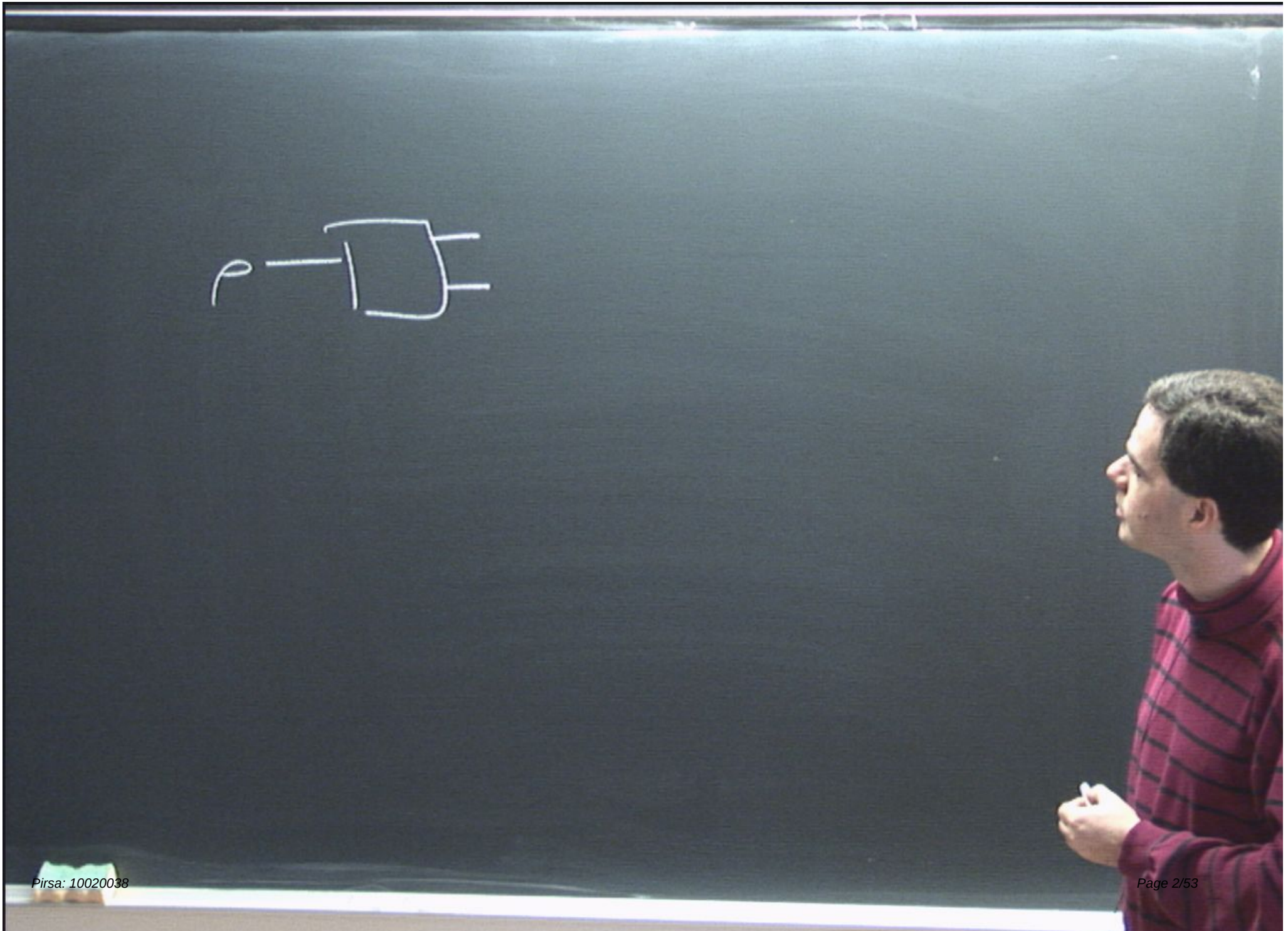
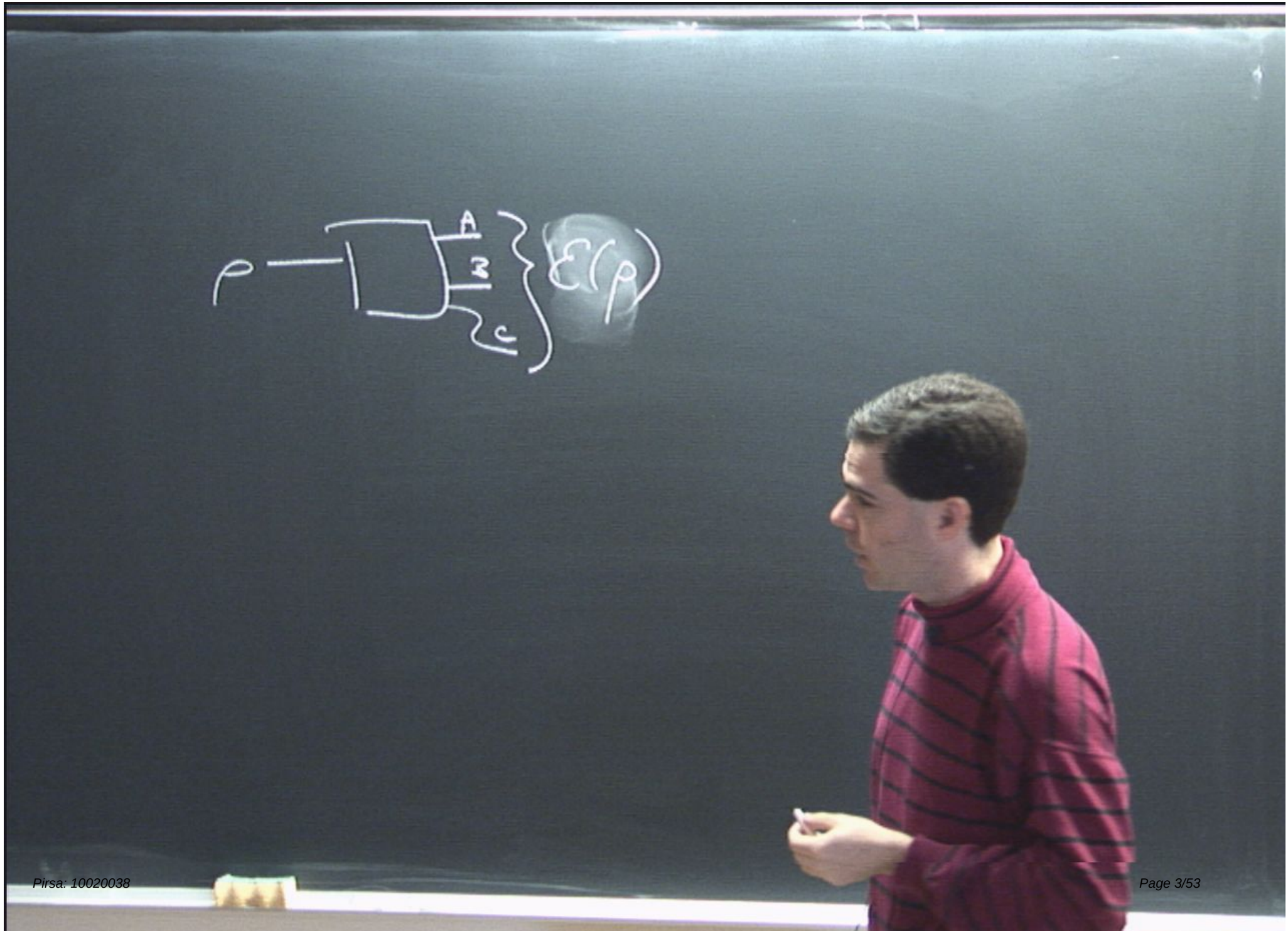
Title: Quantum Information - Review (PHYS 635) - Lecture 6

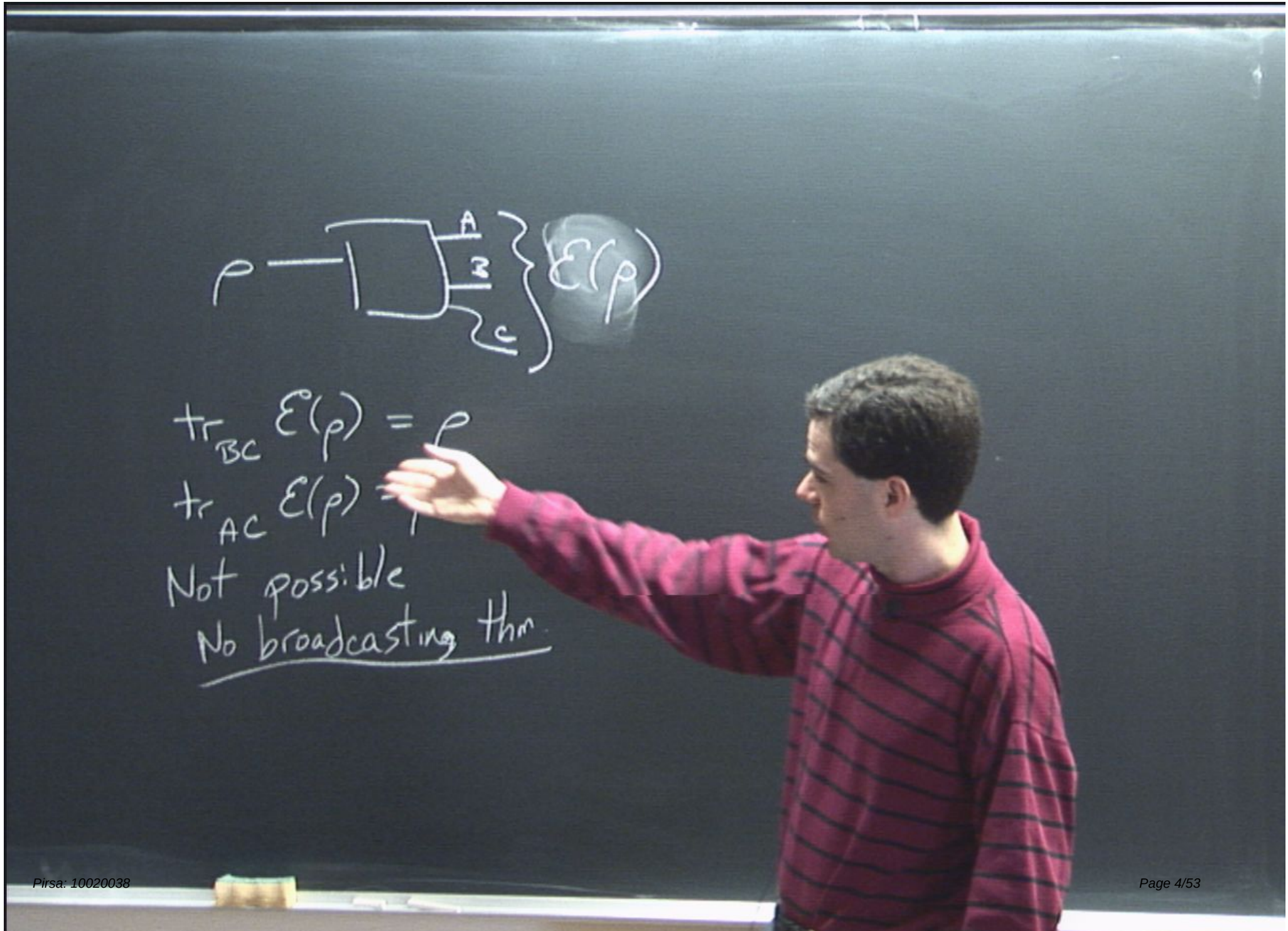
Date: Feb 01, 2010 09:00 AM

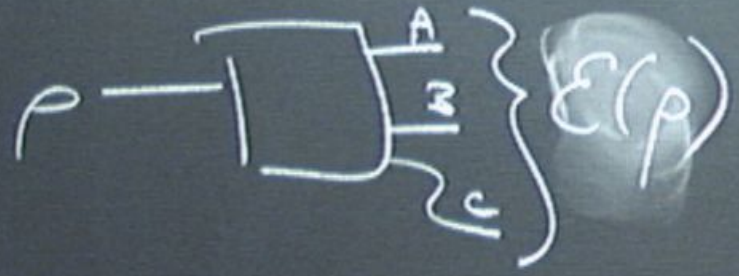
URL: http://pirsa.org/10020038

Abstract: Week 1: Basic topics (Qubits, quantum gates, quantum circuits, density matrices, quantum operations, entropy, entanglement) Week 2: Algorithms and complexity (Languages, complexity classes, oracles, RSA, Deutsch-Jozsa algorithm, Shor's algorithm, Grover's algorithm) Week 3: Information theory and implementations (Overview of implementations, quantum error correction, quantum cryptography, quantum information theory)




A
B
C
ρ
ε(ρ)


ρ
A
B
C
$\mathcal{E}(\rho)$
$\mathrm{tr}_{BC}\,\mathcal{E}(\rho) = \rho$
$\mathrm{tr}_{AC}\,\mathcal{E}(\rho) = \rho$
Not possible
No broadcasting thm

