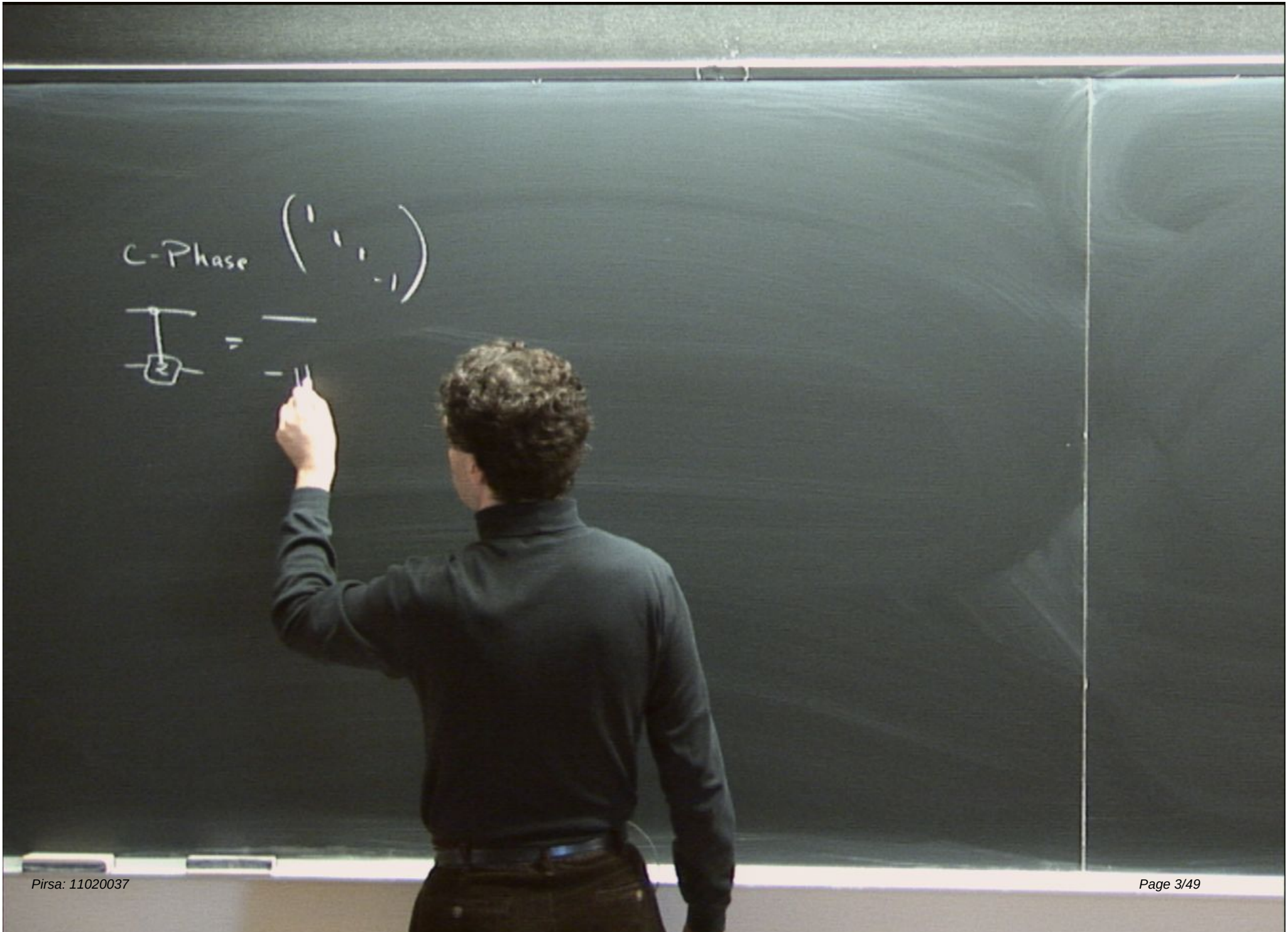
Title: Quantum Information Review - Lecture 5

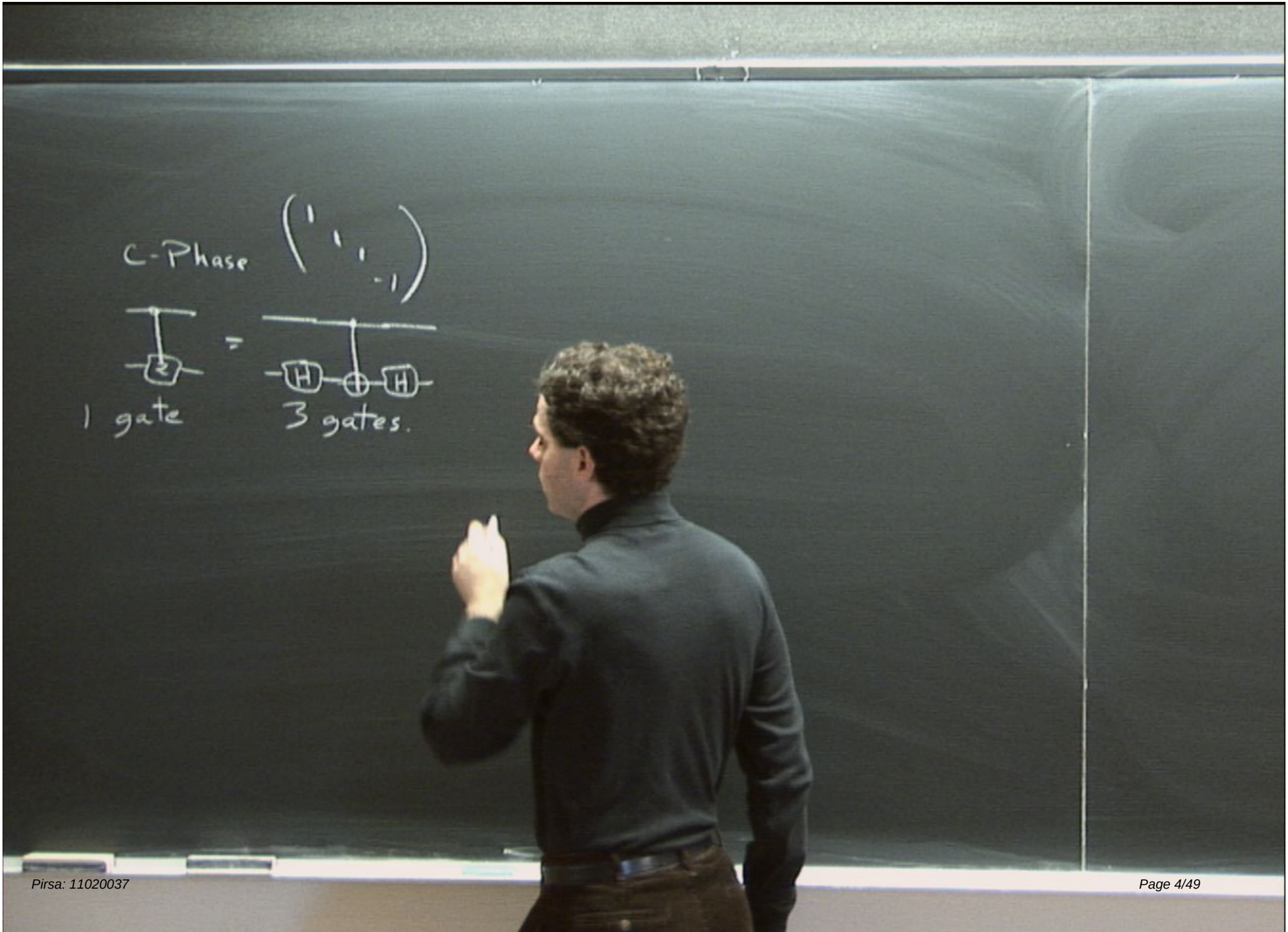
Date: Feb 18, 2011 09:00 AM

URL: http://pirsa.org/11020037

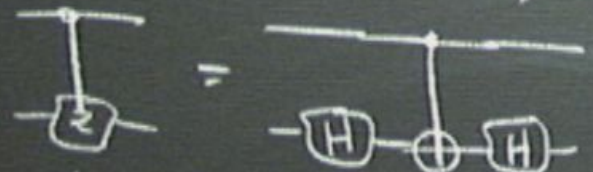
Abstract:


PERIMETER
INSTITUTE FOR THEORETICAL PHYSICS


C-Phase
(1 1 1 -1)

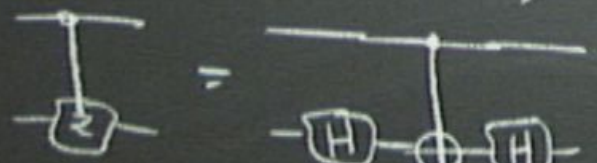

C-Phase
1 gate
3 gates.

C-Phase $\begin{pmatrix} 1 & & & \\ & 1 & & \\ & & 1 & \\ & & & -1 \end{pmatrix}$



1 gate 3 gates.

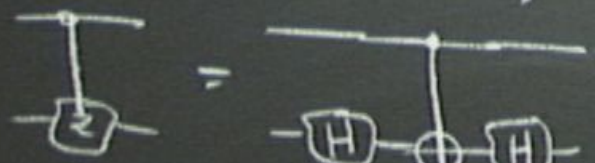
C-Phase $(1, 1, 1, -1)$



1 gate = 3 gates.

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

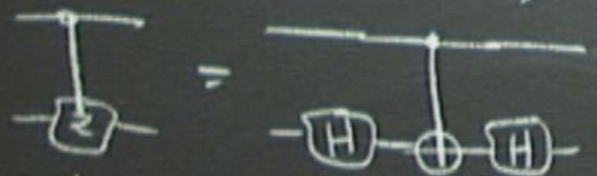
C-Phase $\begin{pmatrix} 1 & & & \\ & 1 & & \\ & & 1 & \\ & & & -1 \end{pmatrix}$



1 gate 3 gates.

Def.: $\{0,1\}^*$ is the set of bit strings of any length.
A language is a subset of $\{0$

C-Phase $(1, 1, 1, -1)$



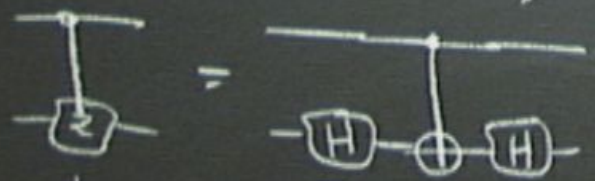
1 gate = 3 gates.

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

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

$\{0,1\}^*$

C-Phase $(1, 1, 1, -1)$



1 gate 3 gates.

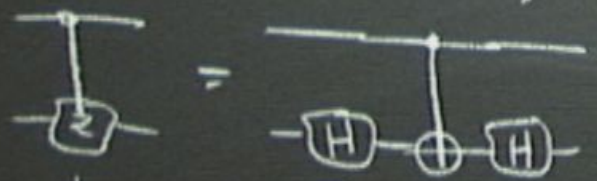
Def.: $\{0,1\}^*$ is the set of bit strings of any length.

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

$\{0,1\}^*$ is set of possible inputs.

Language L is set of "

C-Phase $\begin{pmatrix} 1 & & & \\ & 1 & & \\ & & 1 & \\ & & & -1 \end{pmatrix}$



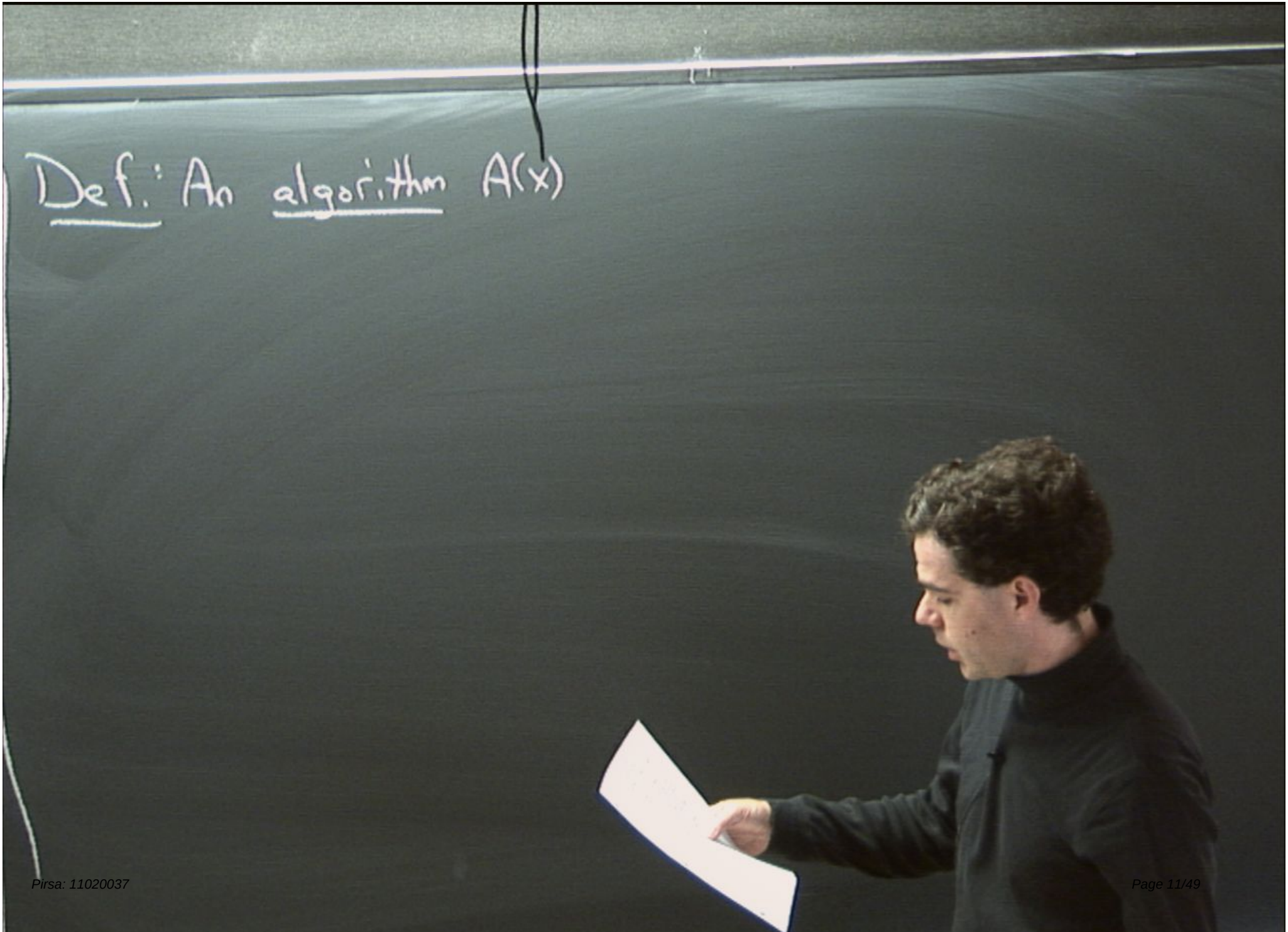
1 gate 3 gates.

