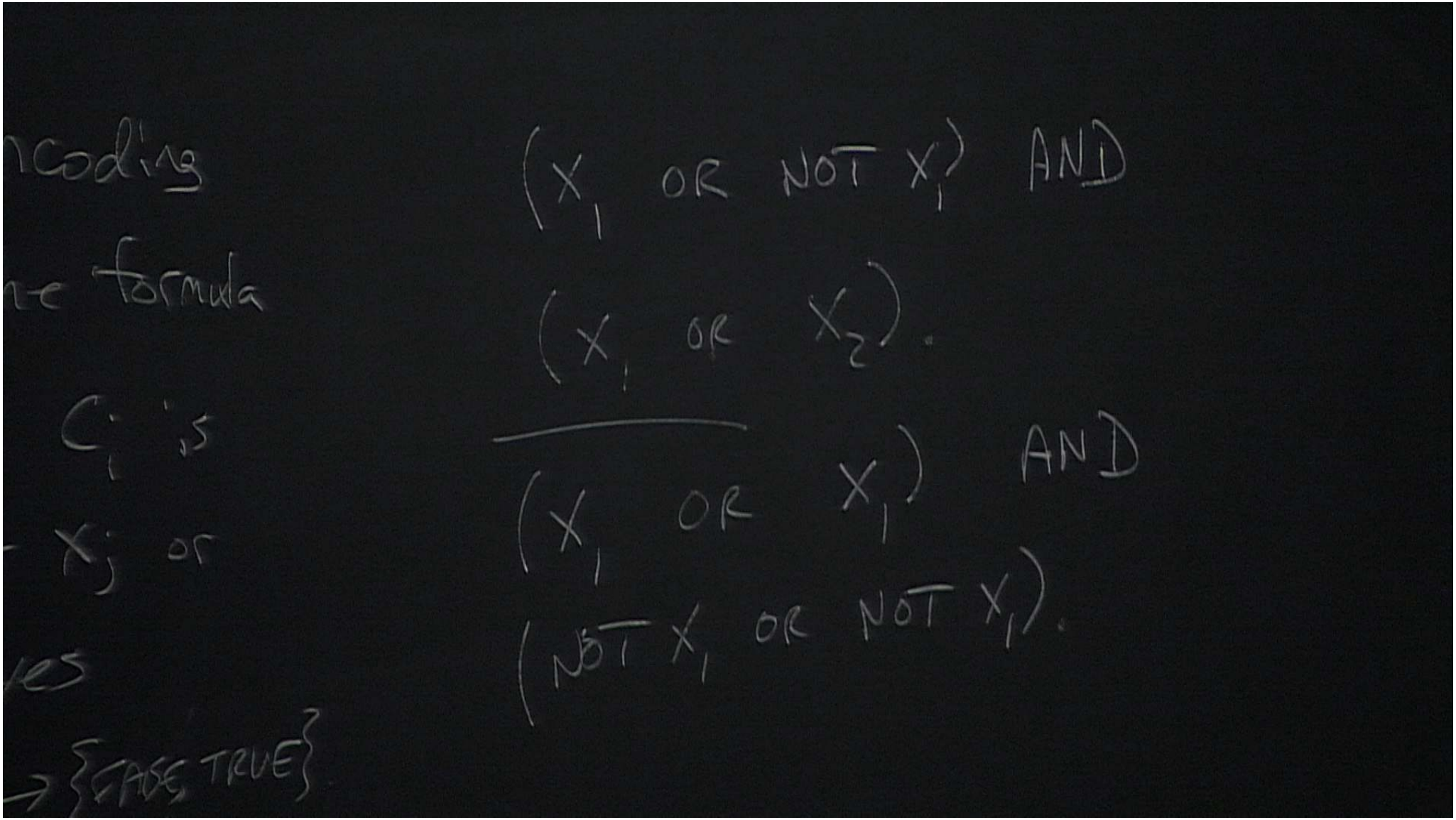
Def: NP is the class of languages L s.t. $\exists$ a "verifier" algorithm $V(x, w)$ s.t.

1. $V(x, w)$ runs in time polynomial in $|x|$ $\forall x, w$ s.t. $|w| = \text{poly}(|x|)$
2. $\forall x \in L$, $\exists w_x$ s.t. $V(x, w_x) = \text{yes}$, $|w_x| = \text{poly}(|x|)$
3. $\forall x \notin L$, $\forall w$ s.t. $|w| = \text{poly}(|x|)$, $V(x, w) = \text{no}$.

w_x is <u>witness</u> for $x \in L$.

Example: k-SAT. Instance is a binary encoding of a Boolean formula on n variables $\{x_i\}$. The formula is the AND of $m = \text{poly}(n)$ clauses C_i. Each C_i is the OR of k "atoms," each of which is either x_j or NOT x_j for some j. The formula $f(\vec{x})$ is a yes instance of k-SAT if $\exists$ satisfying assignment $x_i \to \{0,1\}$ s.t. $f(\vec{x}) = \text{TRUE}$.

Example: k-SAT. Instance is a binary encoding of a Boolean formula on n variables $\{x_i\}$. The formula is the AND of $m = \text{poly}(n)$ clauses C_i. Each C_i is the OR of k "atoms" each of which is either x_j or NOT x_j for some j. The formula $f(\vec{x})$ is a yes instance of k-SAT if $\exists$ satisfying assignment $x_i \to \{\text{FALSE, TRUE}\}$ s.t. $f(\vec{x}) = \text{TRUE}$.


ncoding
ne formula
C_i is
x_j or
les
→ {FALSE, TRUE}
$(x_1$ OR NOT $x_1)$ AND
$(x_1$ OR $x_2)$.
$(x_1$ OR $x_1)$ AND
(NOT x_1 OR NOT x_1).