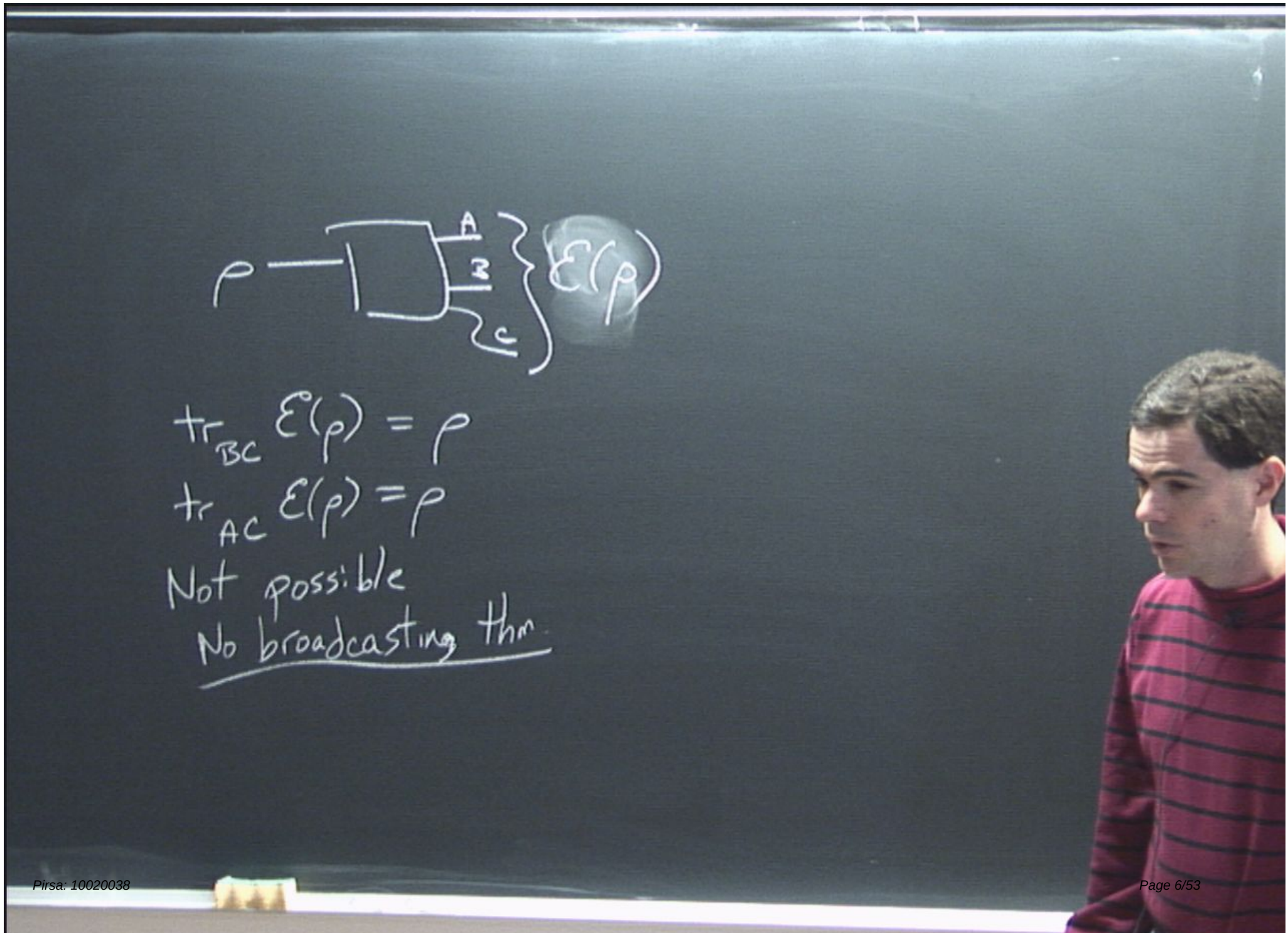


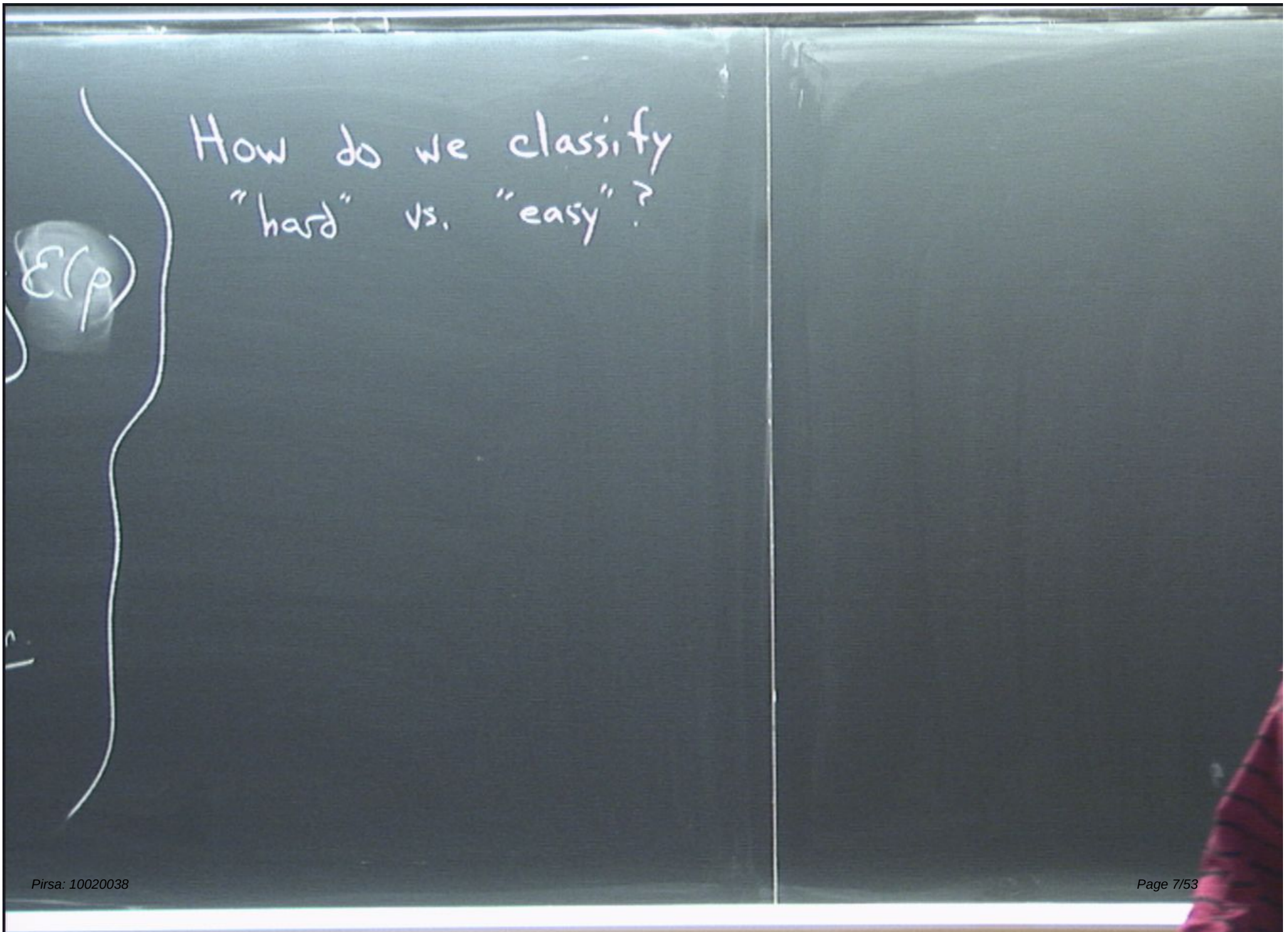
$$tr_{BC}\,\mathcal{E}(\rho) = \rho$$

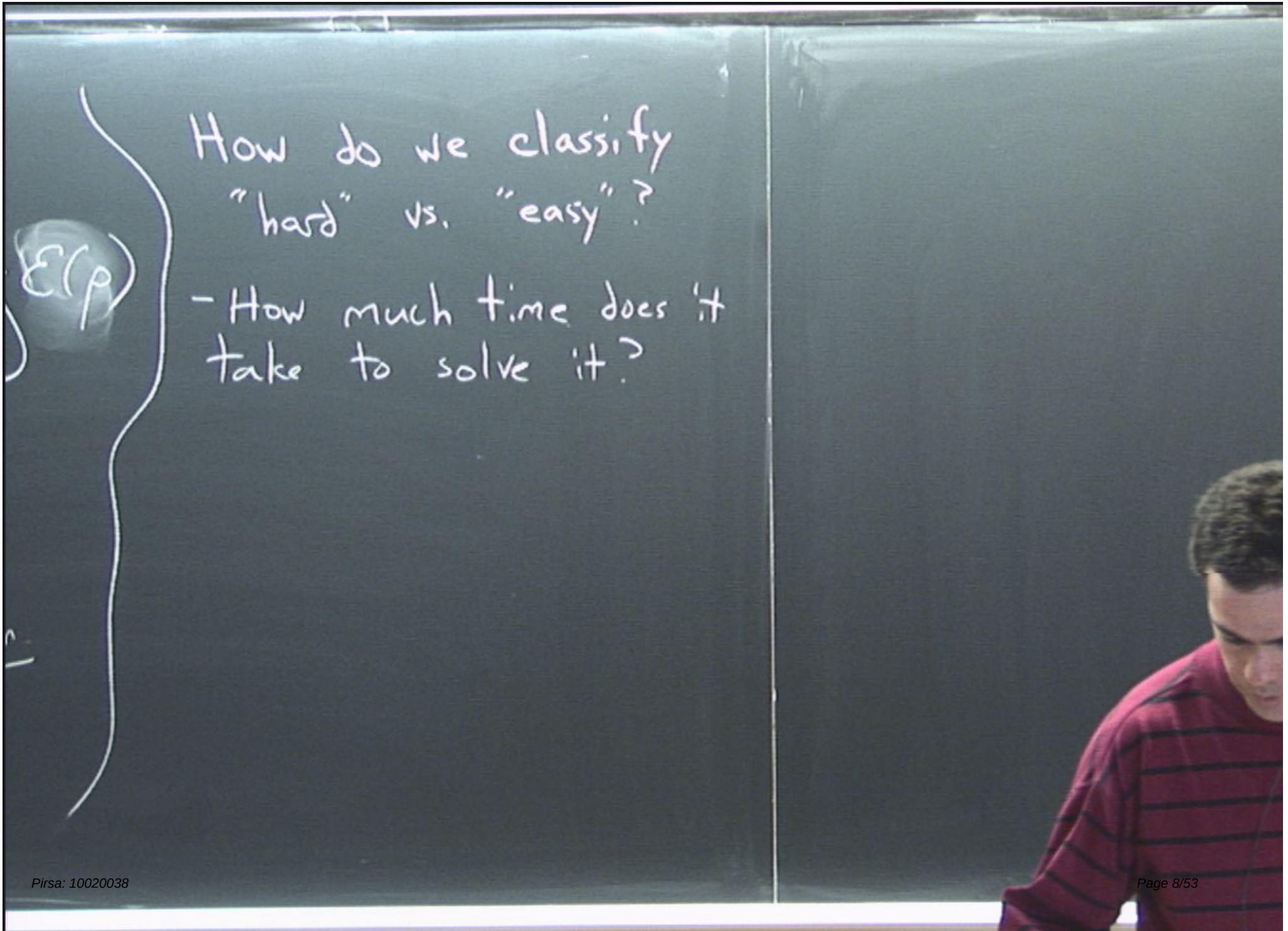
$$tr_{AC}\,\mathcal{E}(\rho) = \rho$$

Not possible

No broadcasting thm


ρ
A
B
C
$\mathcal{E}(\rho)$
$\mathrm{tr}_{BC}\,\mathcal{E}(\rho) = \rho$
$\mathrm{tr}_{AC}\,\mathcal{E}(\rho) = \rho$
Not possible
No broadcasting thm


How do we classify
"hard" vs. "easy"?
E(ρ)


How do we classify "hard" vs. "easy"?
- How much time does it take to solve it?

How do we classify "hard" vs. "easy"?

- How much time does it take to solve it?

↳ How many computational steps using the best algorithm?

How do we classify "hard" vs. "easy"?

- How much time does it take to solve it?

↳ How many computational steps using the best algorithm?

- Could pre-compute certain answers

⇒ Can't define difficulty of a single problem. Should look at classes of problems.

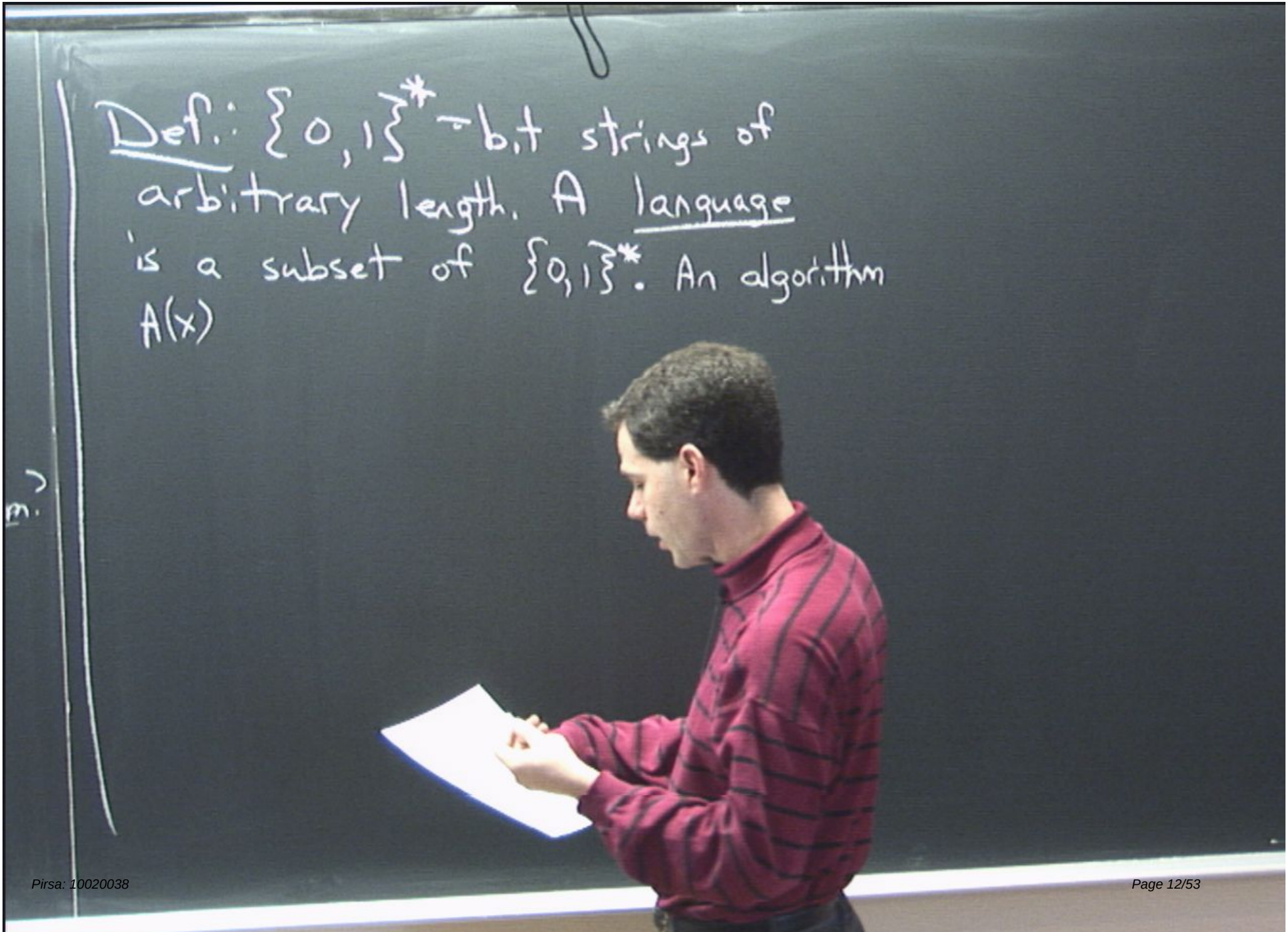
How do we classify "hard" vs. "easy"?