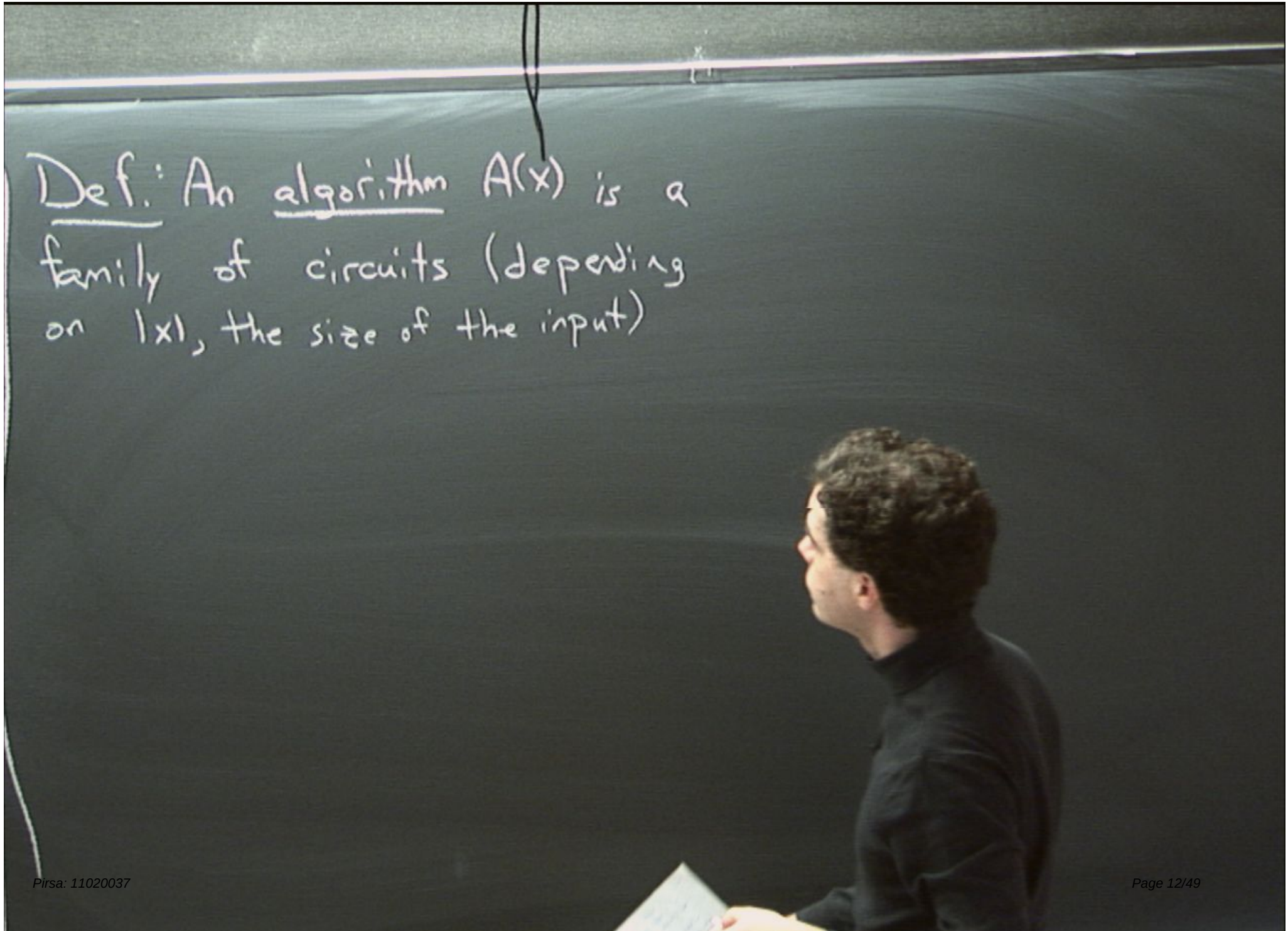
Def.: $\{0,1\}^*$ is the set of bit strings of any length.

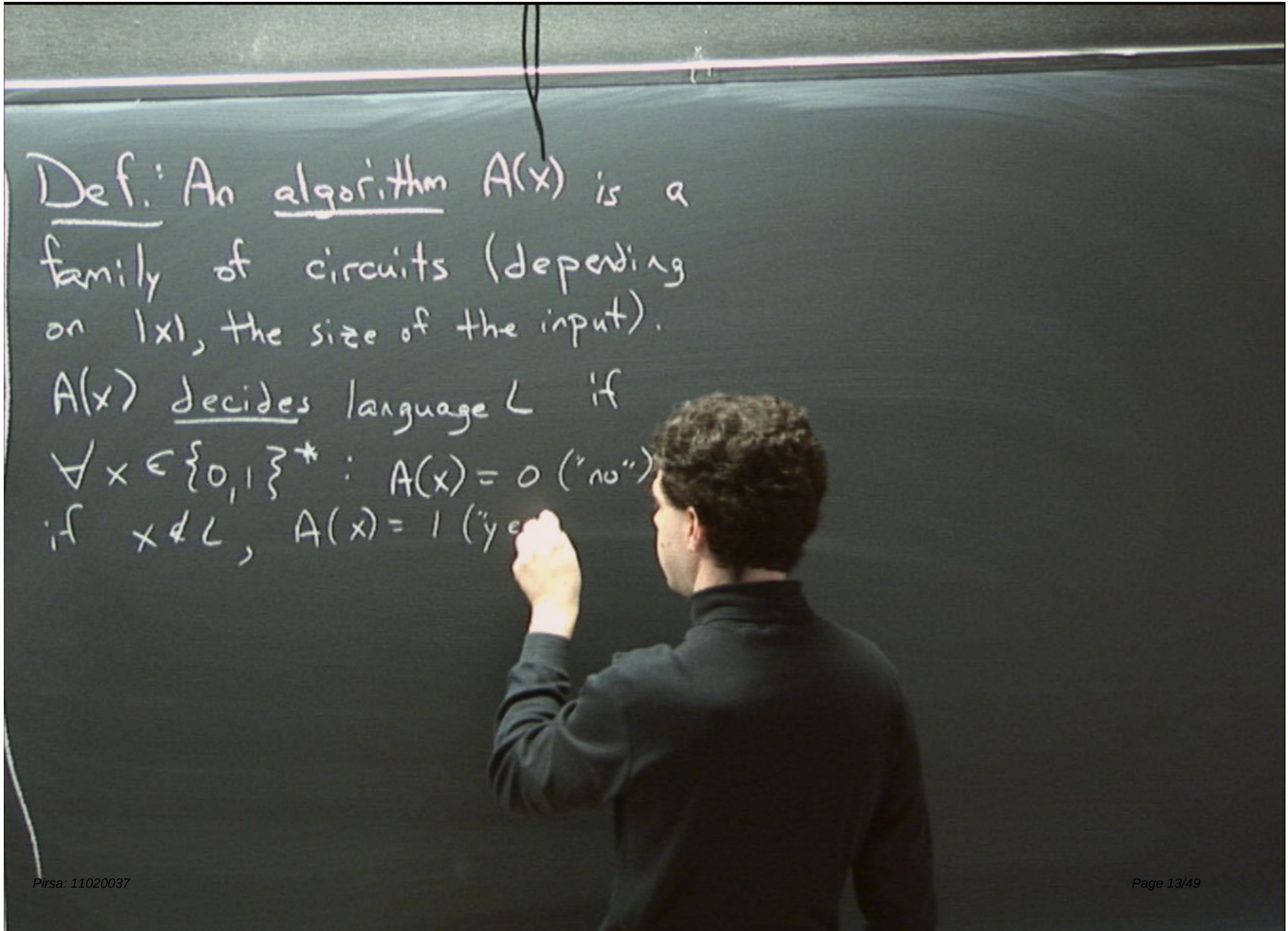
A language is a subset of $\{0,1\}^*$.

$\{0,1\}^*$ is set of possible inputs.

Language L is set of "yes" answers to a yes/no question (decision problem).


Def.: An algorithm A(x)


Def.: An algorithm A(x) is a family of circuits (depending on |x|, the size of the input)


Def.: An algorithm A(x) is a
family of circuits (depending
on |x|, the size of the input).
A(x) decides language L if
∀ x ∈ {0,1}* : A(x) = 0 ("no")
if x ∉ L, A(x) = 1 ("ye

Def.: An *algorithm* $A(x)$ is a family of circuits (depending on $|x|$, the size of the input).

$A(x)$ *decides* language L if

$\forall x \in \{0,1\}^*$: $A(x) = 0$ ("no")

if $x \notin L$, $A(x) = 1$ ("yes") if $x \in L$.

Def.: An *algorithm* $A(x)$ is a family of circuits (depending on $|x|$, the size of the input).

$A(x)$ *decides* language L if

$\forall x \in \{0,1\}^*$: $A(x) = 0$ ("no")

if $x \notin L$, $A(x) = 1$ ("yes") if $x \in L$.

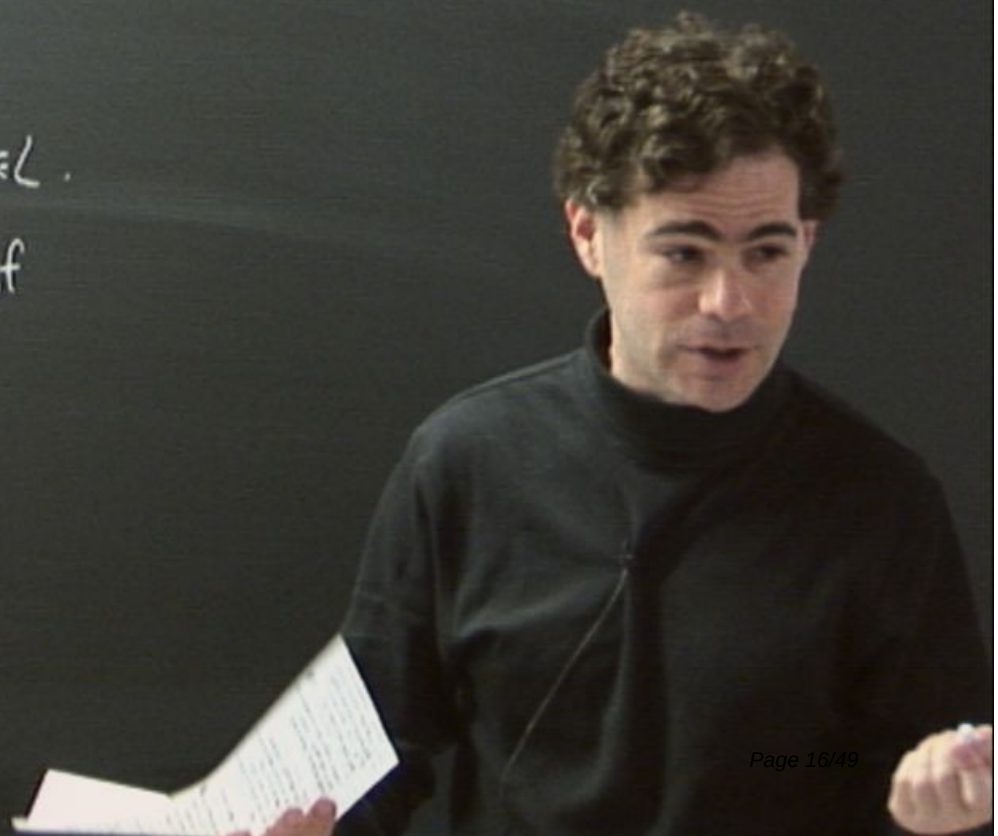
The input x is an *instance* of L.

Def.: An *algorithm* $A(x)$ is a family of circuits (depending on $|x|$, the size of the input).

$A(x)$ *decides* language L if

$\forall x \in \{0,1\}^*$: $A(x) = 0$ ("no") if $x \notin L$, $A(x) = 1$ ("yes") if $x \in L$.