- How much time does it take to solve it?

↳ How many computational steps using the best algorithm?

- Could pre-compute certain answers

⇒ Can't define difficulty of a single problem. Should look at classes of problems.


Def.: $\{0,1\}^*$ - bit strings of arbitrary length. A language is a subset of $\{0,1\}^*$. An algorithm A(x)

Def.: $\{0,1\}^*$ = bit strings of arbitrary length. A *language* is a subset of $\{0,1\}^*$. An algorithm $A(x)$ *decides* a language L if $A(x)=0 \Leftrightarrow x \notin L$, $A(x)=1 \Leftrightarrow x \in L$.

Def.: $\{0,1\}^*$ = bit strings of arbitrary length. A *language* is a subset of $\{0,1\}^*$. An algorithm $A(x)$ *decides* a language L if $A(x)=0 \Leftrightarrow x \notin L$, $A(x)=1 \Leftrightarrow x \in L$.

x is an *instance* of L.

Def.: $\{0,1\}^*$ = bit strings of arbitrary length. A *language* is a subset of $\{0,1\}^*$. An algorithm $A(x)$ *decides* a language L if $A(x)=0 \Leftrightarrow x \notin L$, $A(x)=1 \Leftrightarrow x \in L$.

x is an *instance* of L.

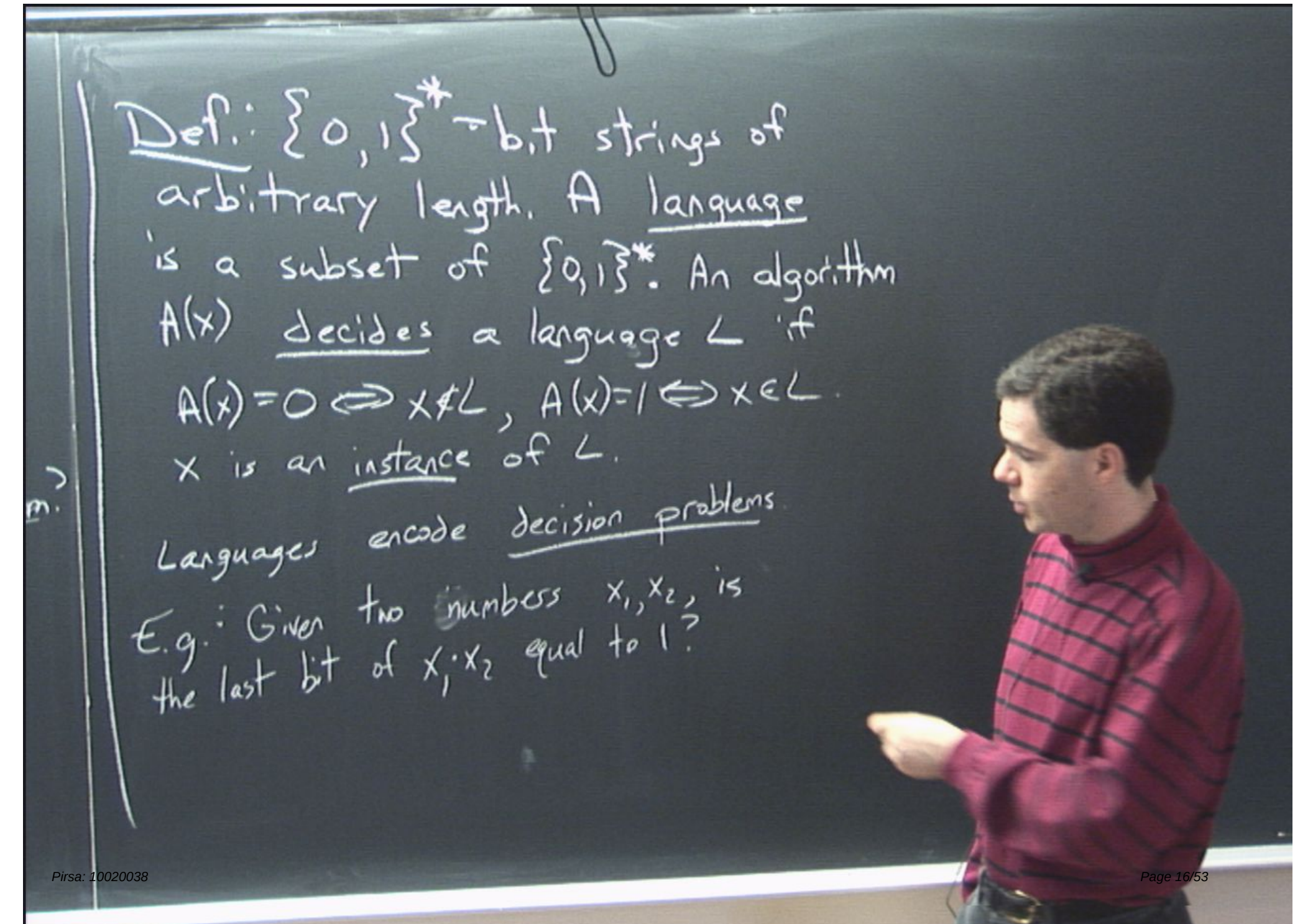
Languages encode *decision problems*.

Def.: $\{0,1\}^*$ = bit strings of arbitrary length. A *language* is a subset of $\{0,1\}^*$. An algorithm $A(x)$ *decides* a language L if $A(x)=0 \Leftrightarrow x \notin L$, $A(x)=1 \Leftrightarrow x \in L$.

x is an *instance* of L.

Languages encode *decision problems*.

E.g.: Given two numbers x_1, x_2, is the last bit of $x_1 \cdot x_2$ equal to 1?

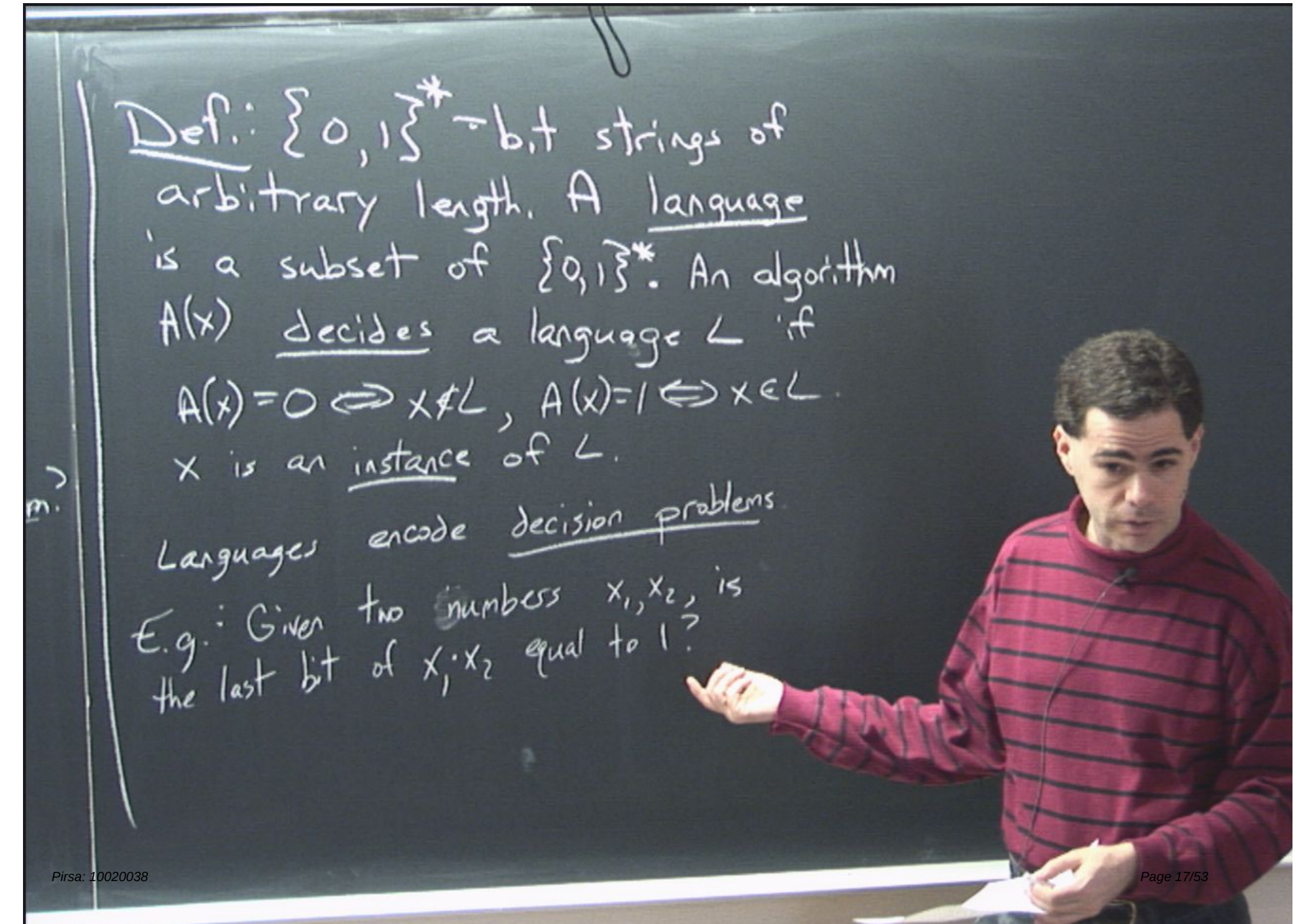


Def.: $\{0,1\}^*$ - bit strings of arbitrary length. A *language* is a subset of $\{0,1\}^*$. An algorithm $A(x)$ *decides* a language L if $A(x)=0 \Leftrightarrow x \notin L$, $A(x)=1 \Leftrightarrow x \in L$.

x is an *instance* of L.

Languages encode *decision problems*.

E.g.: Given two numbers x_1, x_2, is the last bit of $x_1 \cdot x_2$ equal to 1?



Def.: $\{0,1\}^*$ = bit strings of arbitrary length. A *language* is a subset of $\{0,1\}^*$. An algorithm $A(x)$ *decides* a language L if $A(x)=0 \Leftrightarrow x \notin L$, $A(x)=1 \Leftrightarrow x \in L$.

x is an *instance* of L.

Languages encode *decision problems*.

E.g.: Given two numbers x_1, x_2, is the last bit of $x_1 \cdot x_2$ equal to 1?

$(x_1, x_2) \in L \Leftrightarrow$ Last bit of $x_1 \cdot x_2$ is 1

Def.: $\{0,1\}^*$ - bit strings of arbitrary length. A *language* is a subset of $\{0,1\}^*$. An algorithm $A(x)$ *decides* a language L if $A(x)=0 \Leftrightarrow x \notin L$, $A(x)=1 \Leftrightarrow x \in L$.

x is an *instance* of L.

Languages encode *decision problems*.

E.g.: Given two numbers x_1, x_2, is the last bit of $x_1 \cdot x_2$ equal to 1?

$(x_1, x_2) \in L \Leftrightarrow$ Last bit of $x_1 \cdot x_2$ is 1

Def.: $\{0,1\}^*$ - bit strings of arbitrary length. A *language* is a subset of $\{0,1\}^*$. An algorithm $A(x)$ *decides* a language L if $A(x)=0 \Leftrightarrow x \notin L$, $A(x)=1 \Leftrightarrow x \in L$.

x is an *instance* of L.

Languages encode *decision problems*

E.g.: Given two numbers x_1, x_2, is the last bit of $x_1 \cdot x_2$ equal to 1?

$(x_1, x_2) \in L \Leftrightarrow$ Last bit of $x_1 \cdot x_2$ is 1

Eg. "Primality testing

Def.: $\{0,1\}^*$ = bit strings of arbitrary length. A *language* is a subset of $\{0,1\}^*$. An algorithm $A(x)$ *decides* a language L if $A(x)=0 \Leftrightarrow x \notin L$, $A(x)=1 \Leftrightarrow x \in L$.

x is an *instance* of L.

Languages encode *decision problems*

E.g.: Given two numbers x_1, x_2, is the last bit of $x_1 \cdot x_2$ equal to 1?

$(x_1, x_2) \in L \Leftrightarrow$ Last bit of $x_1 \cdot x_2$ is 1

Eg. "Primality testing"

$L = \{ p \in \mathbb{Z}^+ \mid p \text{ prime} \}$

Def.: $\{0,1\}^*$ - bit strings of arbitrary length. A *language* is a subset of $\{0,1\}^*$. An algorithm $A(x)$ *decides* a language L if $A(x)=0 \Leftrightarrow x \notin L$, $A(x)=1 \Leftrightarrow x \in L$.

x is an *instance* of L.

Languages encode *decision problems*

E.g.: Given two numbers x_1, x_2, is the last bit of $x_1 \cdot x_2$ equal to 1?

$(x_1, x_2) \in L \Leftrightarrow$ Last bit of $x_1 \cdot x_2$ is 1

Eg.: "Primality testing"

$L = \{ p \in \mathbb{Z}^+ \mid p \text{ prime} \}$

Consider the asymptotic behavior of the algorithm for large $|x|$

Def.: $\{0,1\}^*$ - bit strings of arbitrary length. A *language* is a subset of $\{0,1\}^*$. An algorithm $A(x)$ *decides* a language L if $A(x)=0 \iff x \notin L$, $A(x)=1 \iff x \in L$.

x is an *instance* of L.

Languages encode *decision problems*

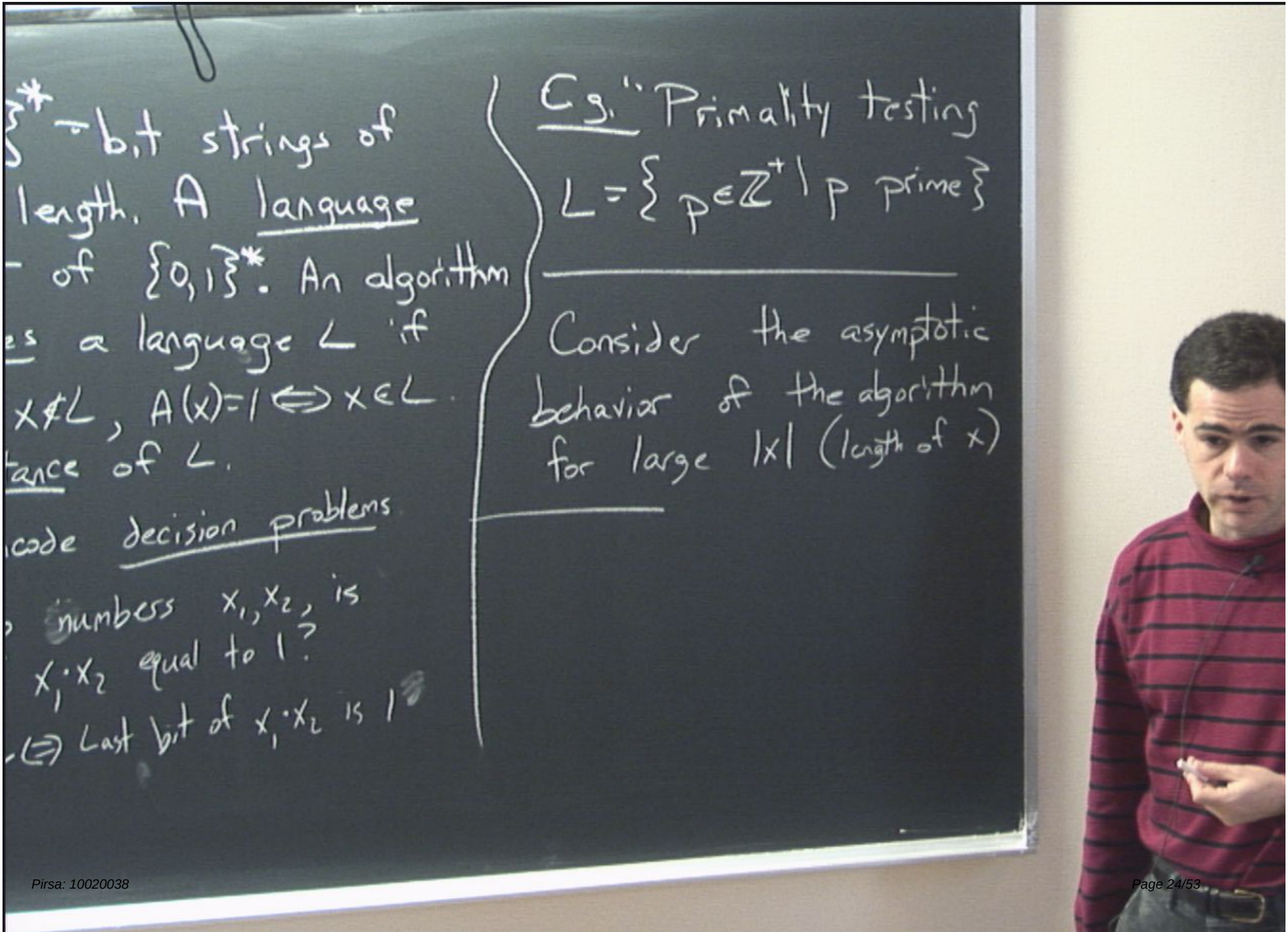
E.g.: Given two numbers x_1, x_2, is the last bit of $x_1 \cdot x_2$ equal to 1?

$(x_1, x_2) \in L \iff$ Last bit of $x_1 \cdot x_2$ is 1

Eg. "Primality testing"

$L = \{ p \in \mathbb{Z}^+ \mid p \text{ prime} \}$

Consider the asymptotic behavior of the algorithm for large $|x|$ (length of x)


$\{0,1\}^*$ - bit strings of length. A language of $\{0,1\}^*$. An algorithm es a language L if $x \notin L$, $A(x)=1 \iff x \in L$. tance of L.
code decision problems
numbers x_1, x_2, is $x_1 \cdot x_2$ equal to 1?
$\iff$ Last bit of $x_1 \cdot x_2$ is 1
Eg. "Primality testing
$L = \{ p \in \mathbb{Z}^+ \mid p \text{ prime} \}$
Consider the asymptotic behavior of the algorithm for large $|x|$ (length of x)

$\{0,1\}^*$ - bit strings of length. A language of $\{0,1\}^*$. An algorithm es a language L if $x \notin L$, $A(x)=1 \iff x \in L$. tance of L.

code decision problems

numbers x_1, x_2, is $x_1 \cdot x_2$ equal to 1?

$\iff$ Last bit of $x_1 \cdot x_2$ is 1

E.g. "Primality testing"

$L = \{ p \in \mathbb{Z}^+ \mid p \text{ prime} \}$

Consider the asymptotic behavior of the algorithm for large $|x|$ (length of x)

But different machines can differ in # of computational steps due to different architectures.

E.g. 1-Dim.

$\{0,1\}^*$ - bit strings of length. A language of $\{0,1\}^*$. An algorithm es a language L if $x \notin L$, $A(x)=1 \iff x \in L$. tance of L.

code decision problems

numbers x_1, x_2, is $x_1 \cdot x_2$ equal to 1?

$\iff$ Last bit of $x_1 \cdot x_2$ is 1

E.g.: Primality testing

$L = \{ p \in \mathbb{Z}^+ \mid p \text{ prime} \}$

Consider the asymptotic behavior of the for large $|x|$ (

But different machine differ in # of computational ste due to different architectures.

E.g.: 1-Dim. vs. 2-Dim. computer vs. full connectivity

$\{0,1\}^*$ - bit strings of length. A language of $\{0,1\}^*$. An algorithm es a language L if $x \notin L$, $A(x)=1 \iff x \in L$. tance of L.

code decision problems

numbers x_1, x_2, is $x_1 \cdot x_2$ equal to 1?

$\iff$ Last bit of $x_1 \cdot x_2$ is 1

E.g.: Primality testing

$L = \{ p \in \mathbb{Z}^+ \mid p \text{ prime} \}$

Consider the asymptotic behavior of the algorithm for large $|x|$ (length of x)

But different machines can differ in # of computational steps due to different architectures.

E.g.: 1-Dim. vs. 2-Dim. computer vs. full connectivity

Could get factor of n speed-up

$\{0,1\}^*$ - bit strings of length. A language of $\{0,1\}^*$. An algorithm es a language L if $x \notin L$, $A(x)=1 \iff x \in L$. tance of L.

code decision problems

numbers x_1, x_2, is $x_1 \cdot x_2$ equal to 1?

$\iff$ Last bit of $x_1 \cdot x_2$ is 1

E.g.: Primality testing

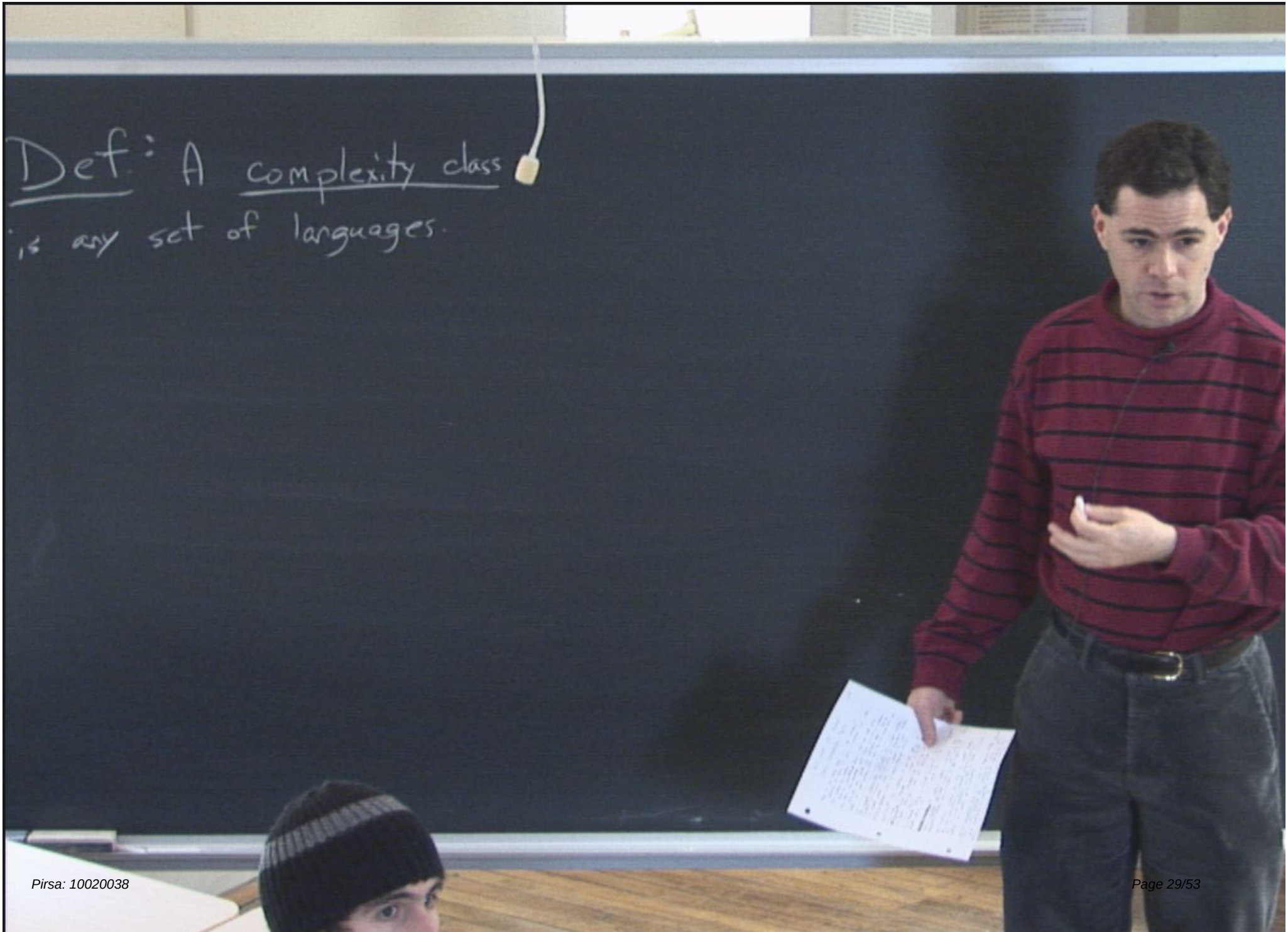
$L = \{ p \in \mathbb{Z}^+ \mid p \text{ prime} \}$

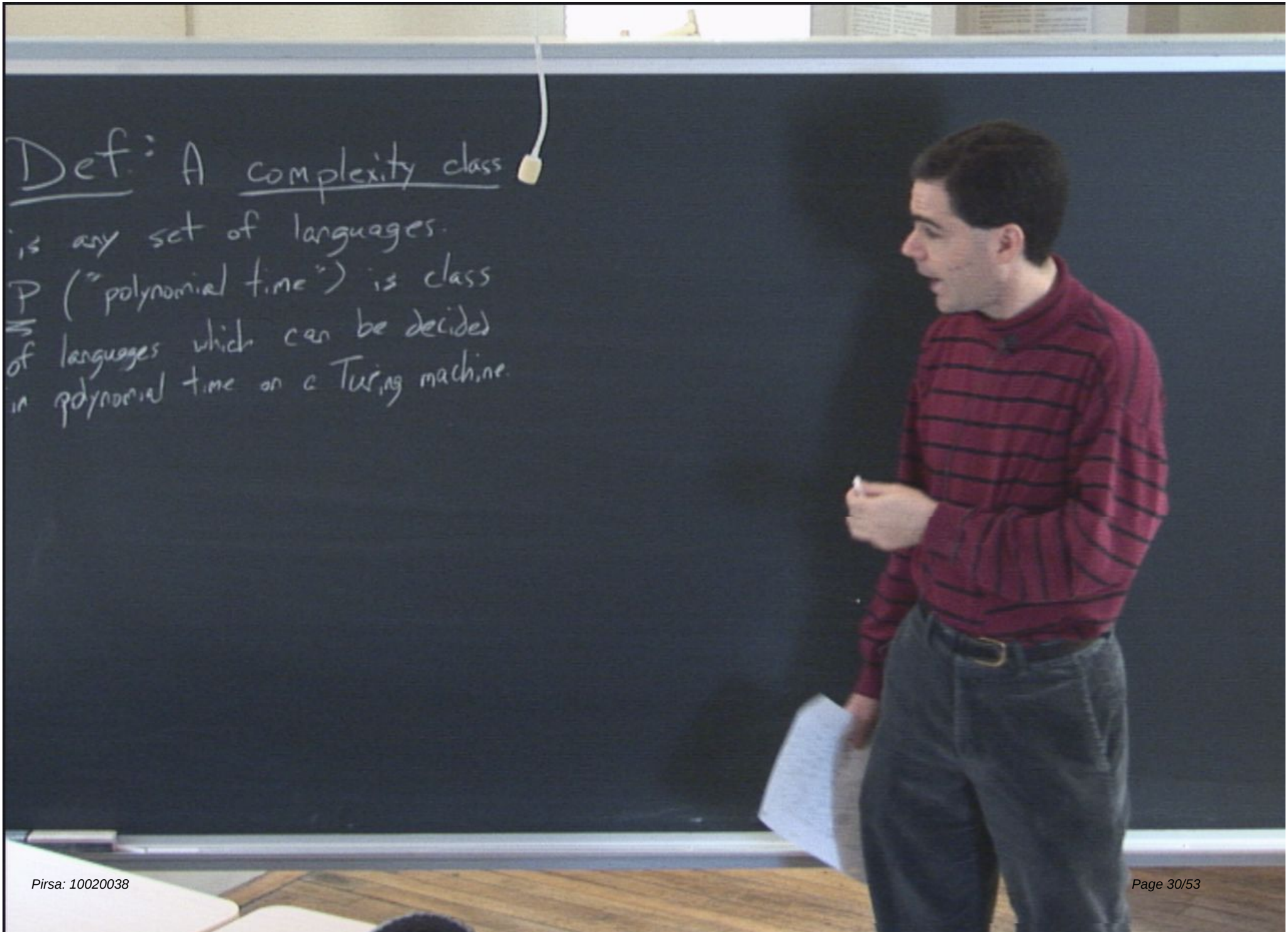
Consider the asymptotic behavior of the algorithm for large $|x|$ (length of x)