The input x is an *instance* of L.



Def.: An *algorithm* $A(x)$ is a family of circuits (depending on $|x|$, the size of the input).

$A(x)$ *decides* language L if

$\forall x \in \{0,1\}^*$: $A(x) = 0$ ("no") if $x \notin L$, $A(x) = 1$ ("yes") if $x \in L$.

The input x is an *instance* of L.

possible inputs.

Language L is set of "yes" answers to a yes/no question (decision problem).

E.g.: Primality testing $L = \{p \in \mathbb{Z}^+ \mid p \text{ prime}\}$

Algorithm to decide should say if p is prime or composite.

if $x \notin L$

The input

L.

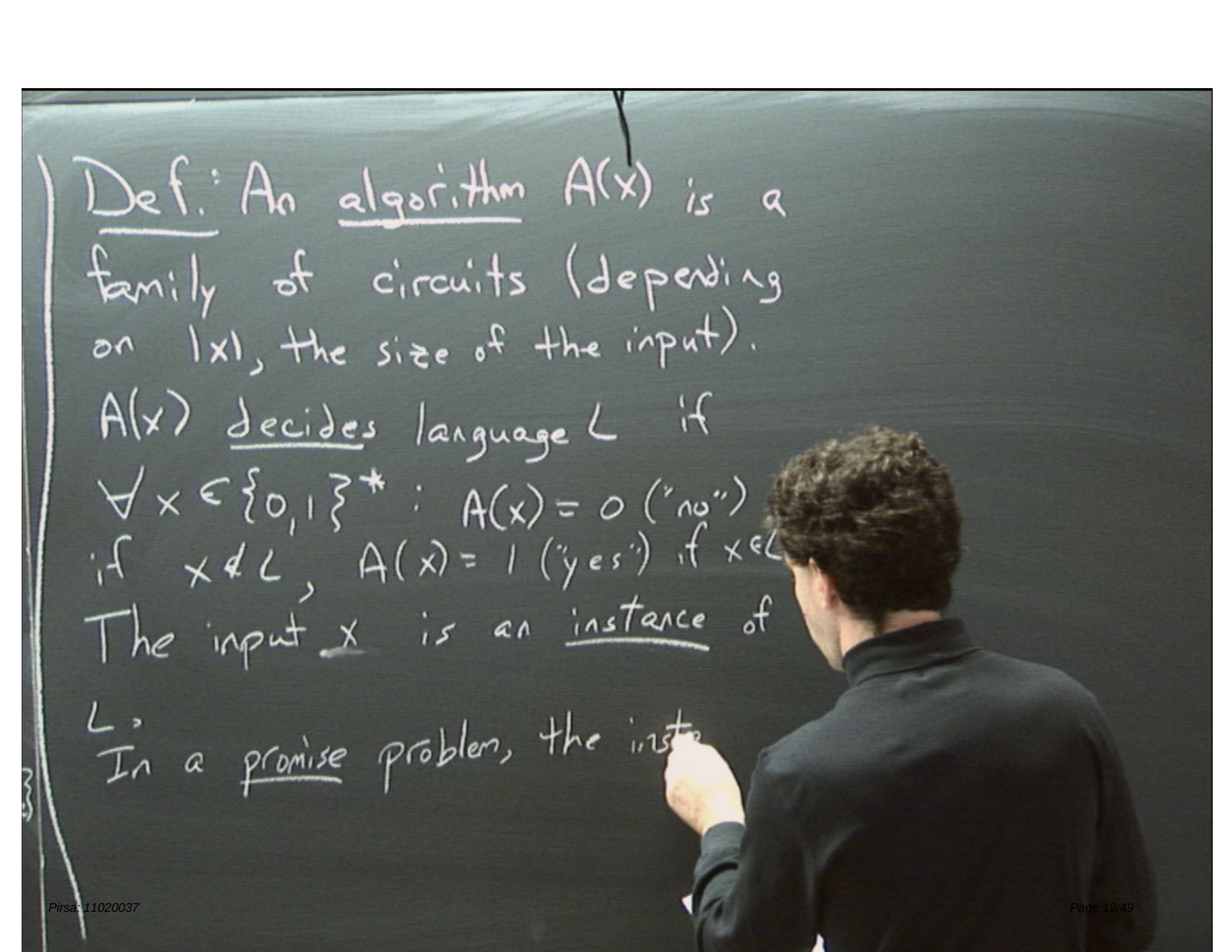
Def: An *algorithm* $A(x)$ is a family of circuits (depending on $|x|$, the size of the input).

$A(x)$ *decides* language L if

$\forall x \in \{0,1\}^*$: $A(x) = 0$ ("no") if $x \notin L$, $A(x) = 1$ ("yes") if $x \in L$

The input x is an *instance* of L.

In a *promise* problem, the inst



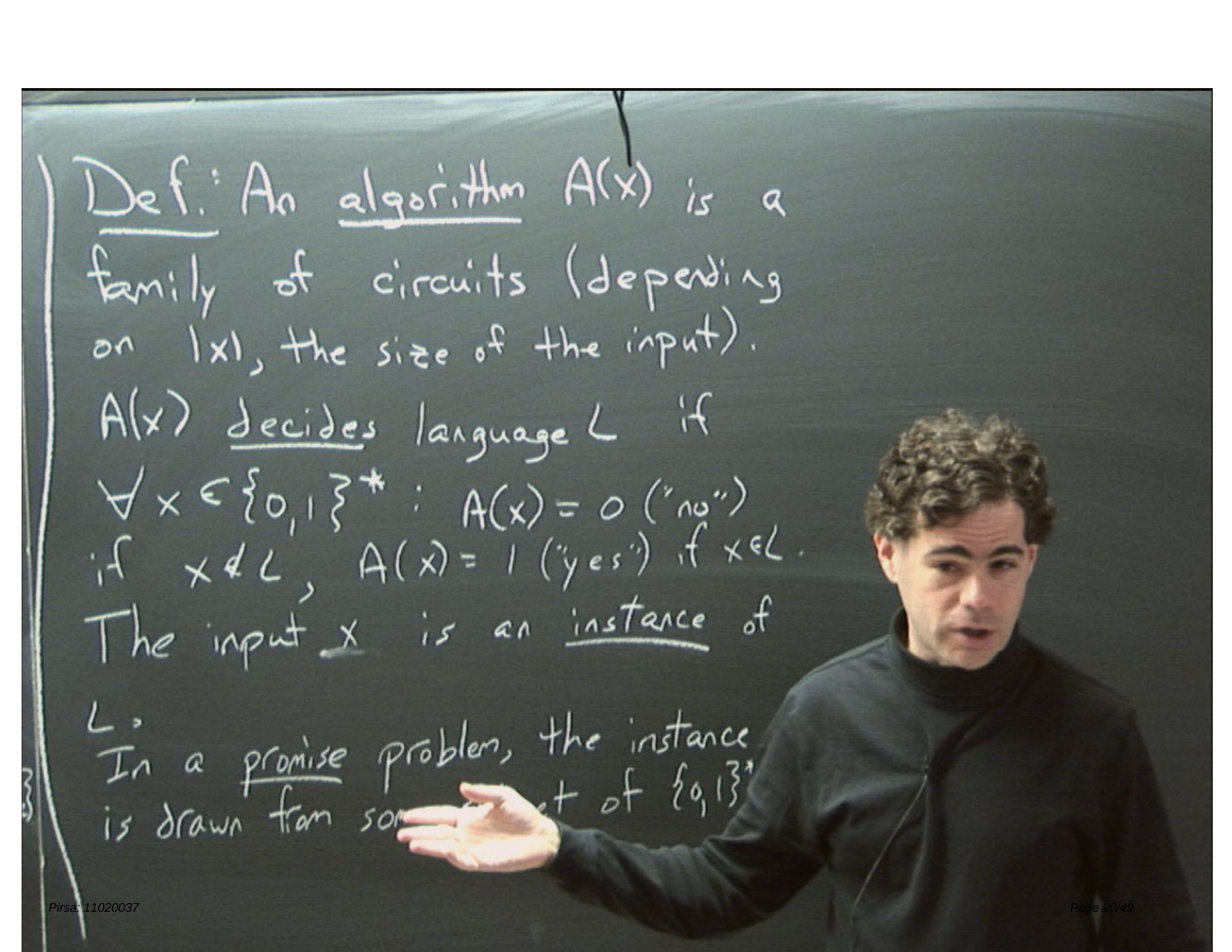
Def: An *algorithm* $A(x)$ is a family of circuits (depending on $|x|$, the size of the input).

$A(x)$ *decides* language L if

$\forall x \in \{0,1\}^*$: $A(x) = 0$ ("no") if $x \notin L$, $A(x) = 1$ ("yes") if $x \in L$.

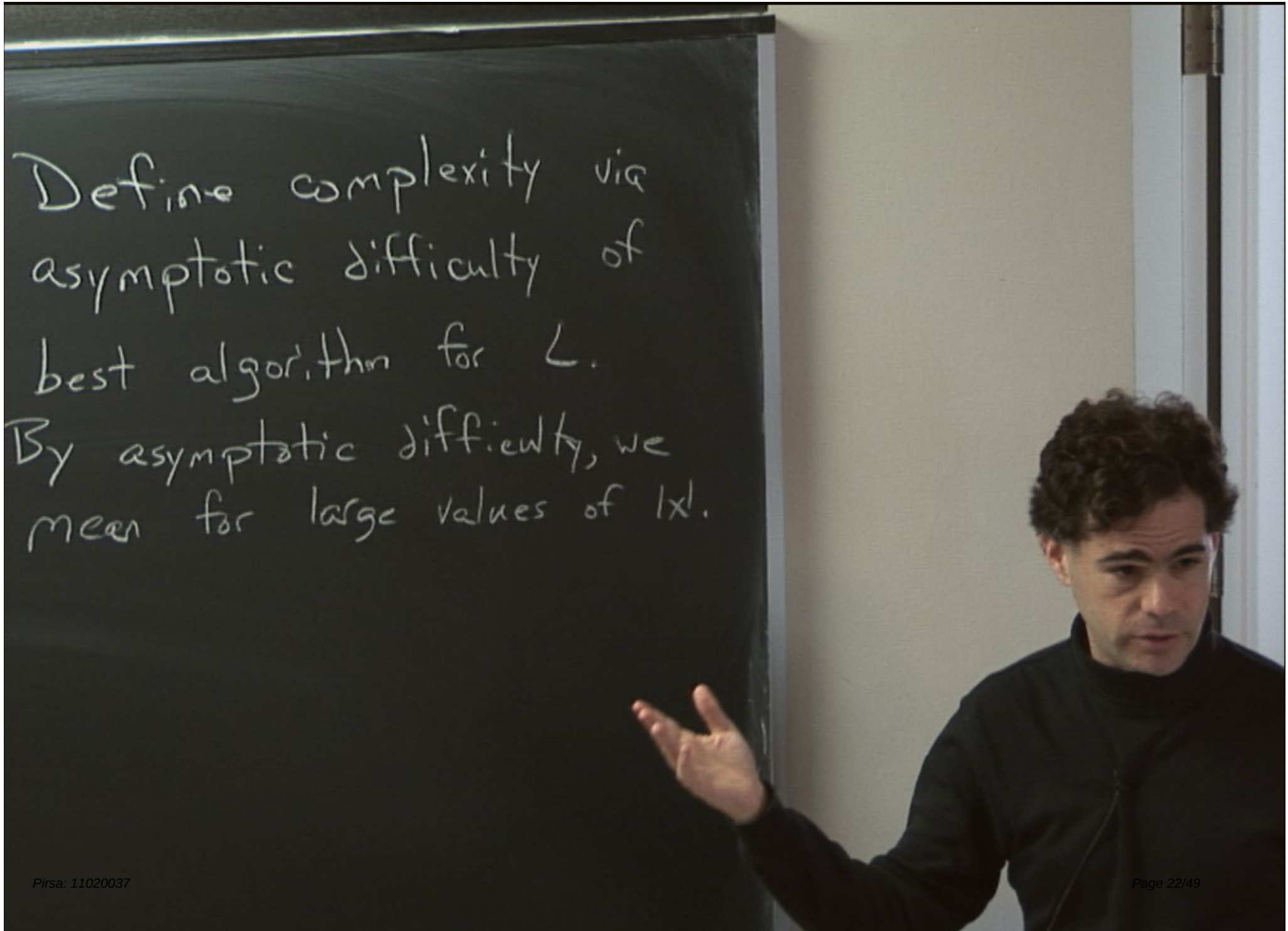
The input x is an *instance* of L.