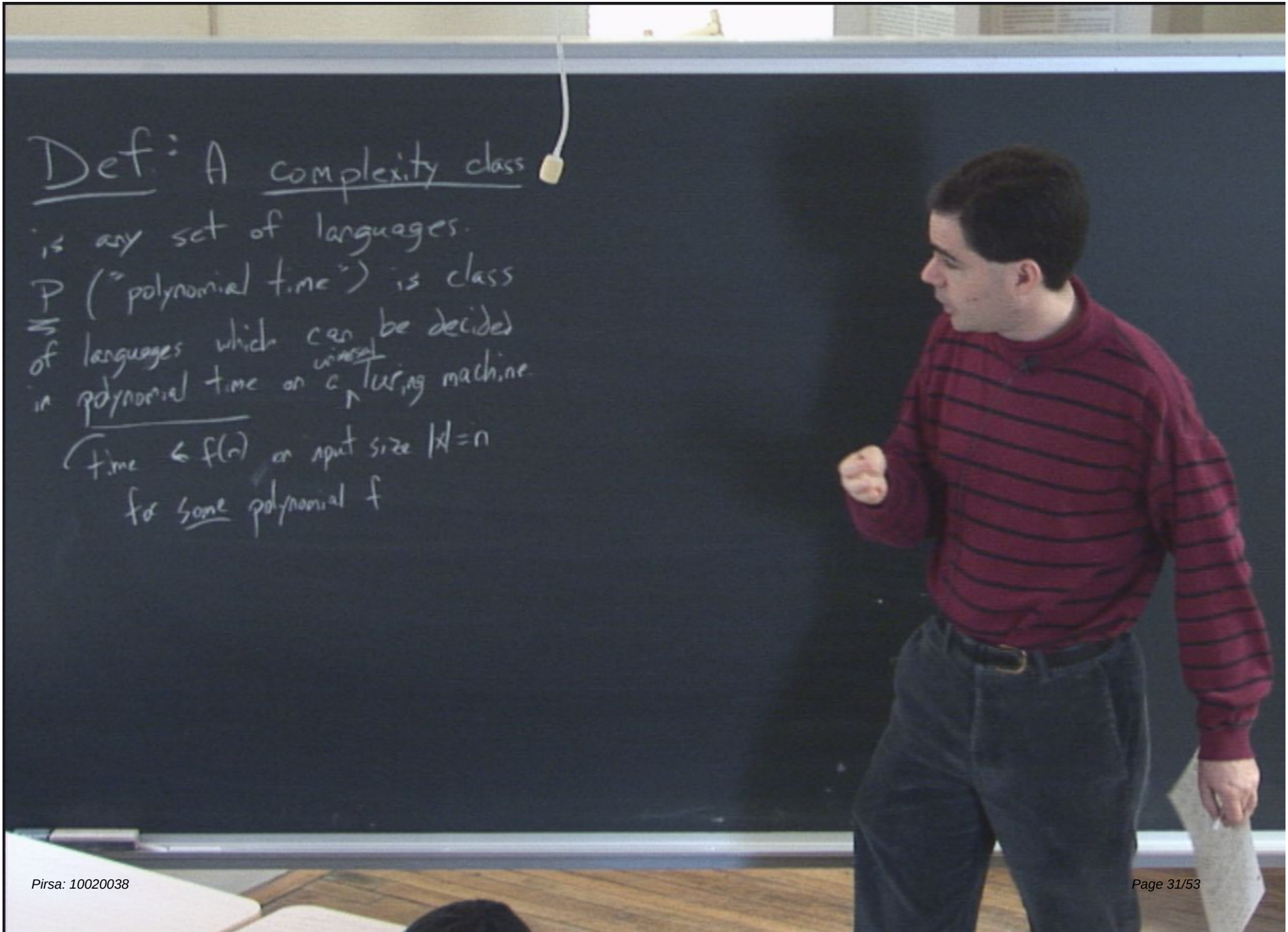
But different machines can differ in # of computational steps due to different architectures.

E.g.: 1-Dim. vs. 2-Dim. computer vs. full connectivity

Could get factor of n speed-up


Def: A complexity class
is any set of languages.


Def: A complexity class
is any set of languages.
P ("polynomial time") is class
of languages which can be decided
in polynomial time on a Turing machine.

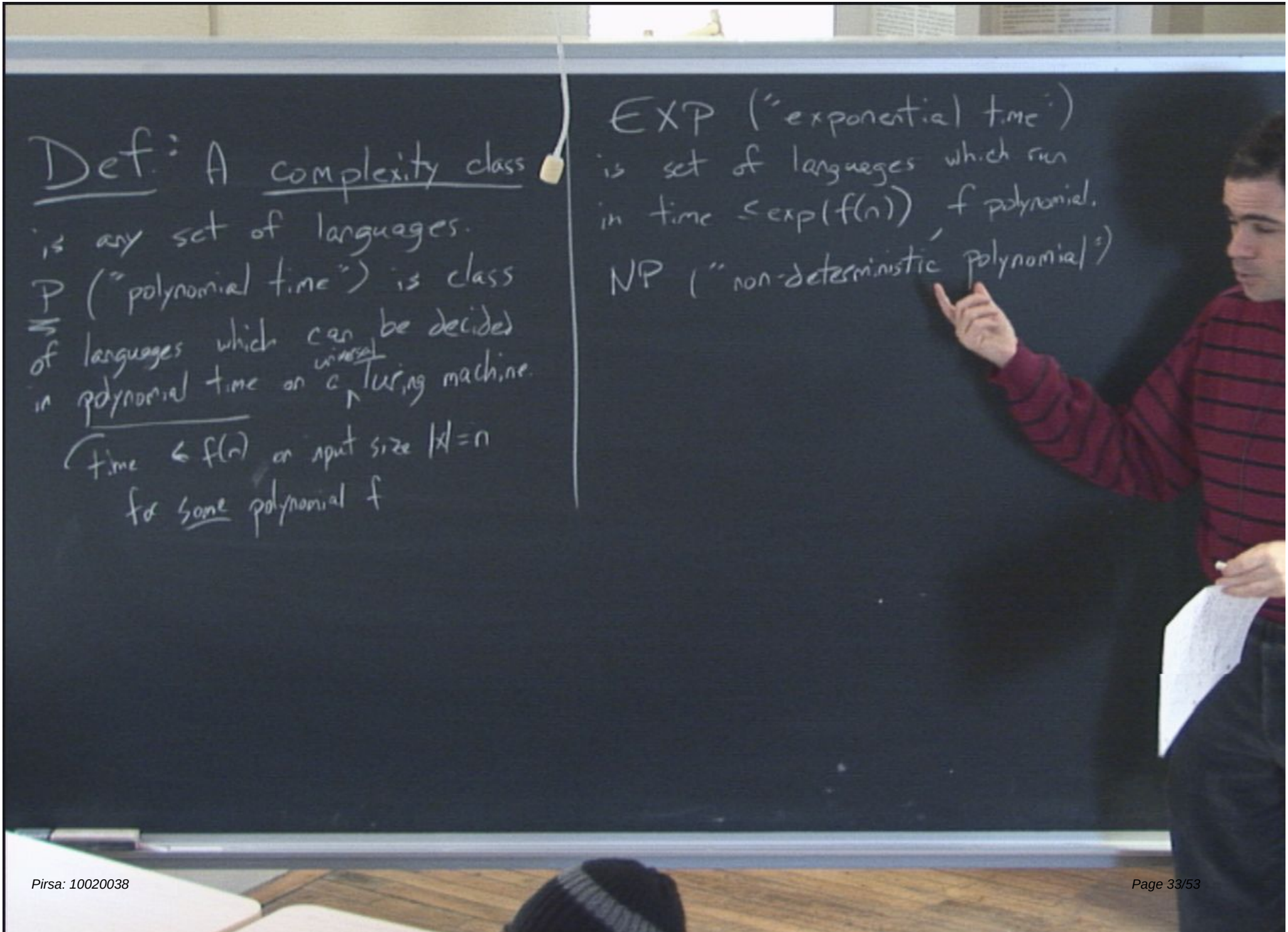

Def: A complexity class
is any set of languages.
P ("polynomial time") is class
of languages which can be decided
in polynomial time on a universal Turing machine.
(time ≤ f(n) on input size |x| = n
for some polynomial f

Def: A complexity class is any set of languages.

P ("polynomial time") is class of languages which can be decided in polynomial time on a universal Turing machine.

(time $\leq f(n)$ on input size $|x| = n$ for some polynomial f)

EXP ("exponential time") is set of languages which run in time $\leq \exp(f(n))$, f polynomial.


Def: A complexity class
is any set of languages.
P ("polynomial time") is class
of languages which can be decided
in polynomial time on a universal Turing machine.
(time ≤ f(n) on input size |x| = n
for some polynomial f
EXP ("exponential time")
is set of languages which run
in time ≤ exp(f(n)), f polynomial.
NP ("non-deterministic polynomial")

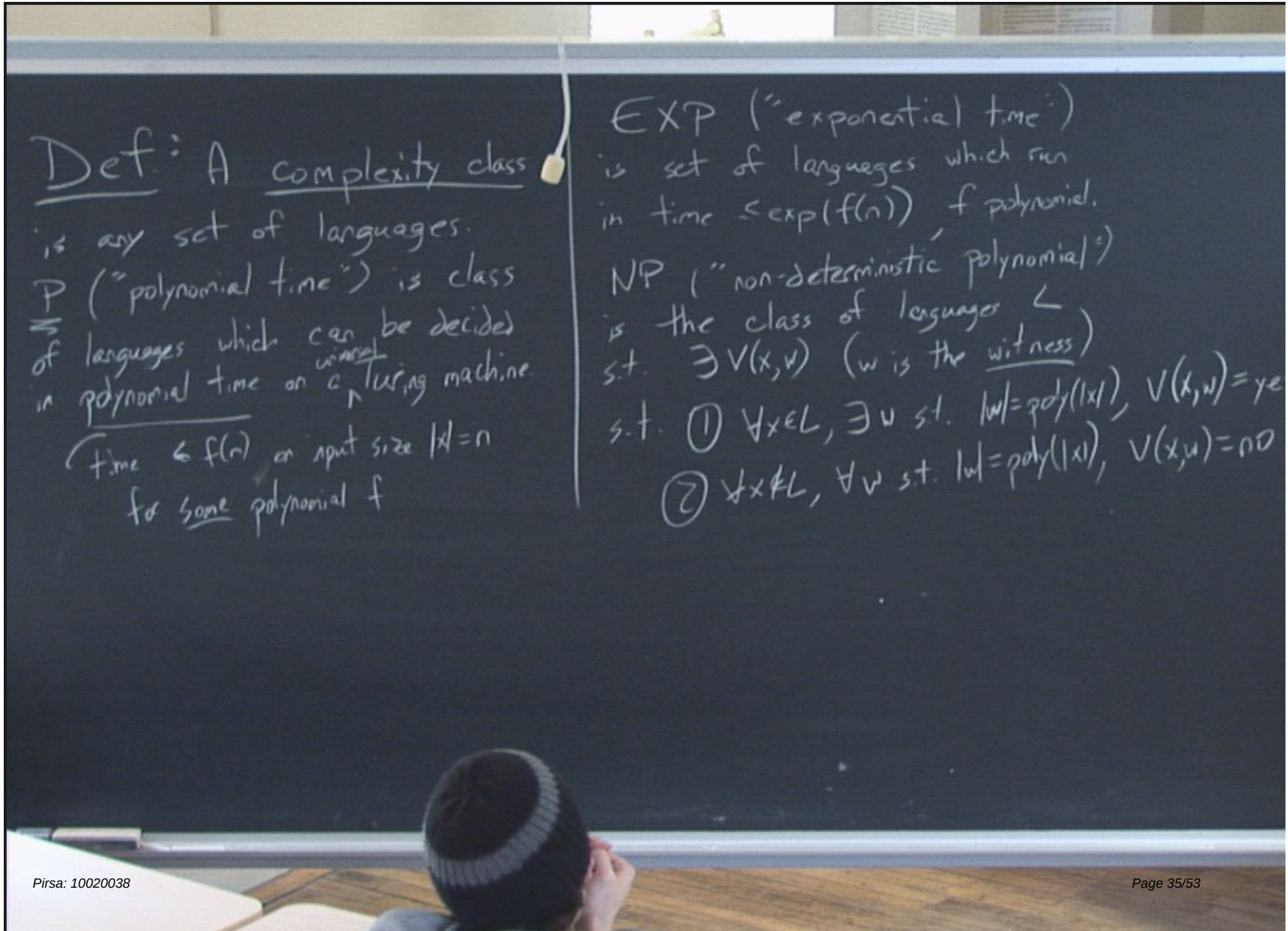
Def: A complexity class is any set of languages.

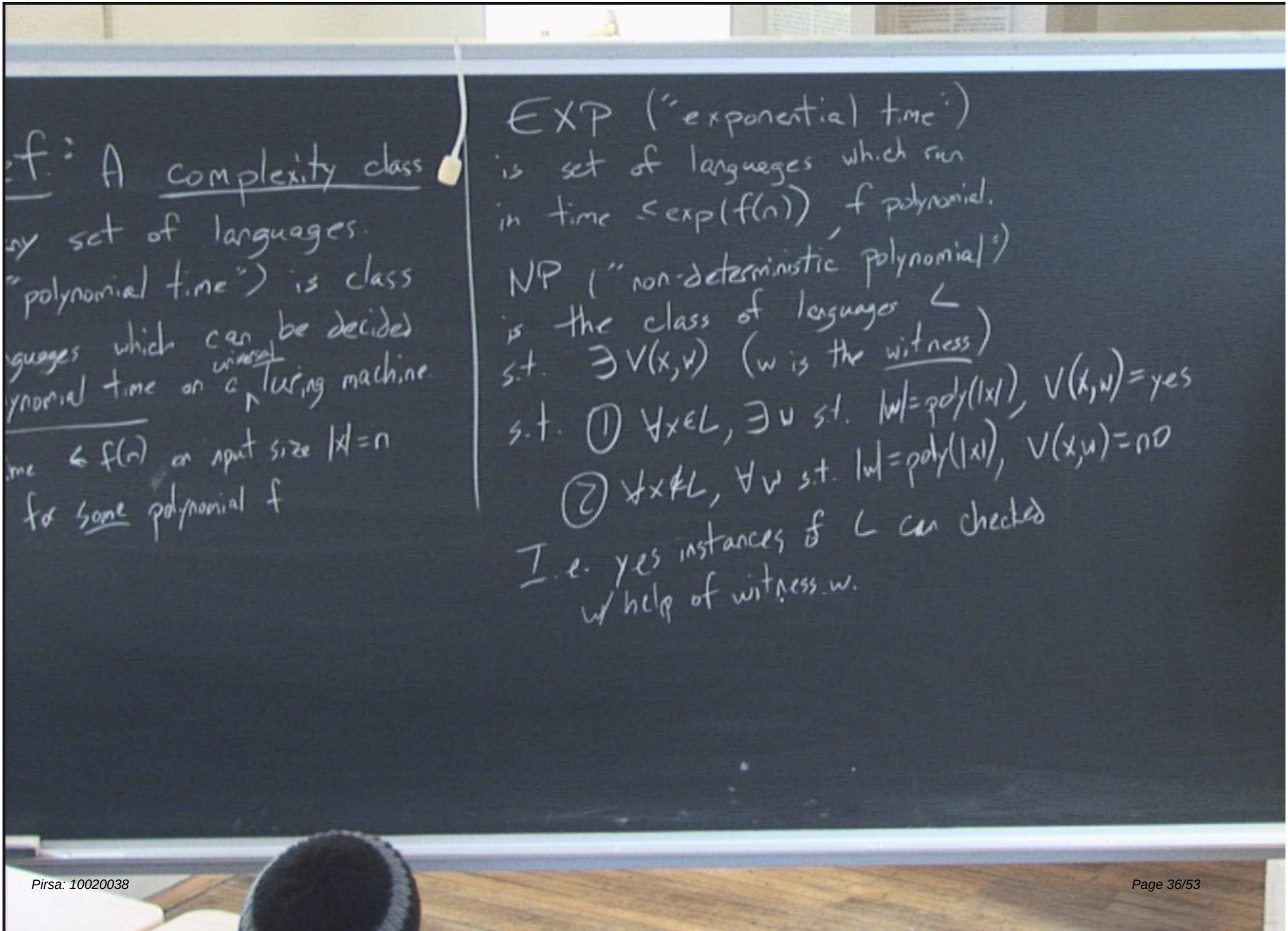
P ("polynomial time") is class of languages which can be decided in polynomial time on a universal Turing machine.