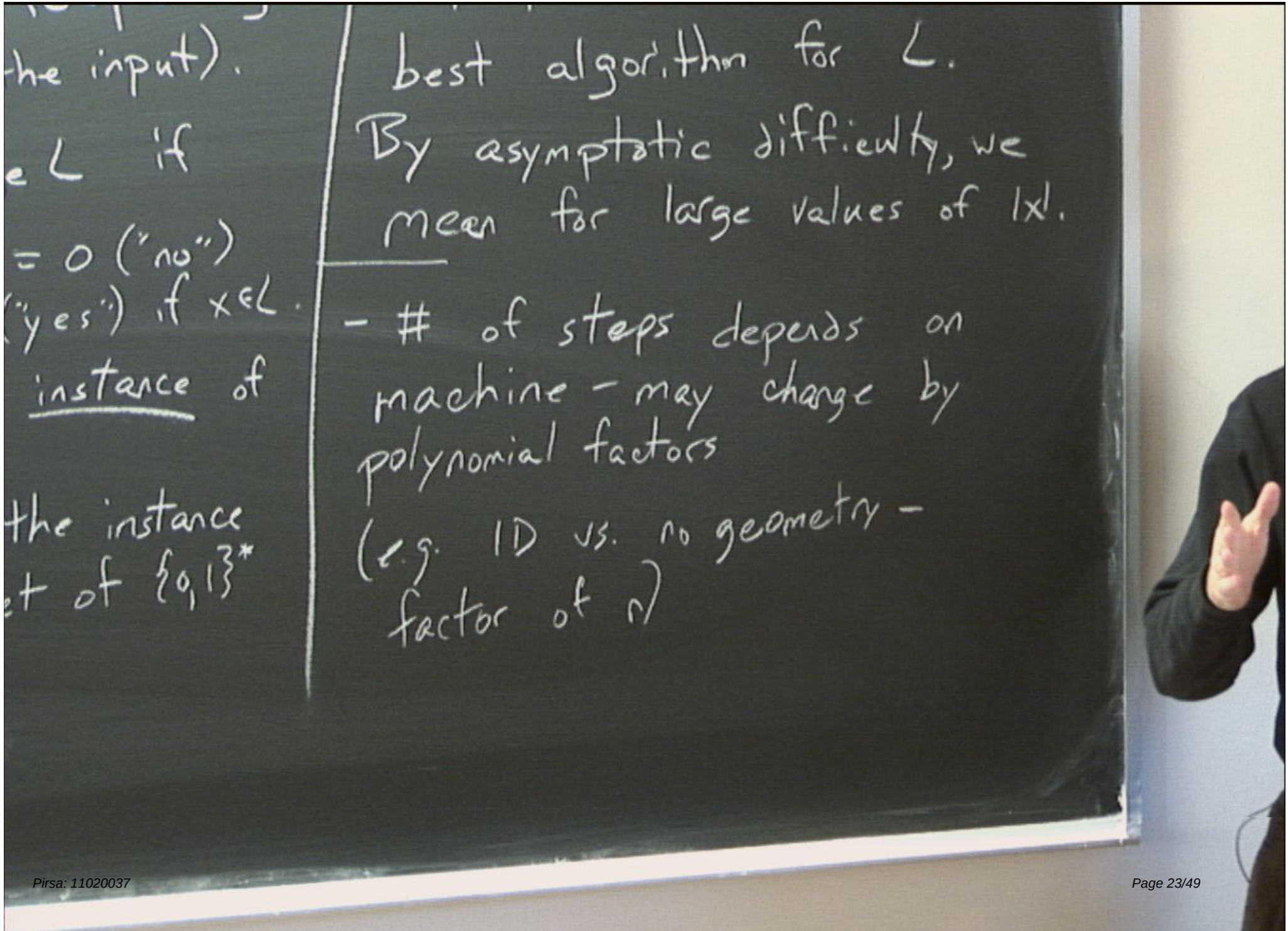
In a *promise* problem, the instance is drawn from some subset of $\{0,1\}^*$

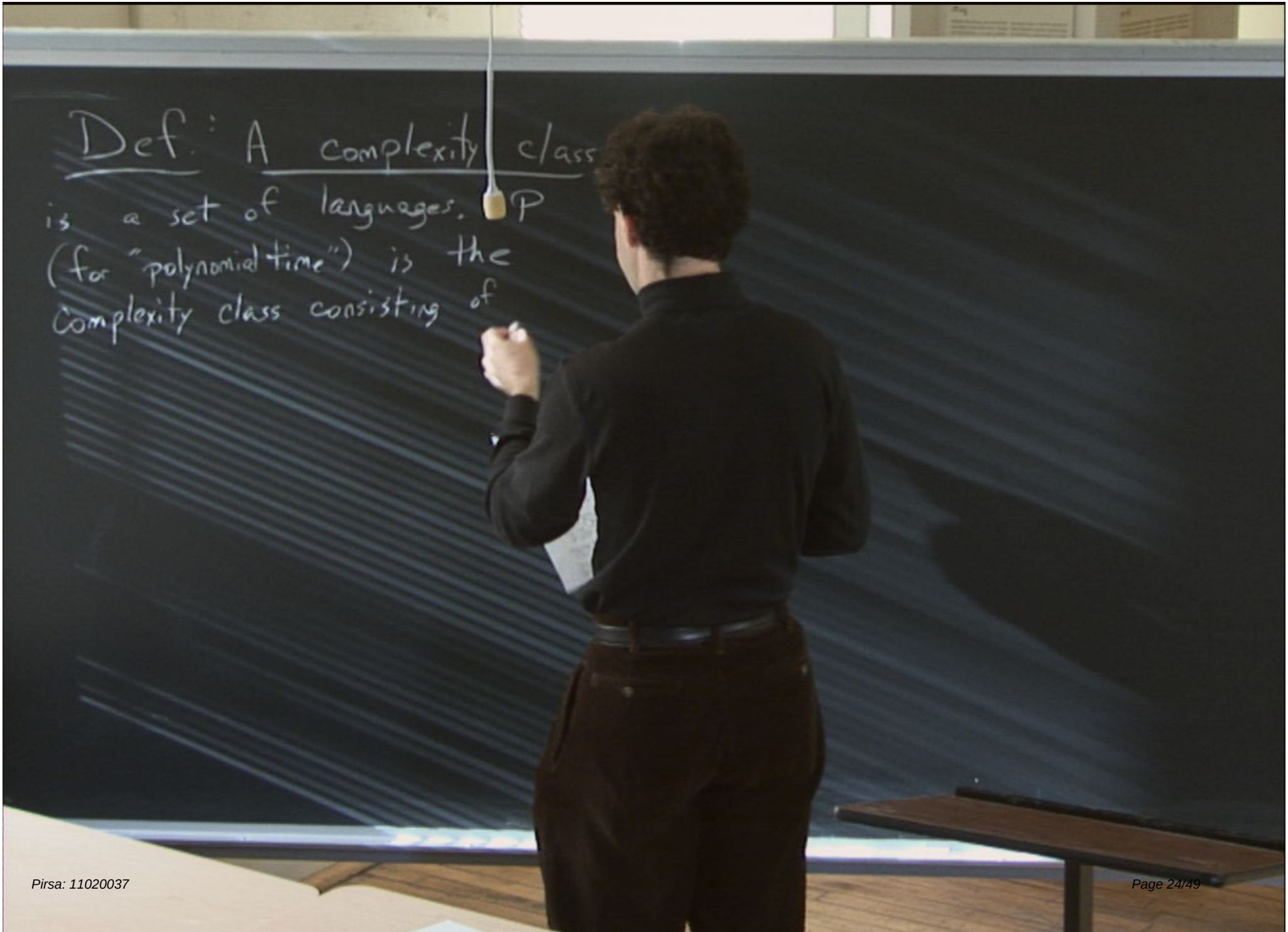


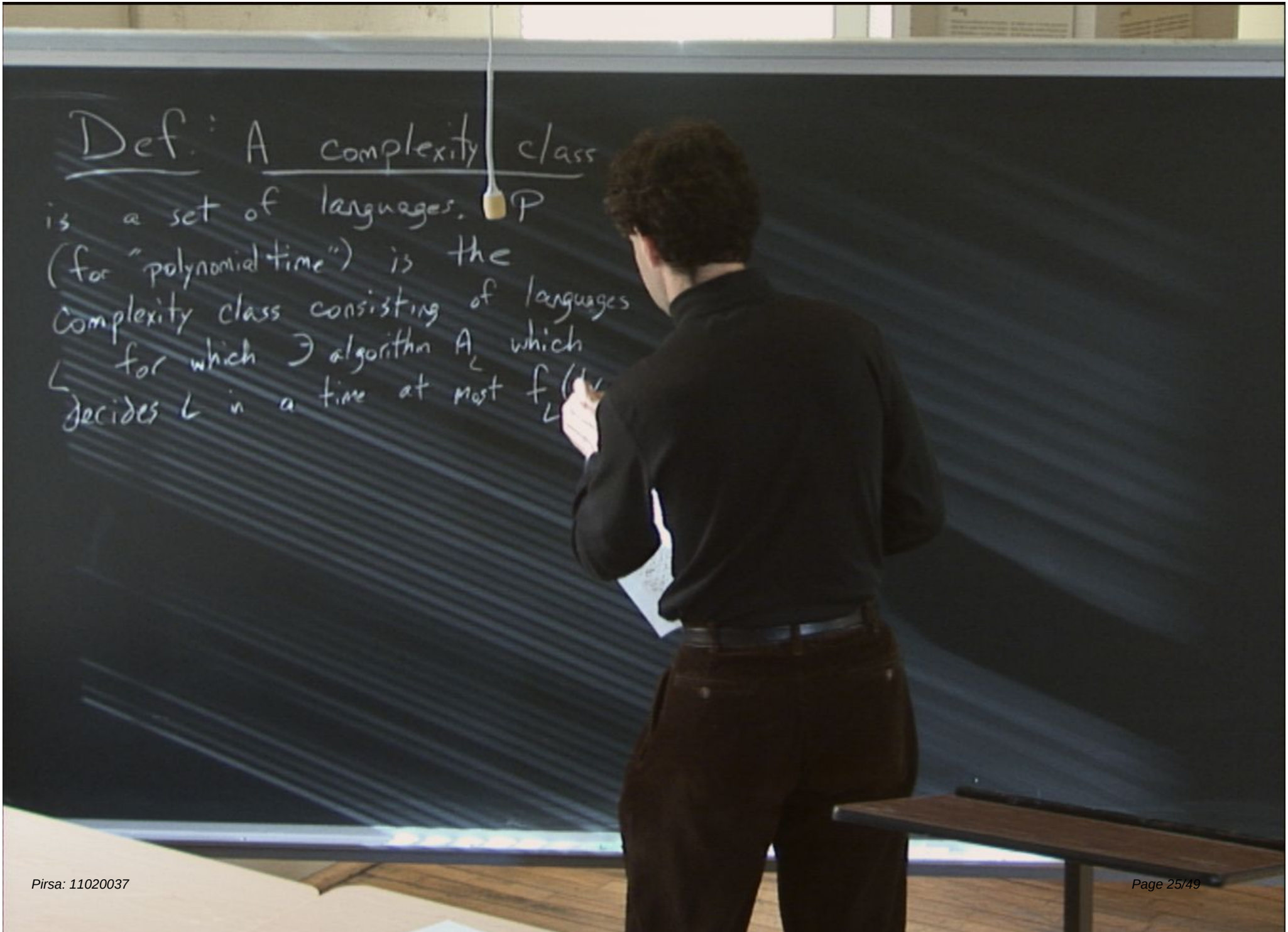
thm A(x) is a
its (depending
of the input).
nguage L if
A(x) = 0 ("no")
= 1 ("yes") if x∈L.
an instance of
lem, the instance
subset of {0,1}*

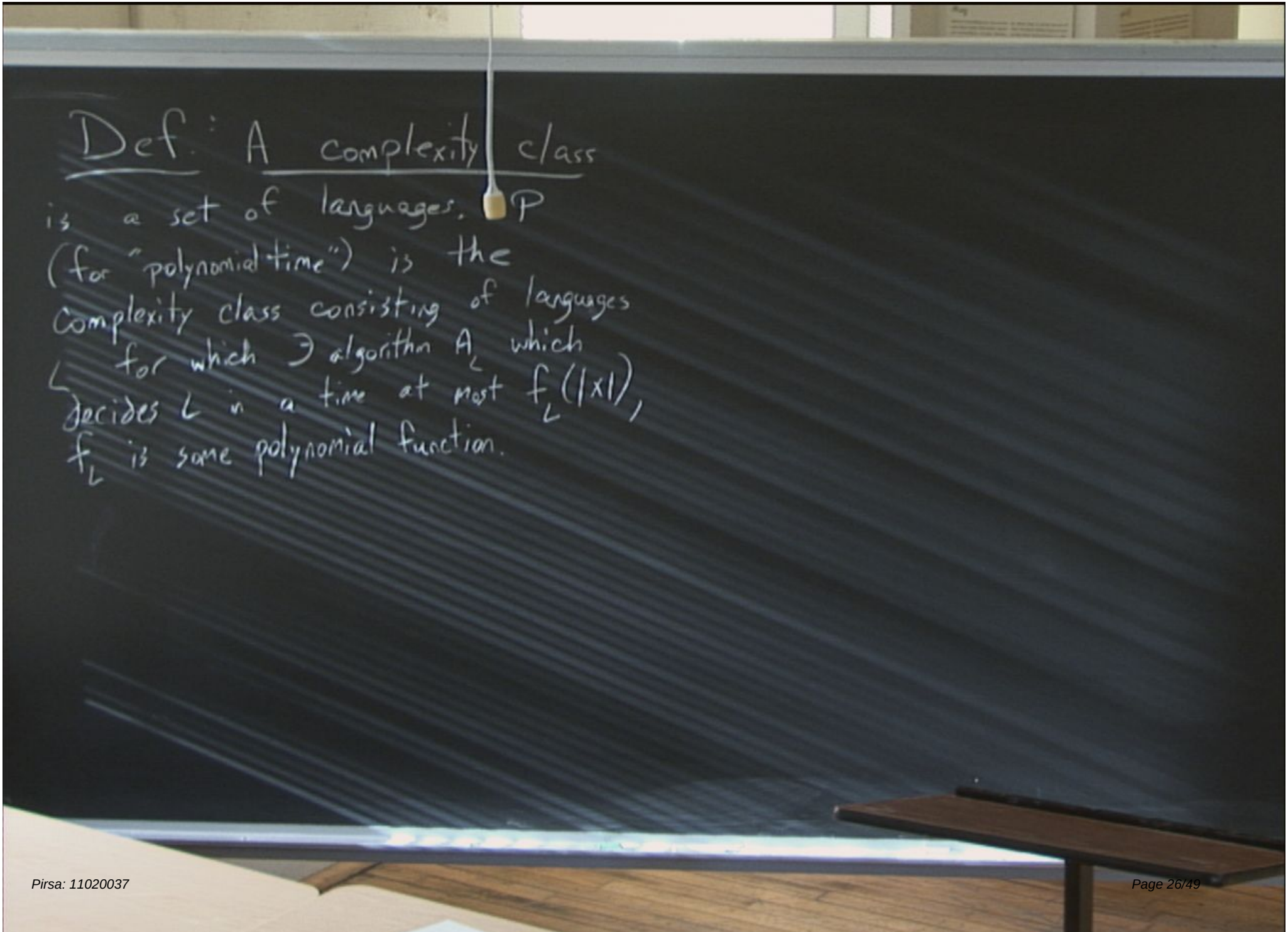
Define complexity via asymptotic difficulty of best algorithm for L.

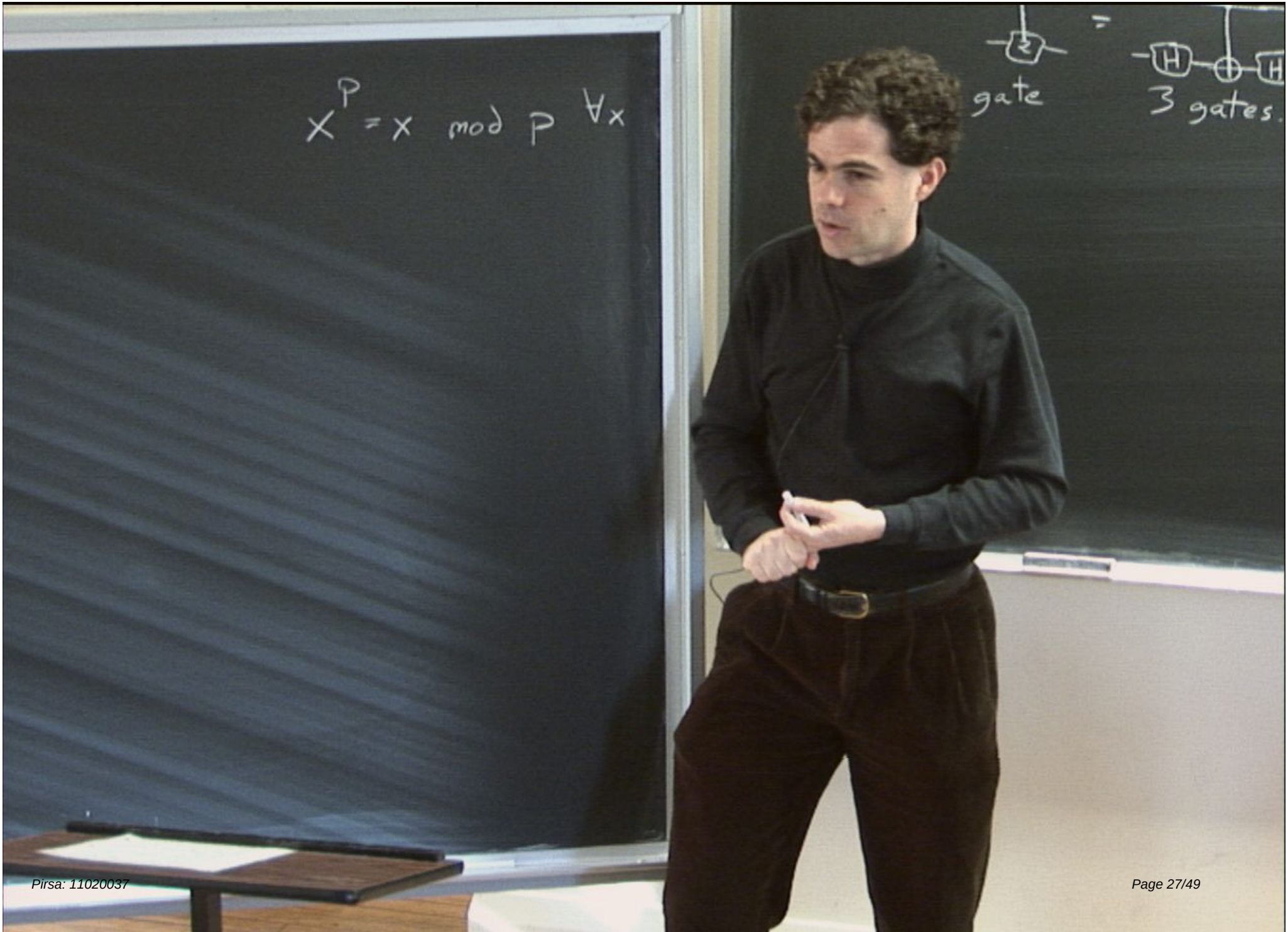

Define complexity via
asymptotic difficulty of
best algorithm for L.
By asymptotic difficulty, we
mean for large values of |x|.


the input).
e L if
= 0 ("no")
("yes") if x∈L.
instance of
the instance
et of {0,1}*
best algorithm for L.
By asymptotic difficulty, we
mean for large values of |x|.
- # of steps depends on
machine - may change by
polynomial factors
(e.g. 1D vs. no geometry -
factor of n)


Def: A complexity class
is a set of languages. P
(for "polynomial time") is the
complexity class consisting of


Def: A complexity class
is a set of languages. P
(for "polynomial time") is the
complexity class consisting of languages
for which ∃ algorithm A_L which
decides L in a time at most f_L(


Def: A complexity class
is a set of languages. P
(for "polynomial time") is the
complexity class consisting of languages
L for which ∃ algorithm A_L which
decides L in a time at most $f_L(|x|)$,
f_L is some polynomial function.


$x^p = x \mod p \quad \forall x$
gate
3 gates.

on $|x|$, the size of the input).

$A(x)$ decides language L if

$\forall x \in \{0,1\}^*$: $A(x) = 0$ ("no") if $x \notin L$, $A(x) = 1$ ("yes") if $x \in L$.

The input x is an instance of L.

In a promise problem, the instance is drawn from some subset of $\{0,1\}^*$

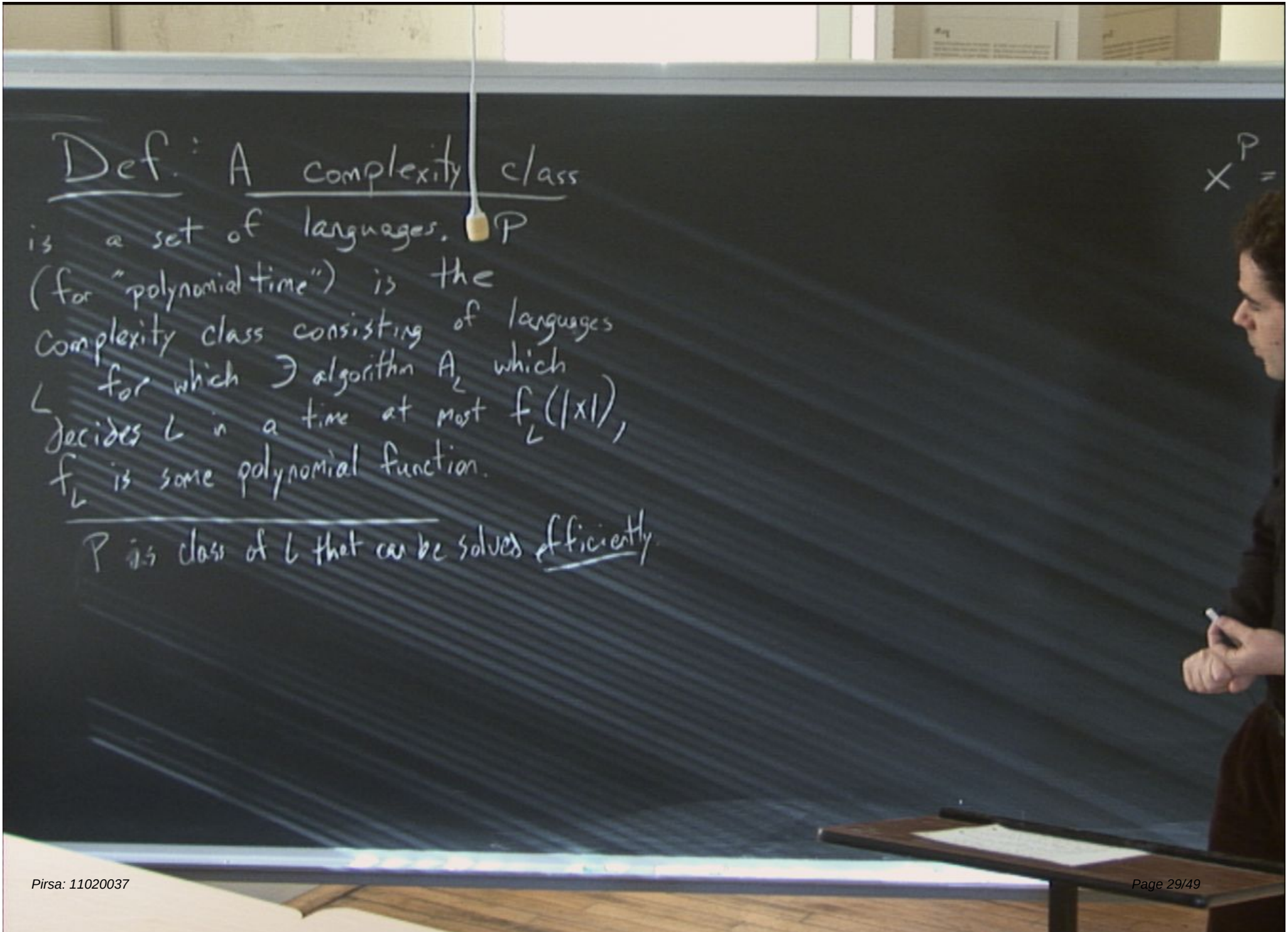
$A(x)$ must work even for worst case.

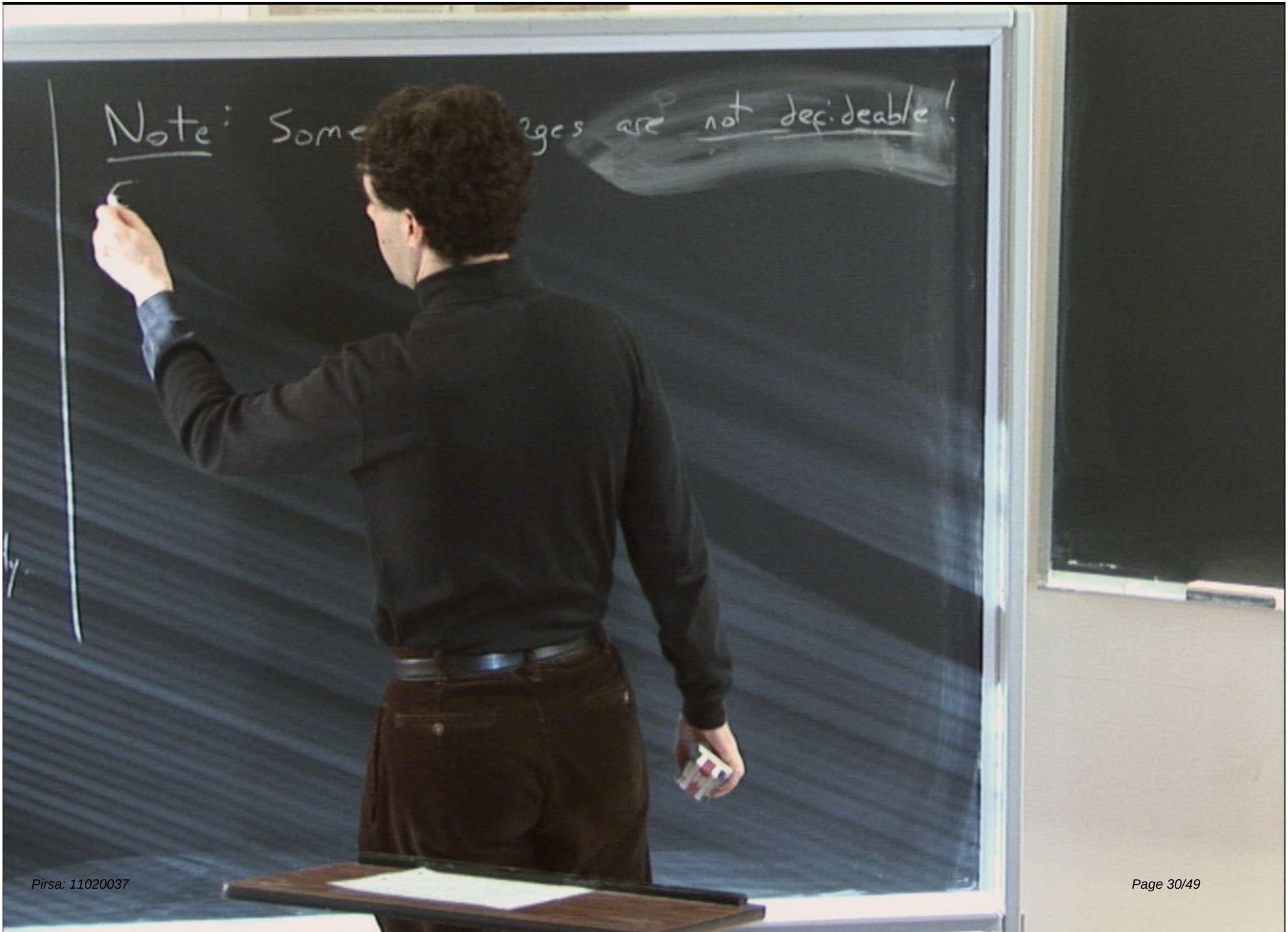
best algorithm for L.