(time $\leq f(n)$ on input size $|x| = n$ for some polynomial f)

EXP ("exponential time") is set of languages which run in time $\leq \exp(f(n))$, f polynomial.

NP ("non-deterministic polynomial") is the class of languages L s.t. $\exists V(x, w)$ (w is the witness)


Def: A complexity class
is any set of languages.
P ("polynomial time") is class
of languages which can be decided
in polynomial time on a universal Turing machine.
(time ≤ f(n) on input size |x| = n
for some polynomial f
EXP ("exponential time")
is set of languages which run
in time ≤ exp(f(n)), f polynomial.
NP ("non-deterministic polynomial")
is the class of languages L
s.t. ∃ V(x,w) (w is the witness)
s.t. ① ∀x∈L, ∃ w s.t. |w| = poly(|x|), V(x,w) = ye
② ∀x∉L, ∀ w s.t. |w| = poly(|x|), V(x,w) = no


f: A complexity class
y set of languages.
"polynomial time") is class
guages which can be decided
ynomial time on a universal Turing machine.
me ≤ f(n) on input size |x| = n
for some polynomial f
EXP ("exponential time")
is set of languages which run
in time ≤ exp(f(n)), f polynomial.
NP ("non-deterministic polynomial")
is the class of languages L
s.t. ∃ V(x,w) (w is the witness)
s.t. ① ∀ x∈L, ∃ w s.t. |w| = poly(|x|), V(x,w) = yes
② ∀ x∉L, ∀ w s.t. |w| = poly(|x|), V(x,w) = no
I.e. yes instances of L can checked
w/ help of witness w.

...ty class

...ges.

...s class

...e decided

...ing machine.

...e $|x|=n$

EXP ("exponential time")

is set of languages which run in time $\leq \exp(f(n))$, f polynomial.

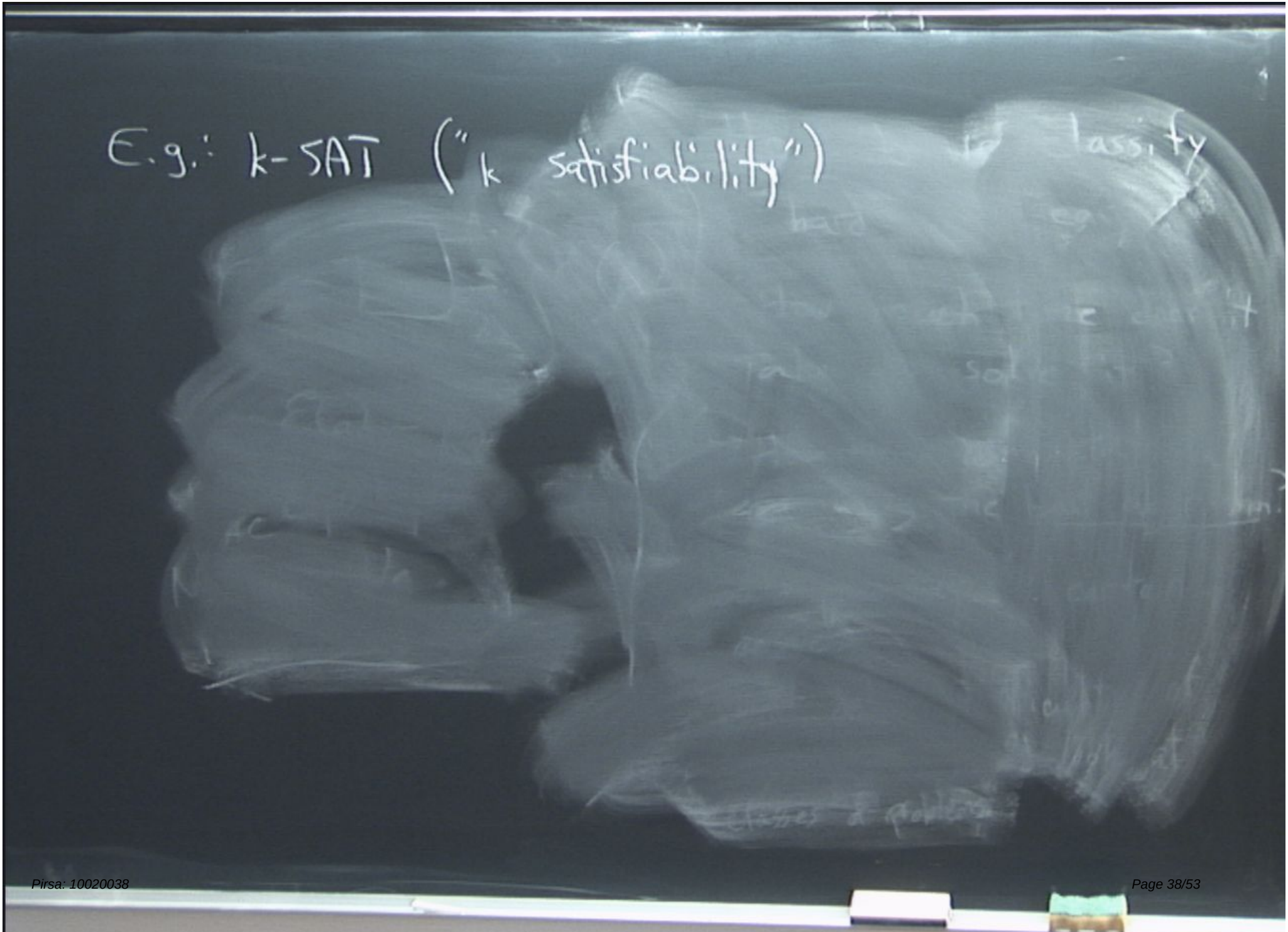
NP ("non-deterministic polynomial")

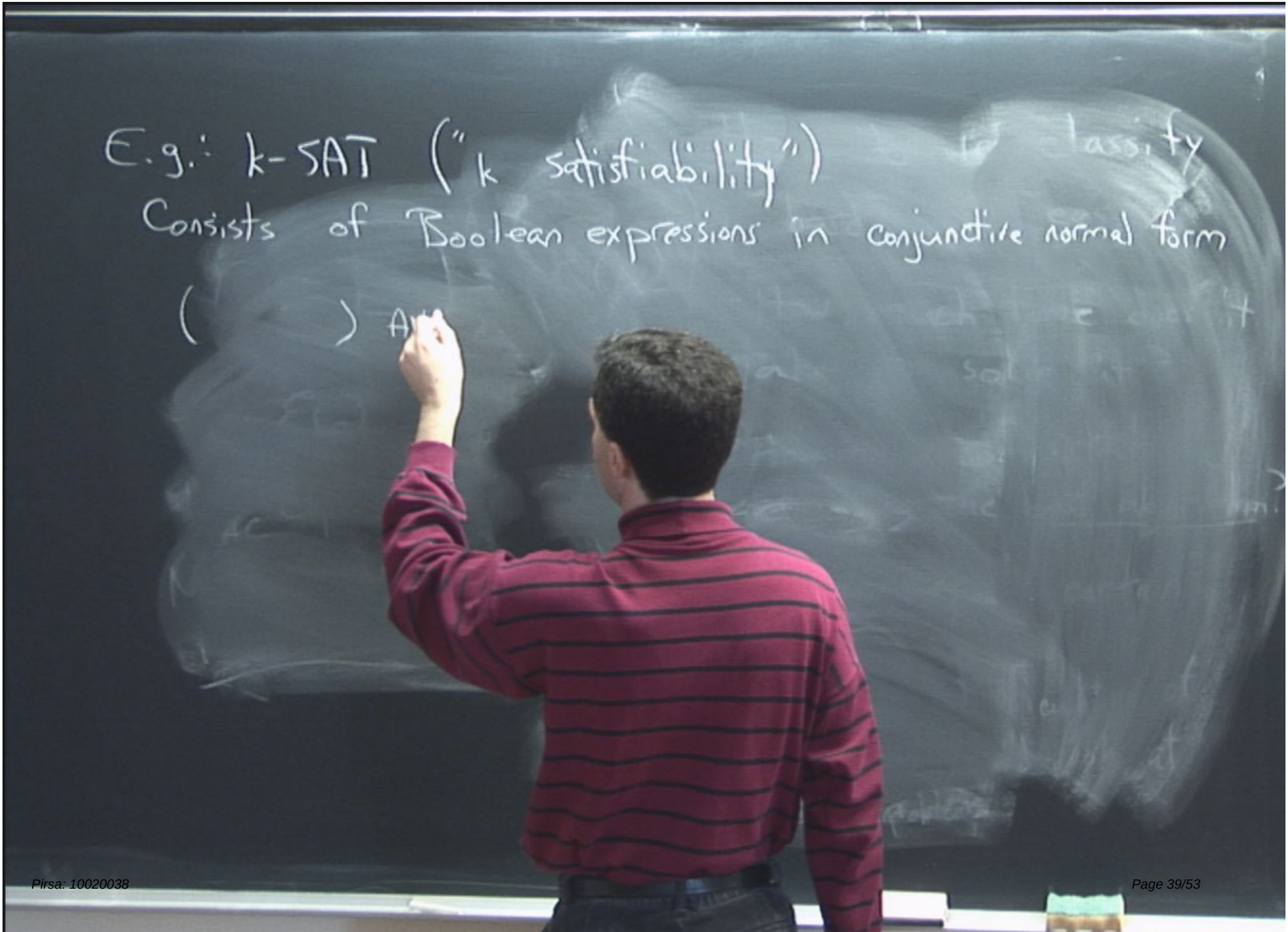
is the class of languages L s.t. $\exists V(x,w)$ (w is the witness), $V(x,w)$ runs in polynomial time

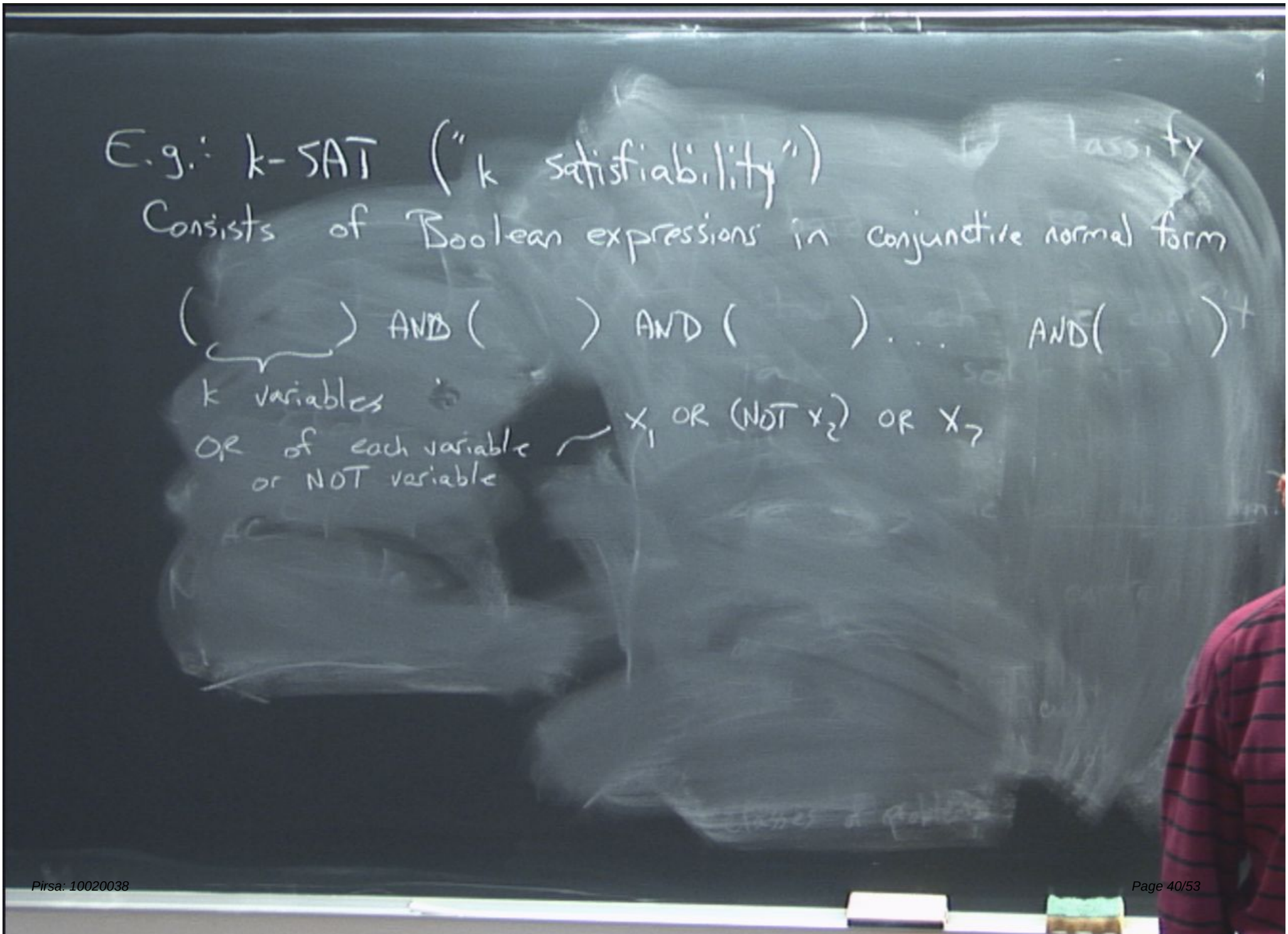
s.t. ① $\forall x \in L$, $\exists w$ s.t. $|w| = \text{poly}(|x|)$, $V(x,w) = \text{yes}$

② $\forall x \notin L$, $\forall w$ s.t. $|w| = \text{poly}(|x|)$, $V(x,w) = \text{no}$

I.e. yes instances of L can checked w/ help of witness w.