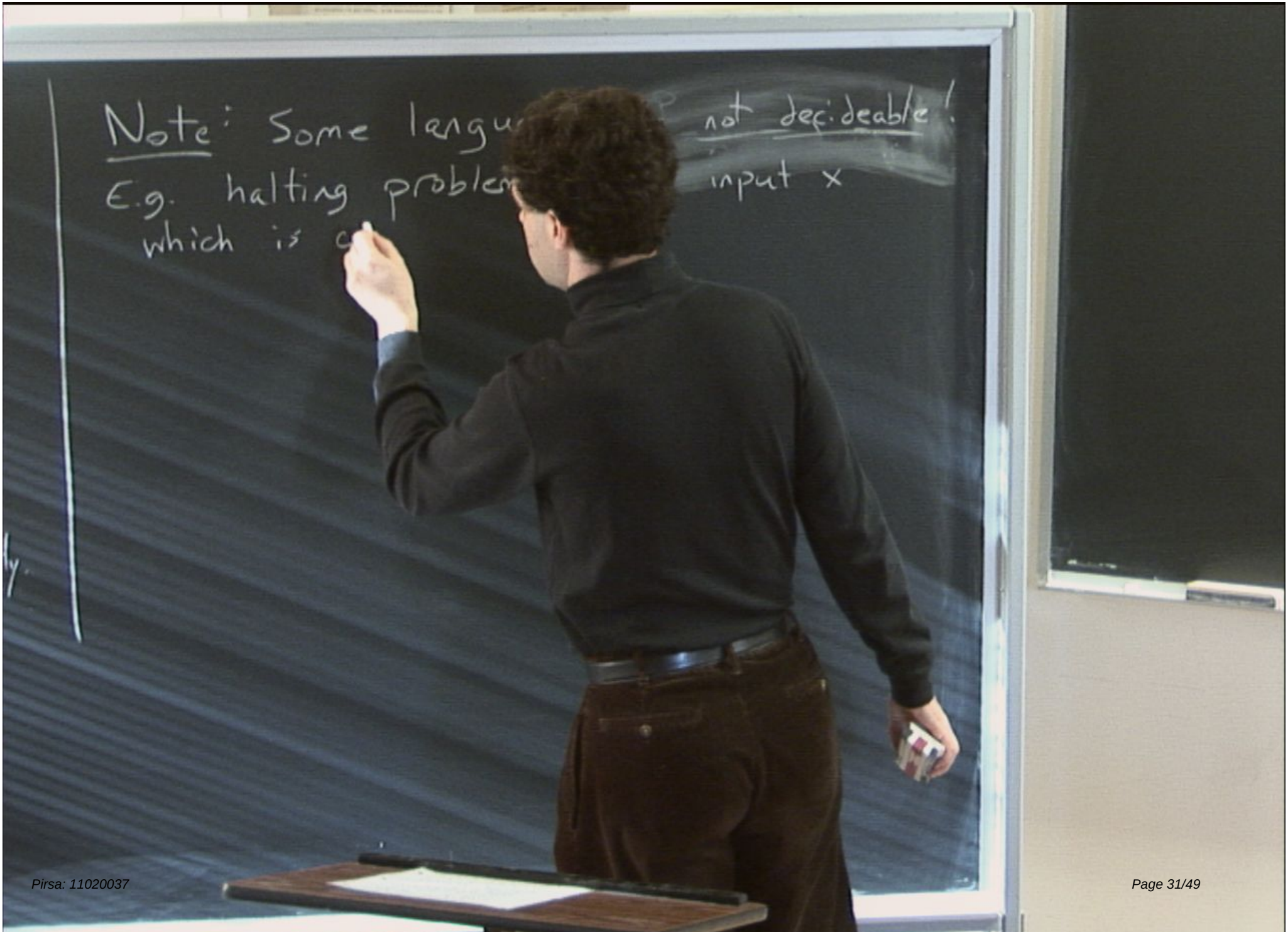
By asymptotic difficulty, we mean for large values of $|x|$.

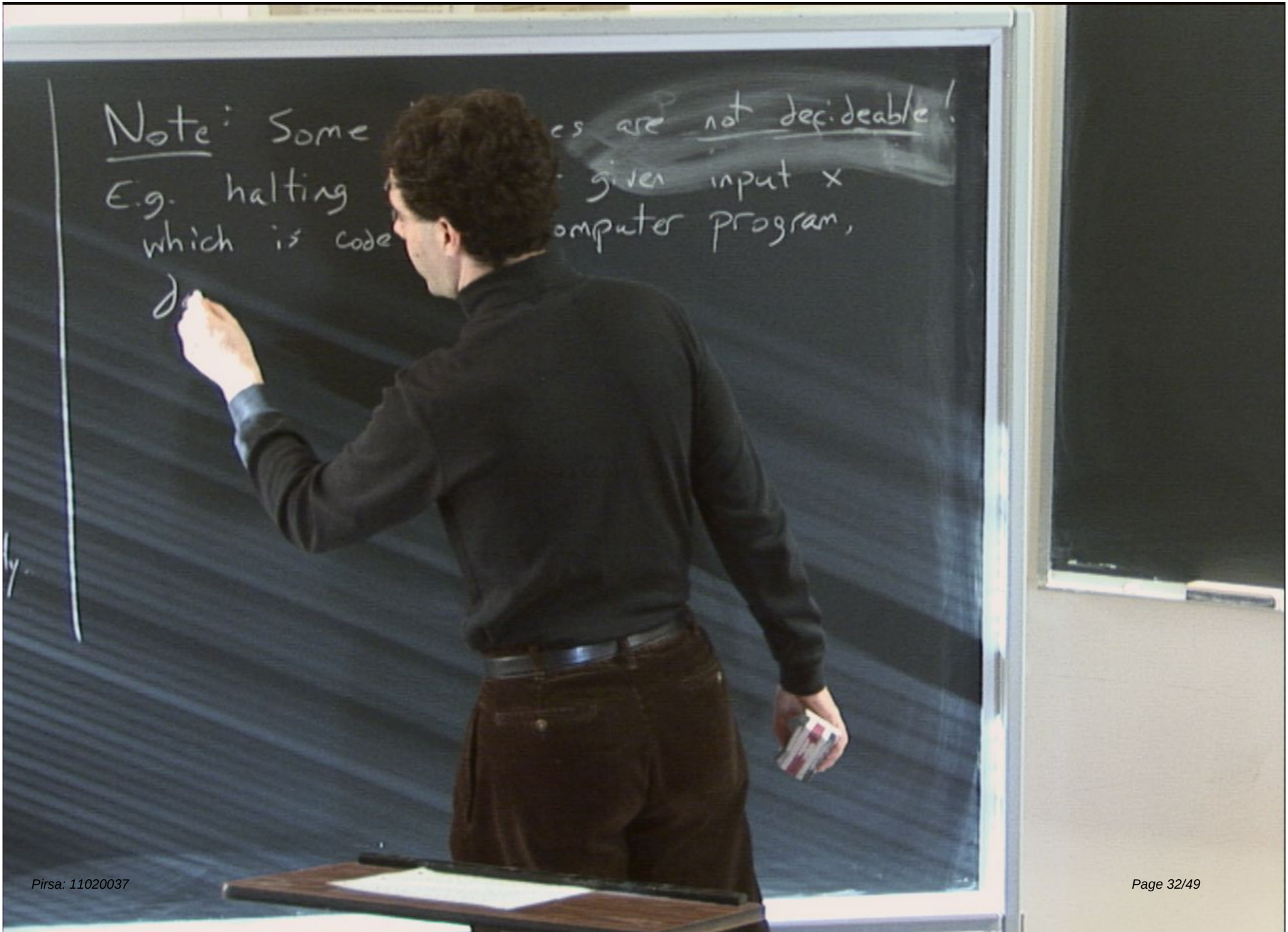
- # of steps depends on machine - may change by polynomial factors

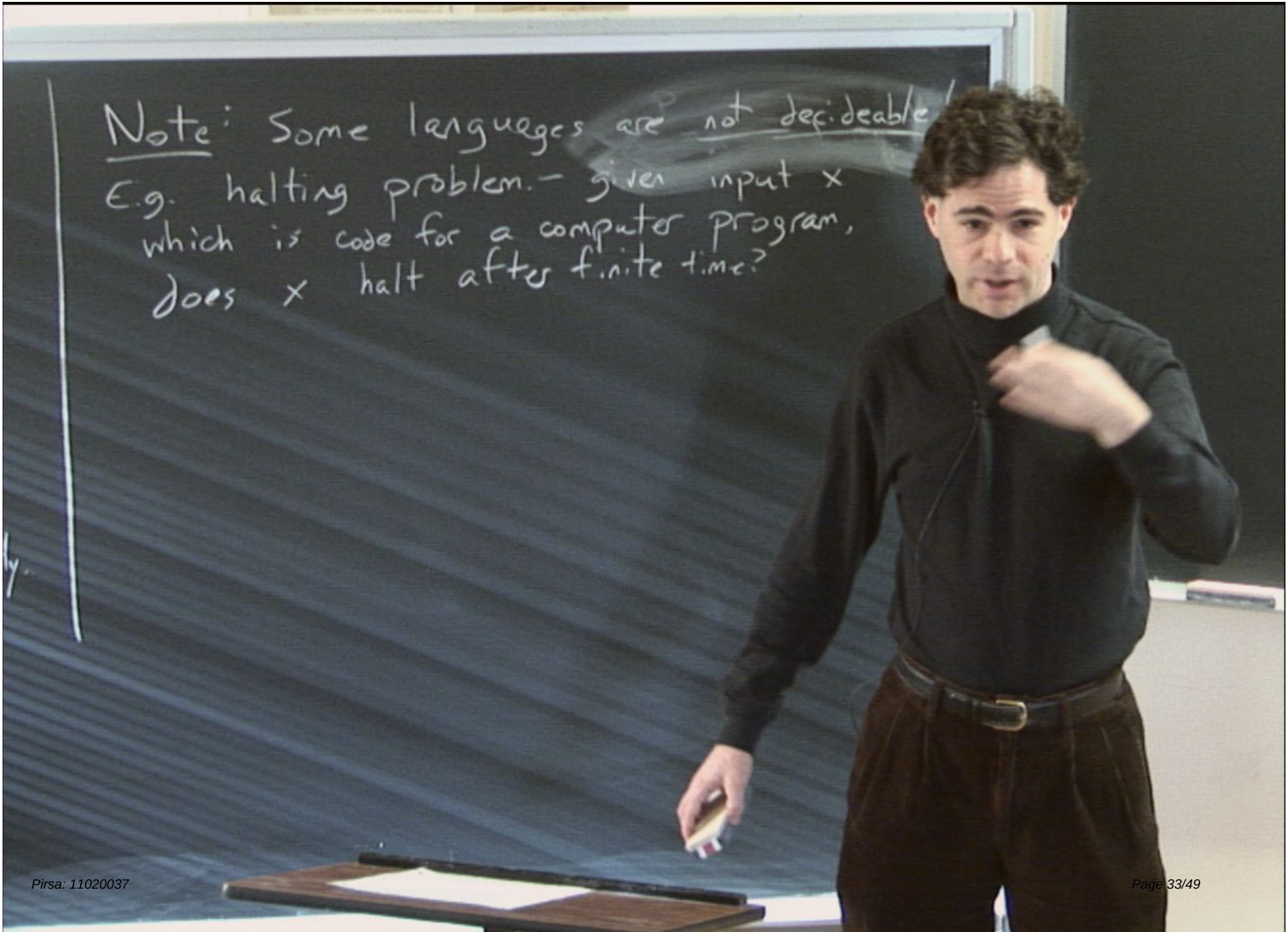
(e.g. 1D vs. no geometry - factor of n)

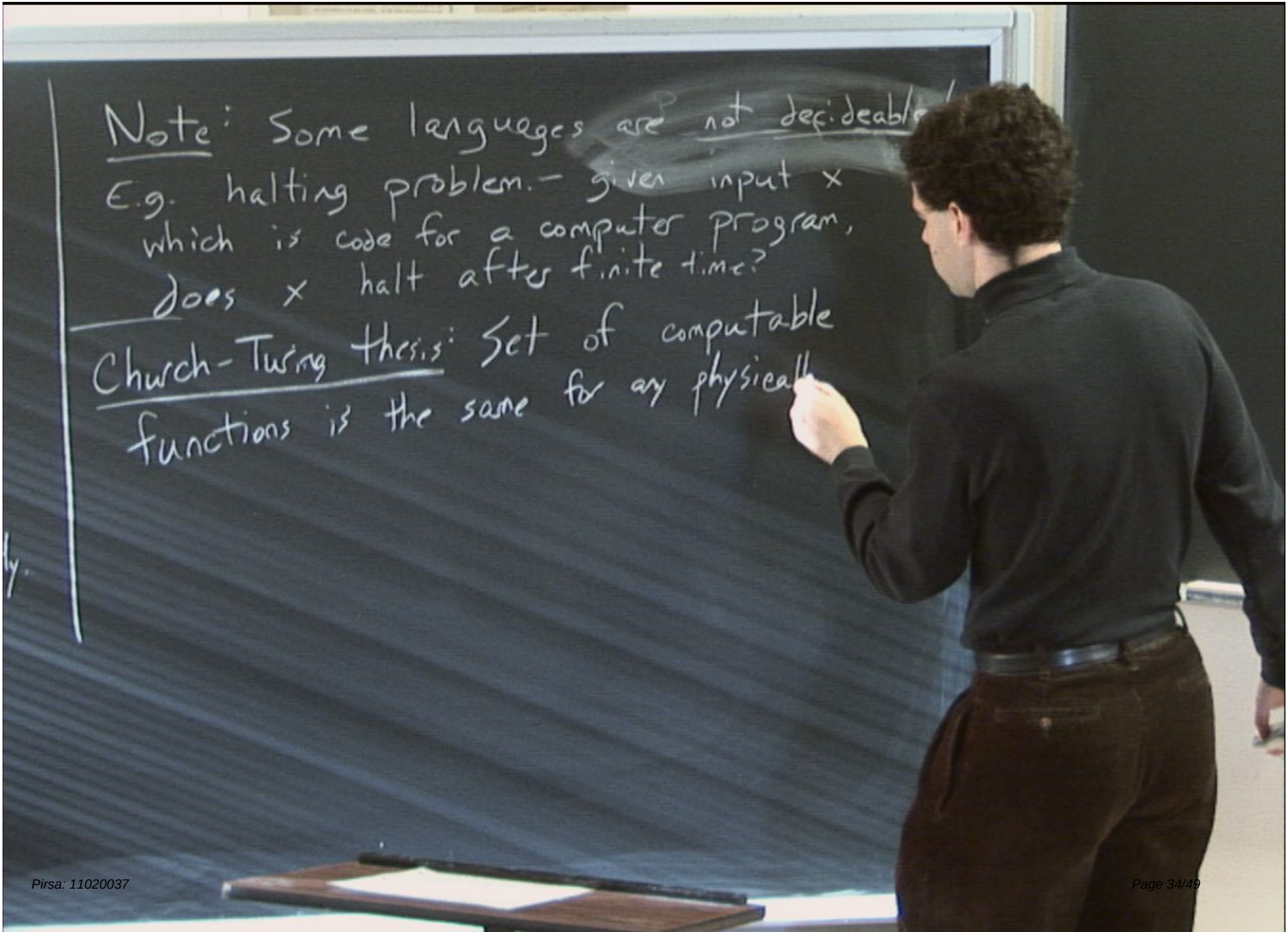

Def: A complexity class
is a set of languages. P
(for "polynomial time") is the
complexity class consisting of languages
L for which ∃ algorithm A_L which
decides L in a time at most $f_L(|x|)$,
f_L is some polynomial function.
P is class of L that can be solved efficiently.
$X^P =$

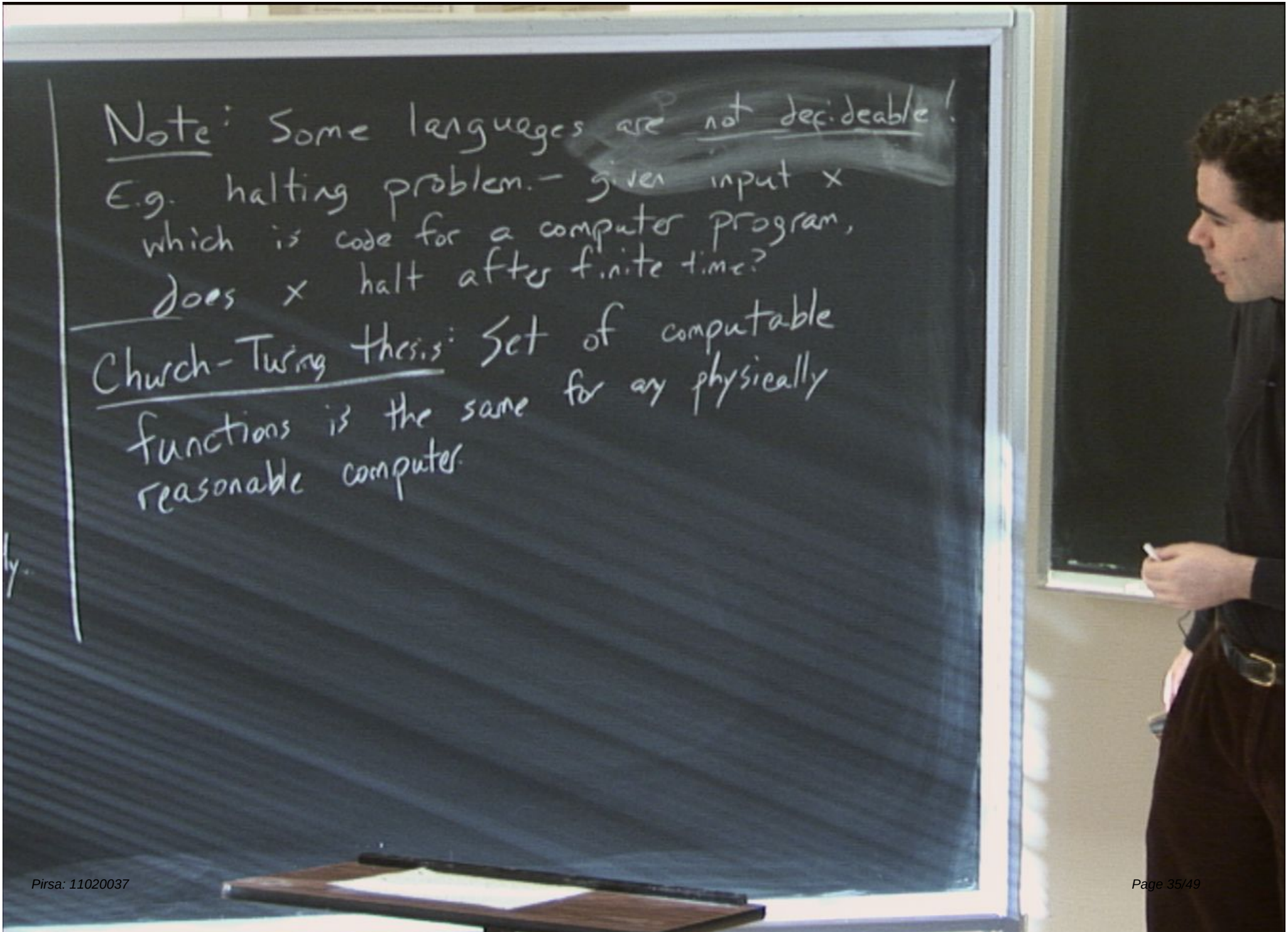

Note: Some
ges are not dec. deable!

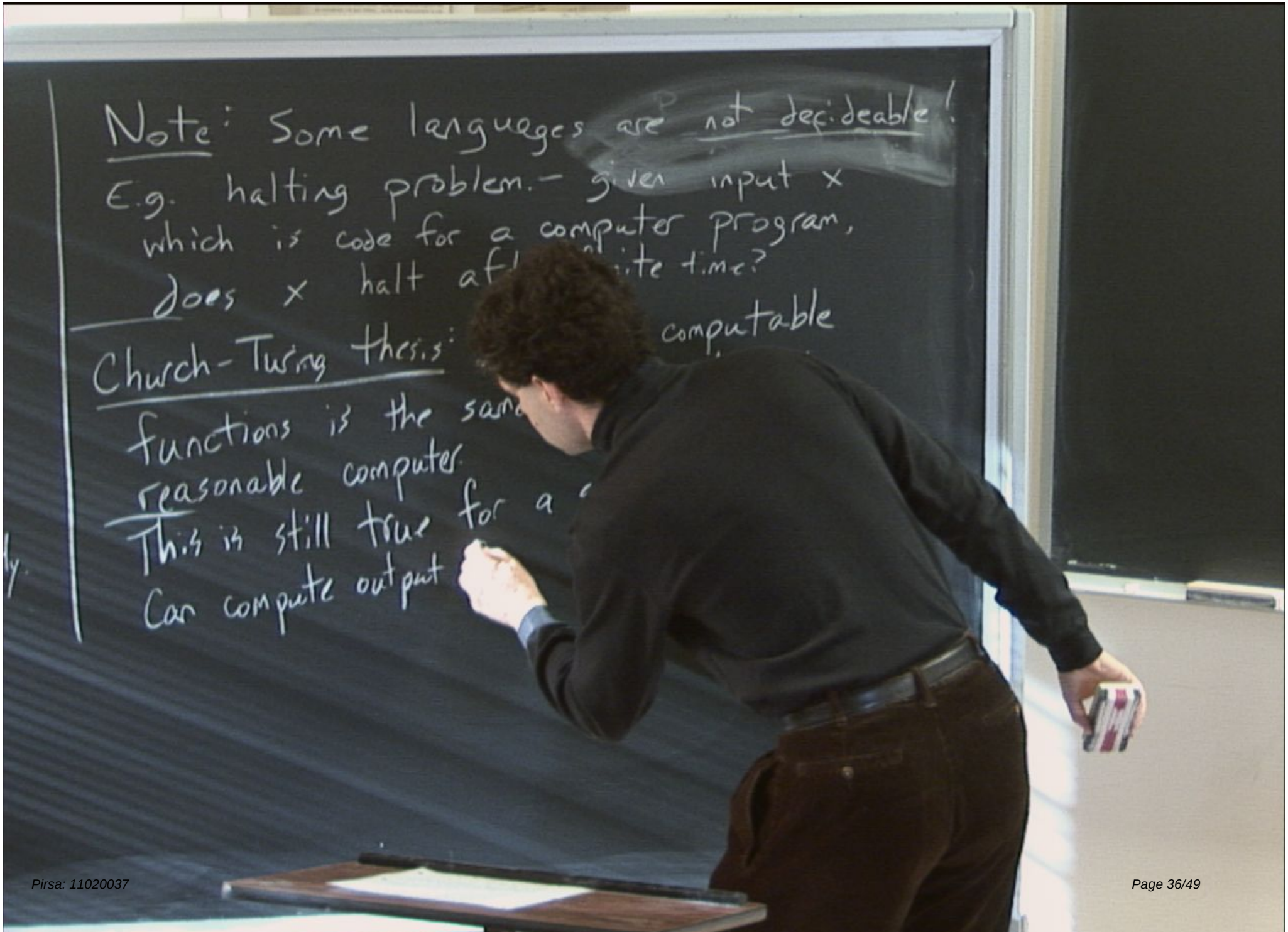

Note: Some langu not decidable!
E.g. halting proble input x
which is

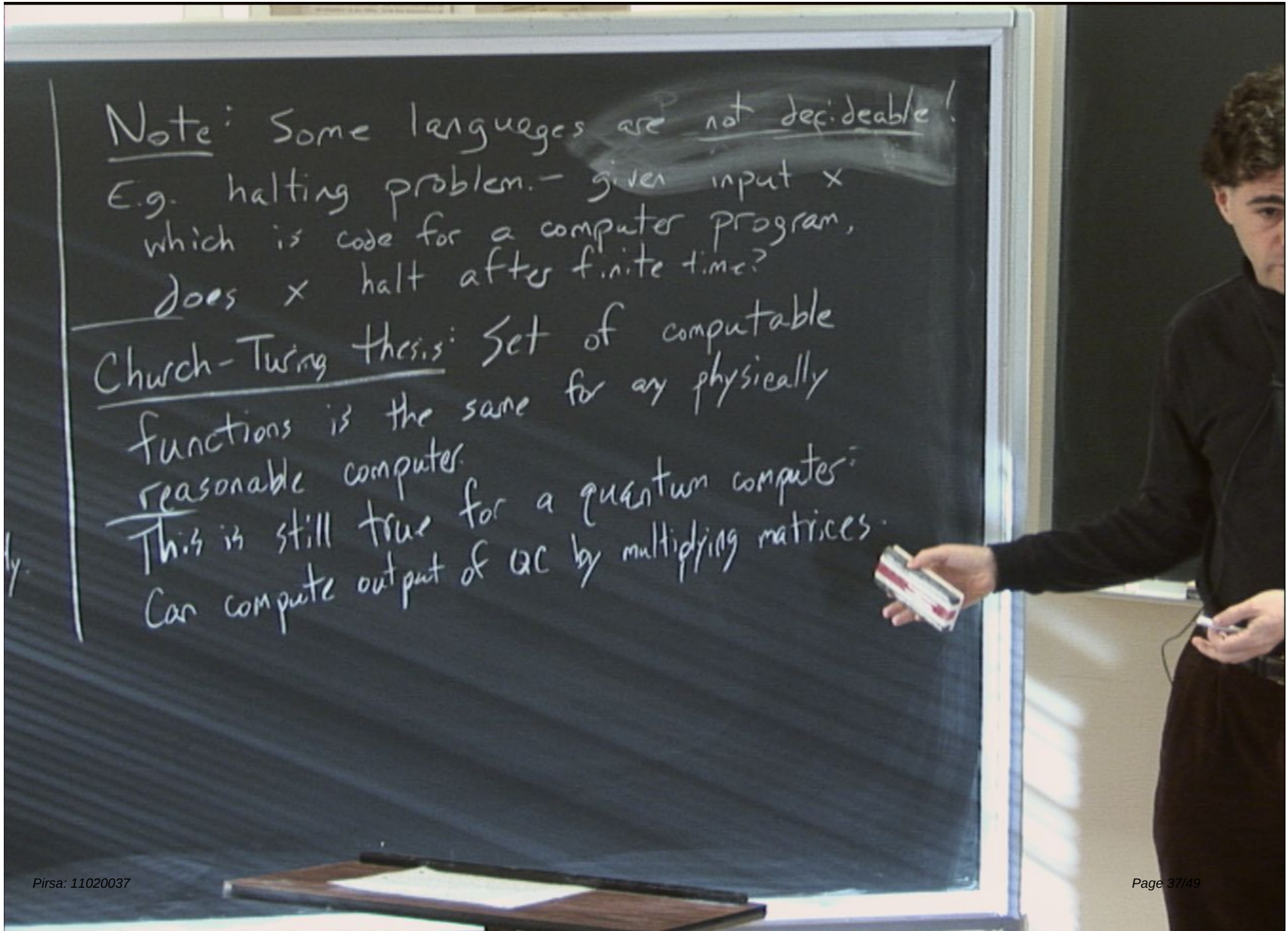

Note: Some
es are not decideable!
E.g. halting
given input x
which is code
omputer program,