E.g.: k-SAT ("k satisfiability")


E.g.: k-SAT ("k satisfiability")
Consists of Boolean expressions in conjunctive normal form
( ) A


E.g.: k-SAT ("k satisfiability")
Consists of Boolean expressions in conjunctive normal form
( ) AND ( ) AND ( ) . . . AND ( )
k variables
OR of each variable or NOT variable
x_1 OR (NOT x_2) OR x_7

E.g.: k-SAT ("k satisfiability")

Consists of Boolean expressions in conjunctive normal form

( ) AND ( ) AND ( ) ... AND ( )

≤k variables

OR of each variable or NOT variable ~ x_1 OR (NOT x_2) OR x_7

A formula is in k-SAT (yes instance) if

∃ assignment of variables ($x_1 = ?$, $x_2 = ?$, $x_3 = ?$,)

s.t. formula evaluates to TRUE

E.g.: k-SAT ("k satisfiability")

Consists of Boolean expressions in conjunctive normal form

( ) AND ( ) AND ( ) ... AND ( )

≤k variables

OR of each variable or NOT variable ~ x_1 OR (NOT x_2) OR x_7

A formula is in k-SAT (yes instance) if

∃ assignment of variables ($x_1 = ?, x_2 = ?, x_3 = ?, \ldots$)

s.t. formula evaluates to TRUE

(x_1 OR x_2 OR NOT x_3) AND (NOT x_1 OR x_2 OR x_3)

E.g.: k-SAT ("k satisfiability")

Consists of Boolean expressions in conjunctive normal form

( ) AND ( ) AND ( ) ... AND ( )

≤k variables

OR of each variable or NOT variable ~ x_1 OR (NOT x_2) OR x_7

A formula is in k-SAT (yes instance) if

∃ assignment of variables ($x_1 = ?, x_2 = ?, x_3 = ?, \ldots$)

s.t. formula evaluates to TRUE

(x_1 OR x_2 OR NOT x_3) AND (NOT x_1 OR x_2 OR x_3)

x_1 = TRUE, x_2 = TRUE, x_3 = FALSE

E.g.: k-SAT ("k satisfiability")

Consists of Boolean expressions in conjunctive normal form

( ) AND ( ) AND ( ) ... AND ( )

≤k variables

OR of each variable or NOT variable ~ x_1 OR (NOT x_2) OR x_7

A formula is in k-SAT (yes instance) if

∃ assignment of variables ($x_1 = ?$, $x_2 = ?$, $x_3 = ?$,)

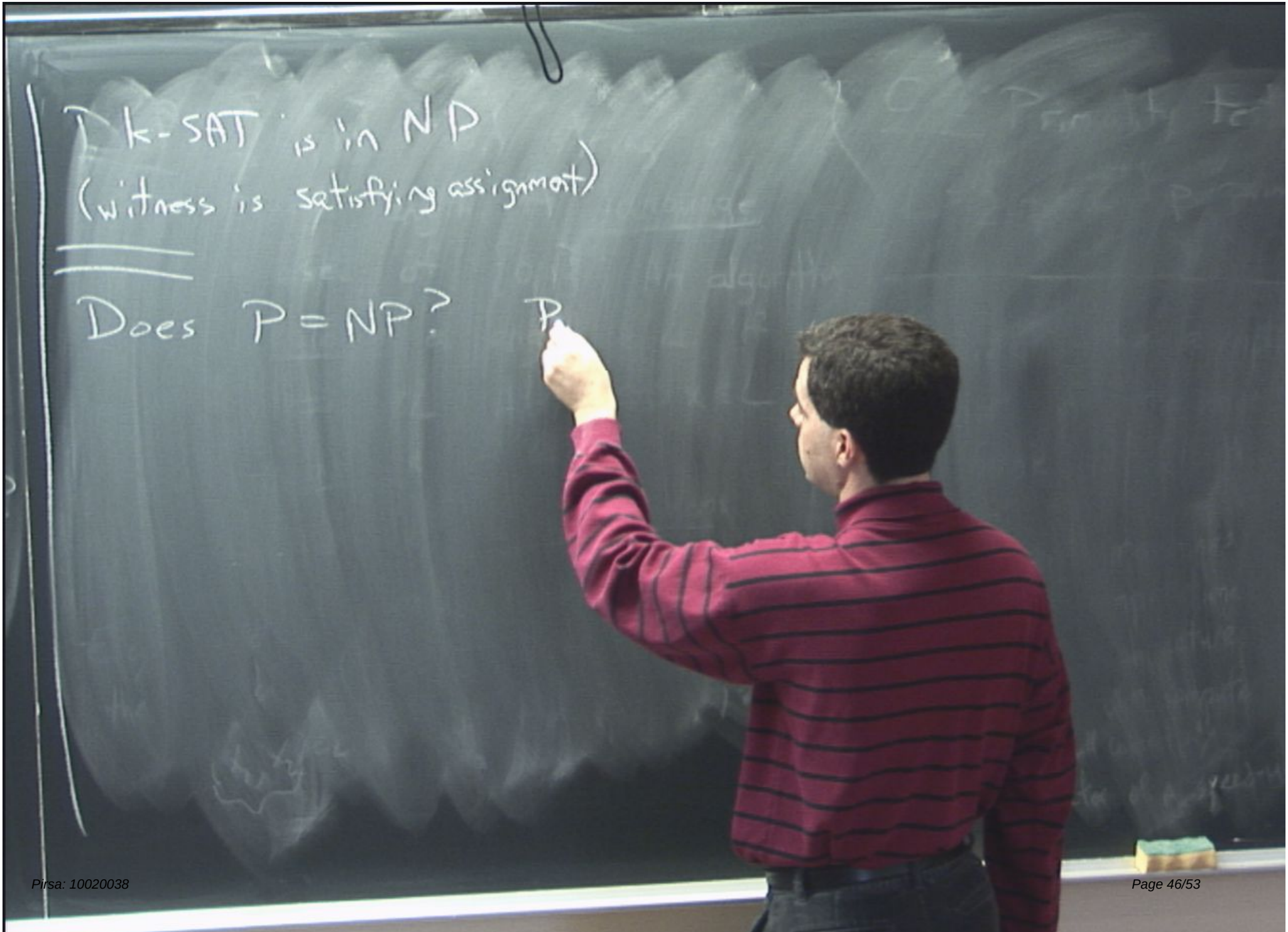
s.t. formula evaluates to TRUE

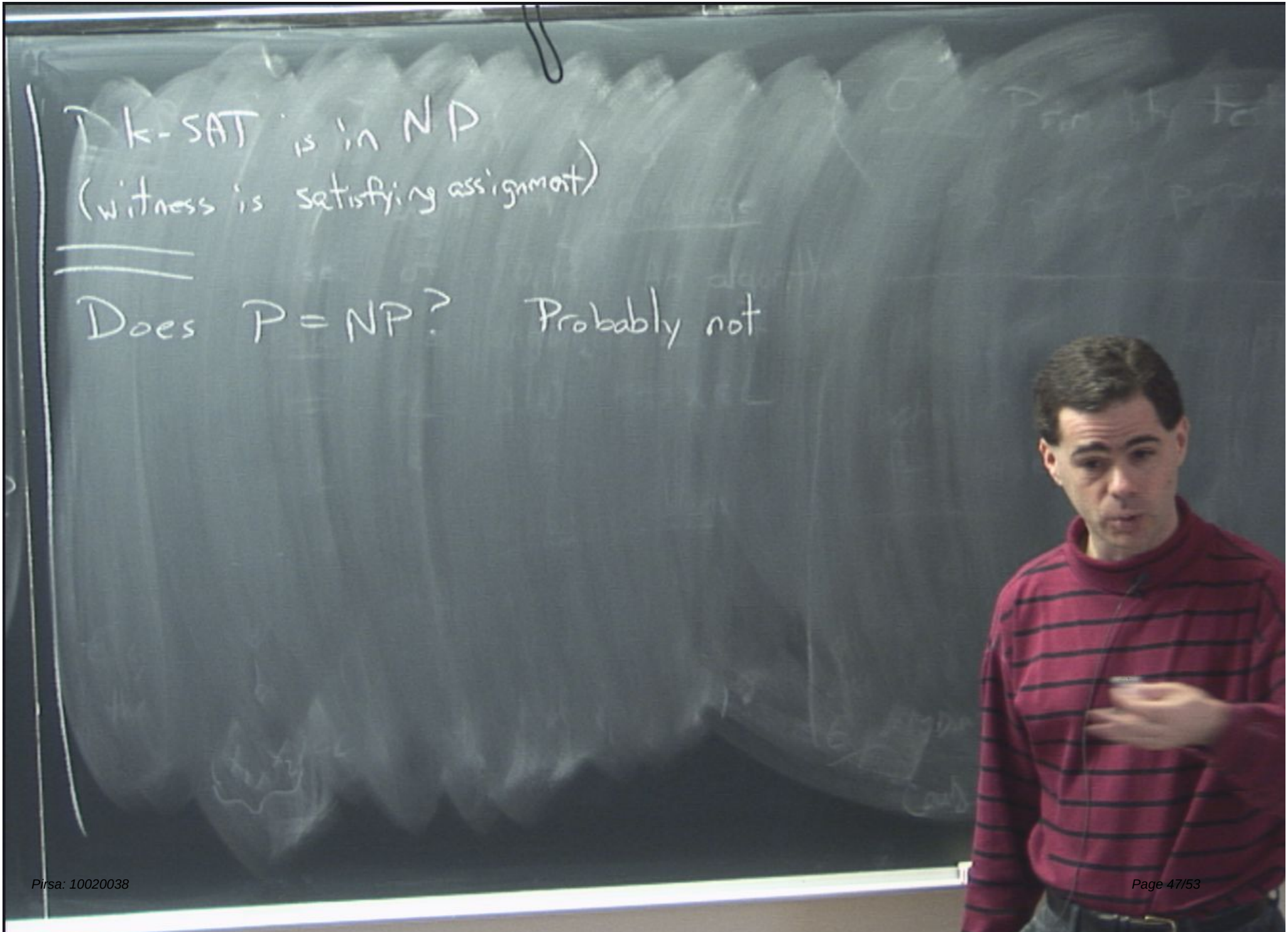
(x_1 OR x_2 OR NOT x_3) AND (NOT x_1 OR x_2 OR x_3)

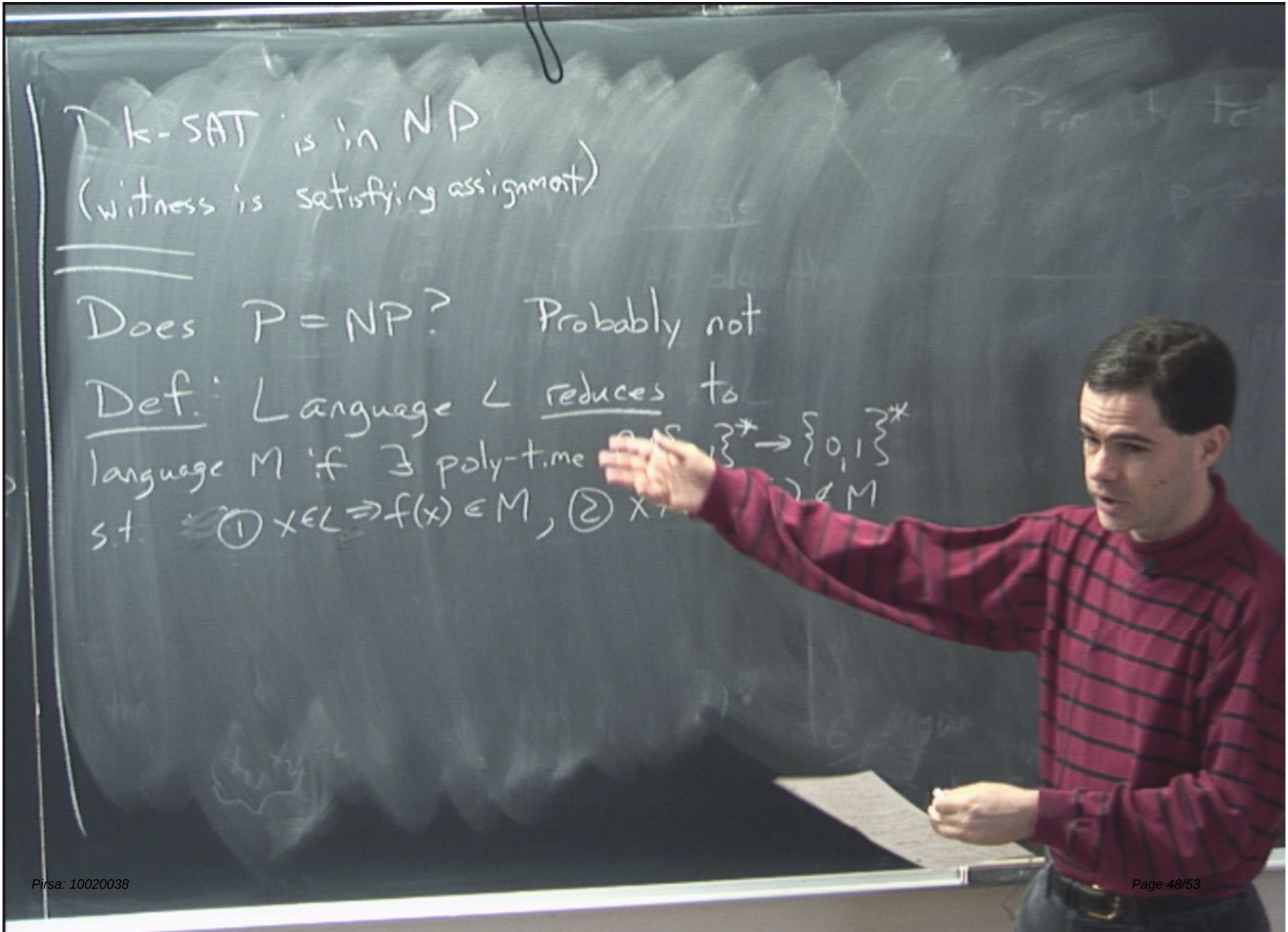
x_1 = TRUE, x_2 = TRUE, x_3 = FALSE — witness

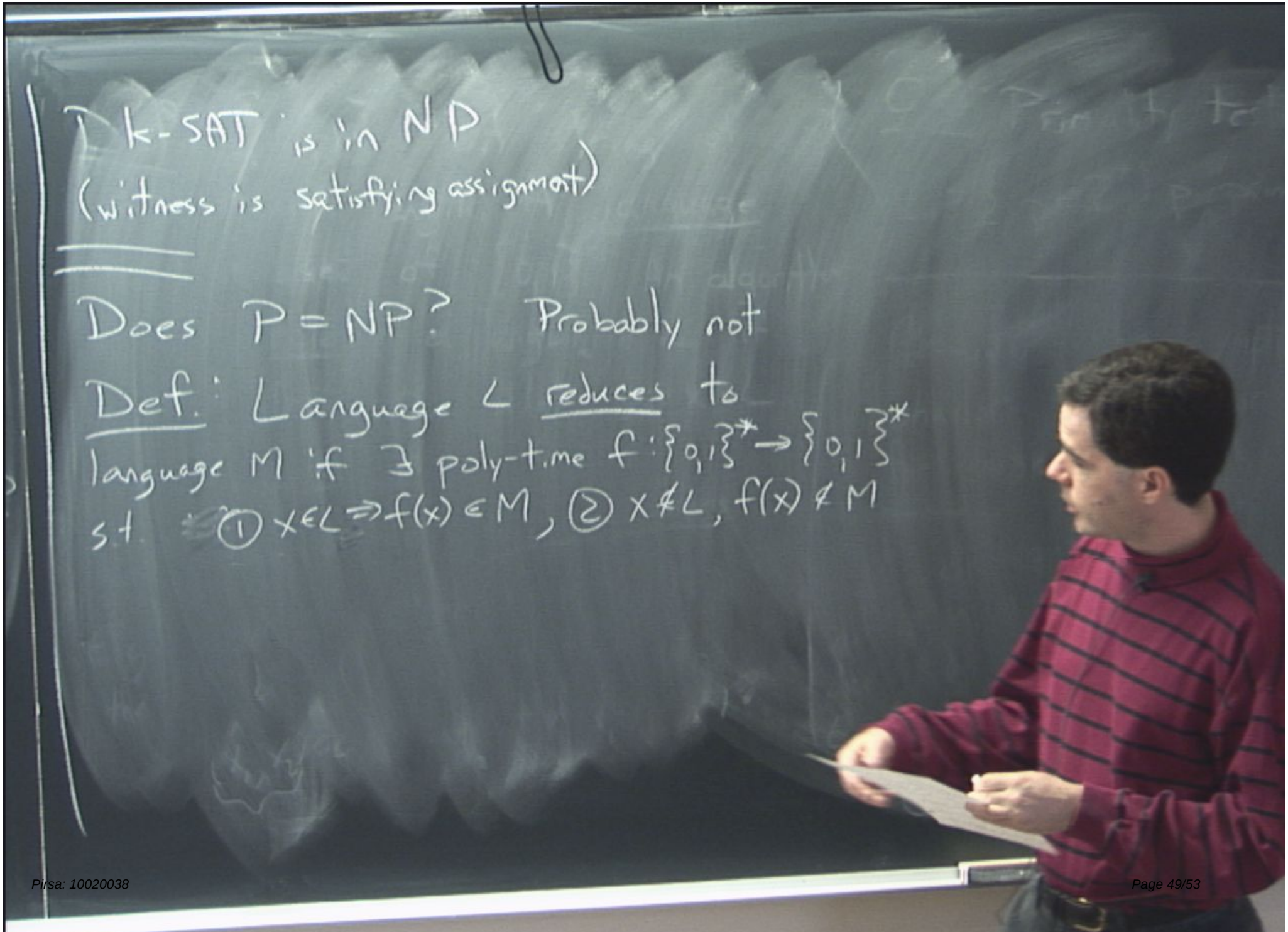
k-SAT is in NP

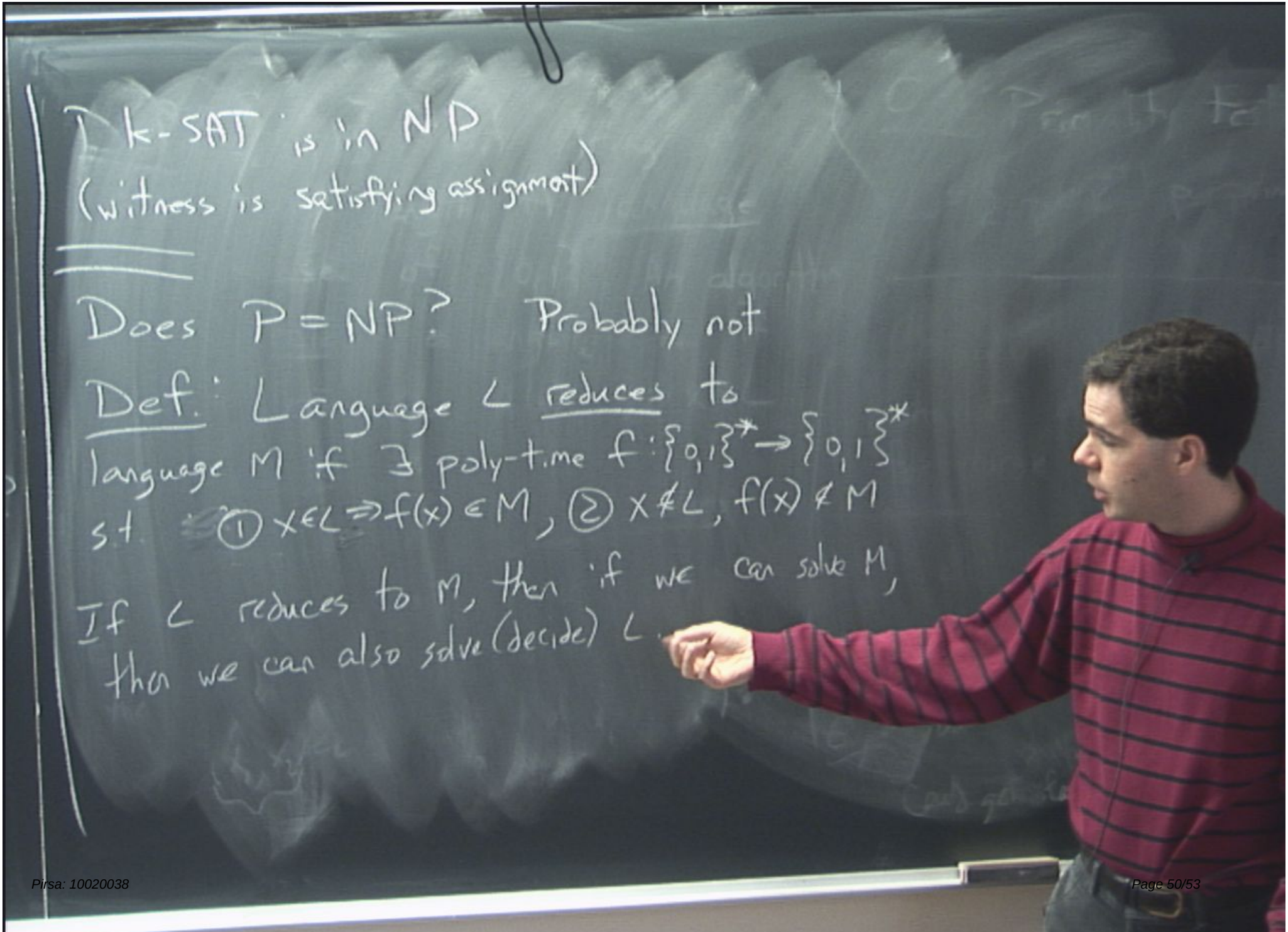
(witness is satisfying assignment)

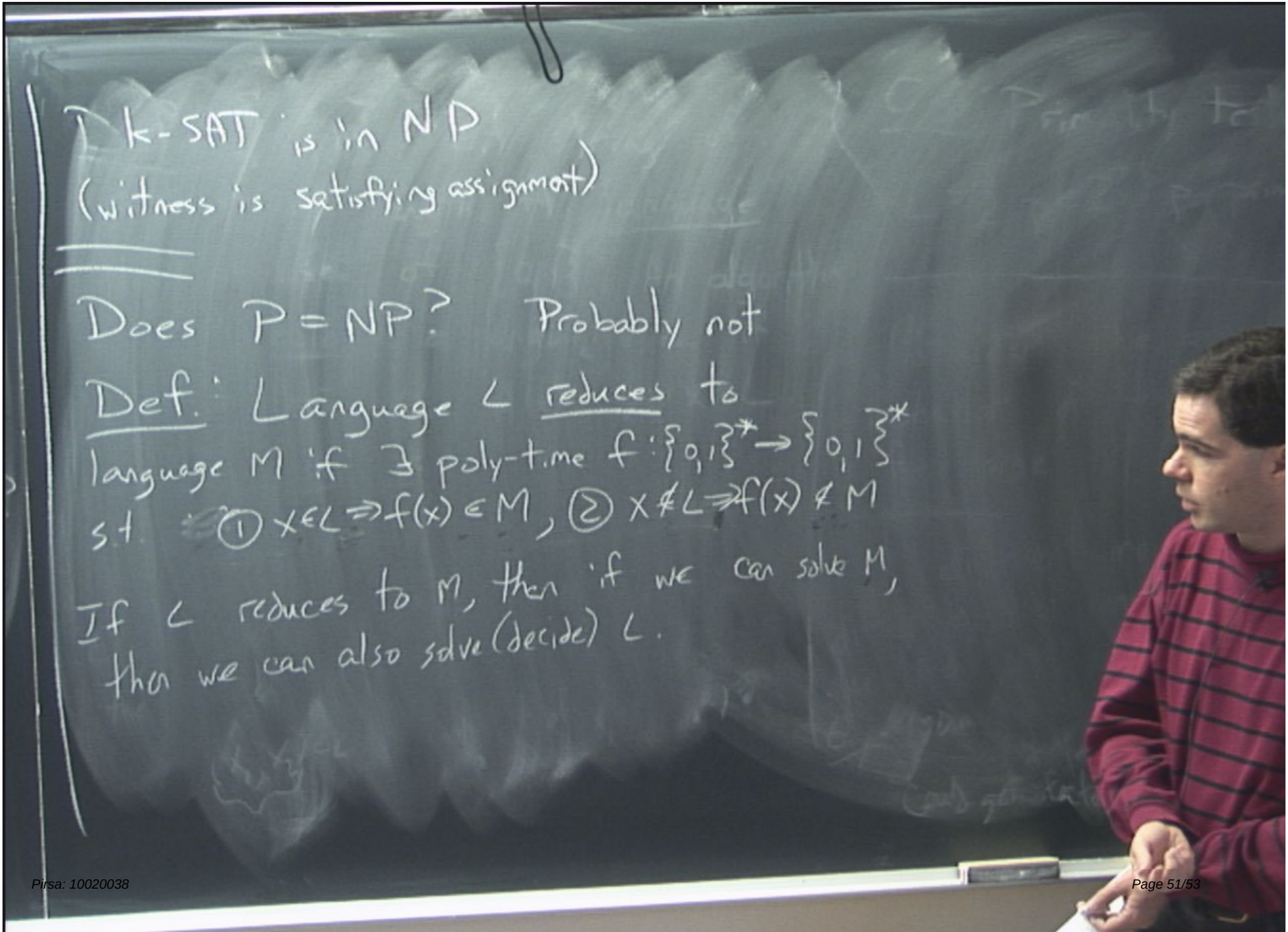

k-SAT is in NP
(witness is satisfying assignment)
Does P=NP? P


k-SAT is in NP
(witness is satisfying assignment)
Does P = NP? Probably not


k-SAT is in NP
(witness is satisfying assignment)
Does P = NP? Probably not
Def: Language L reduces to
language M if ∃ poly-time
s.t. ① x∈L ⇒ f(x) ∈ M, ②


k-SAT is in NP
(witness is satisfying assignment)
Does P = NP? Probably not
Def: Language L reduces to
language M if ∃ poly-time f: {0,1}* → {0,1}*
s.t. ① x∈L ⇒ f(x)∈M, ② x∉L, f(x)∉M


k-SAT is in NP
(witness is satisfying assignment)
Does P = NP? Probably not
Def: Language L reduces to
language M if ∃ poly-time f: {0,1}* → {0,1}*
s.t. ① x∈L ⇒ f(x)∈M, ② x∉L, f(x)∉M
If L reduces to M, then if we can solve M,
then we can also solve (decide) L.


k-SAT is in NP
(witness is satisfying assignment)
Does P = NP? Probably not
Def: Language L reduces to
language M if ∃ poly-time f: {0,1}* → {0,1}*
s.t. ① x∈L ⇒ f(x)∈M, ② x∉L ⇒ f(x)∉M
If L reduces to M, then if we can solve M,
then we can also solve (decide) L.

...T is in NP

... is satisfying assignment)

P = NP? Probably not

... Language L reduces to

... M if ∃ poly-time $f: \{0,1\}^* \to \{0,1\}^*$

① $x \in L \Rightarrow f(x) \in M$, ② $x \notin L \Rightarrow f(x) \notin M$

... reduces to M, then if we can solve M,

... we can also solve (decide) L.

Def: A language M is complete for complexity class C if $\forall L \in C$

...T is in NP
... is satisfying assignment)

P = NP? Probably not

: Language L reduces to
... M if $\exists$ poly-time $f: \{0,1\}^* \to \{0,1\}^*$
... ① $x \in L \Rightarrow f(x) \in M$, ② $x \notin L \Rightarrow f(x) \notin M$

... reduces to M, then if we can solve M,
... we can also solve (decide) L.

Def: A language M is complete for complexity class C if $\forall L \in C$, L reduces to M.