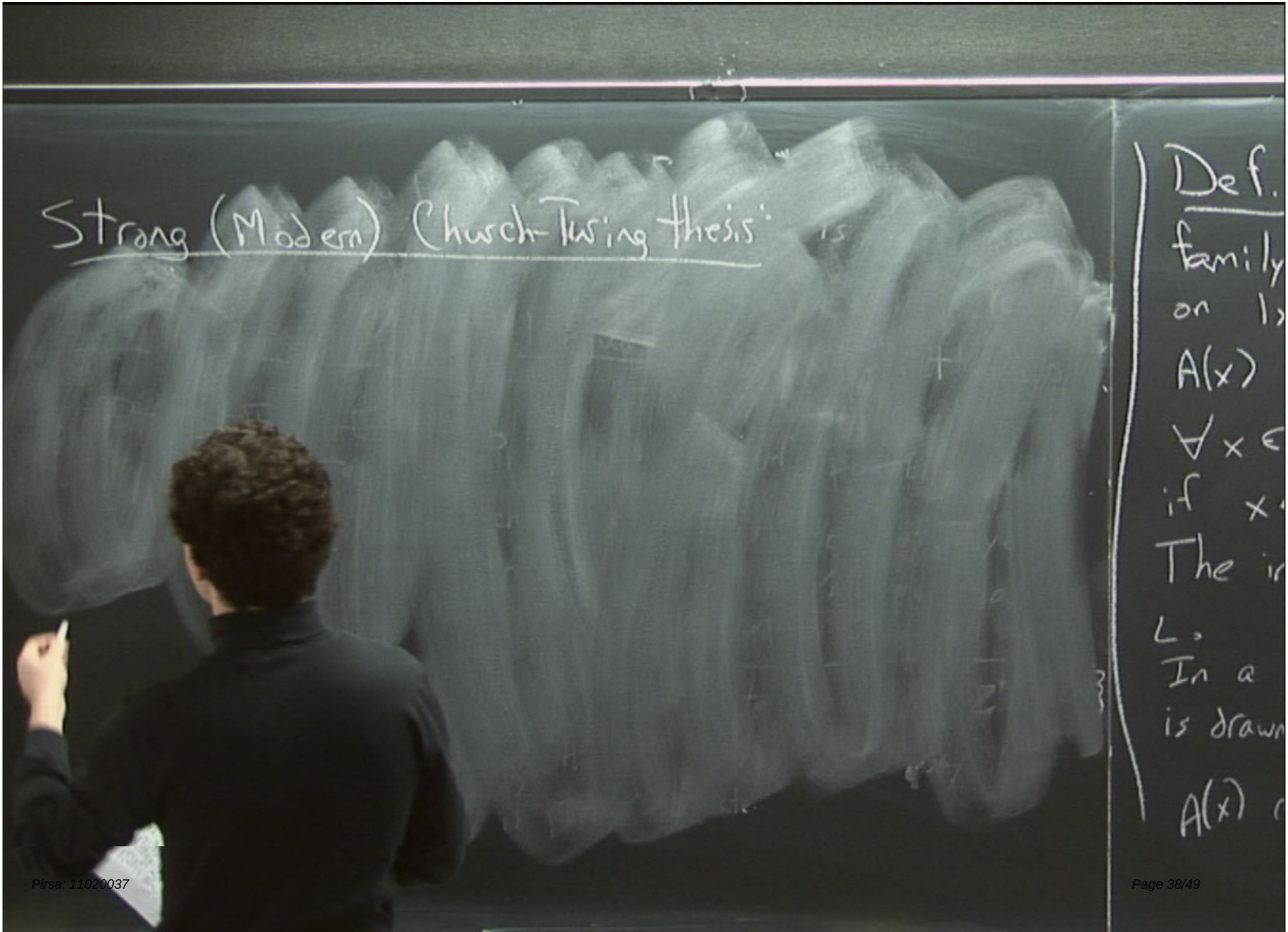

Note: Some languages are not decidable!
E.g. halting problem. – given input x
which is code for a computer program,
does x halt after finite time?

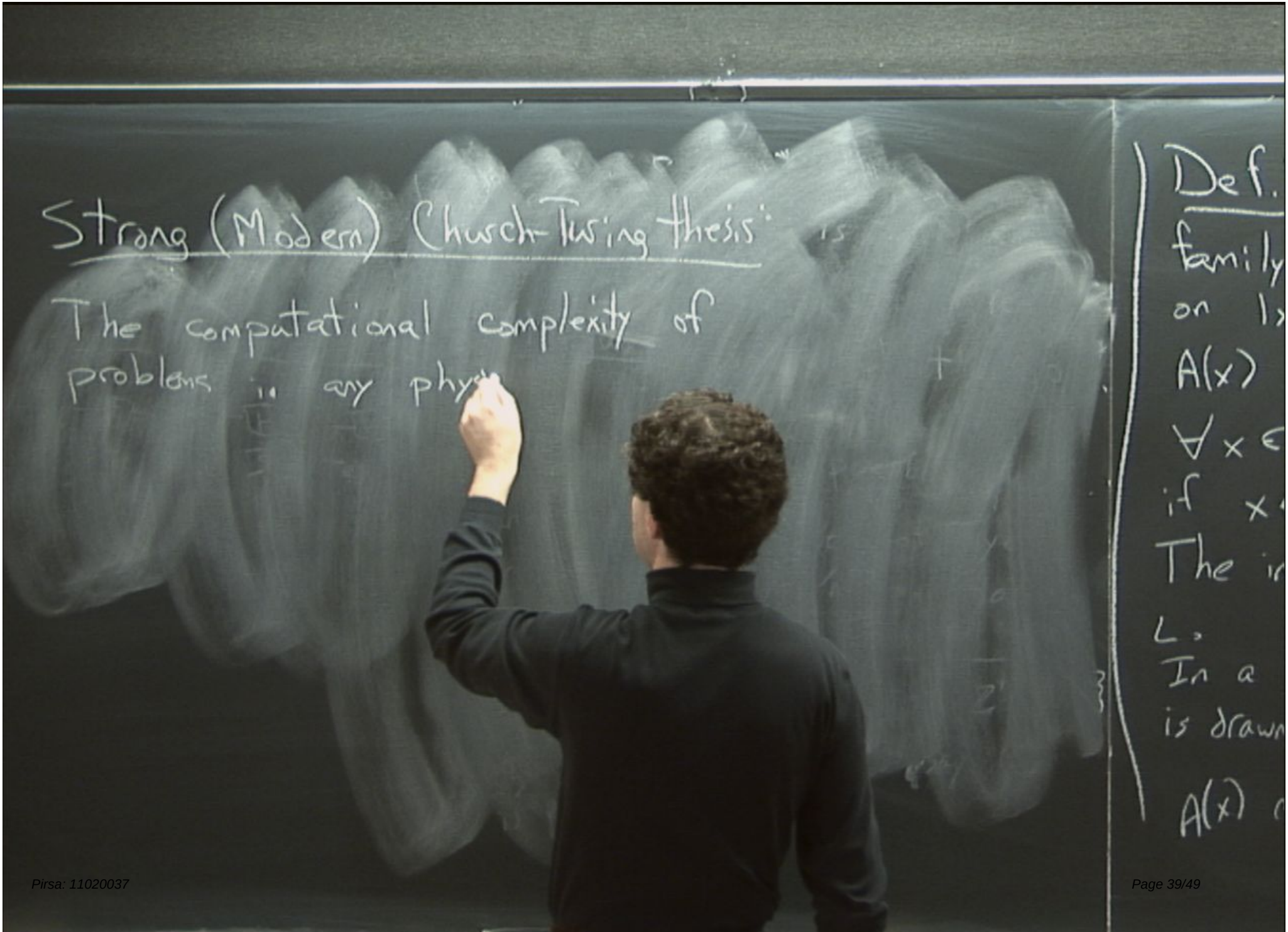

Note: Some languages are not decidable!
E.g. halting problem. – given input x
which is code for a computer program,
does x halt after finite time?
Church-Turing thesis: Set of computable
functions is the same for any physicall

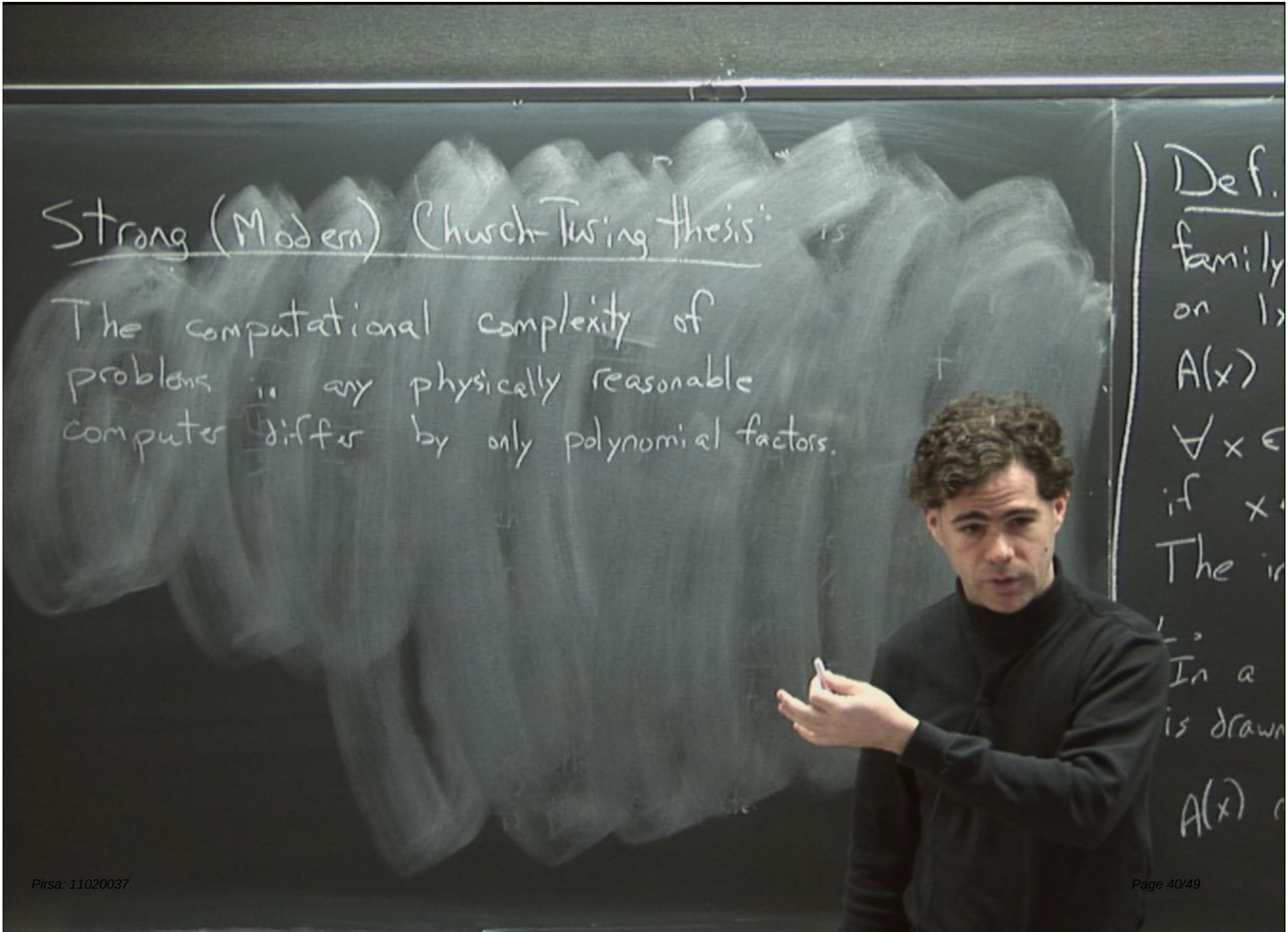

Note: Some languages are not decideable!
E.g. halting problem. - given input x
which is code for a computer program,
does x halt after finite time?
Church-Turing thesis: Set of computable
functions is the same for any physically
reasonable computer.

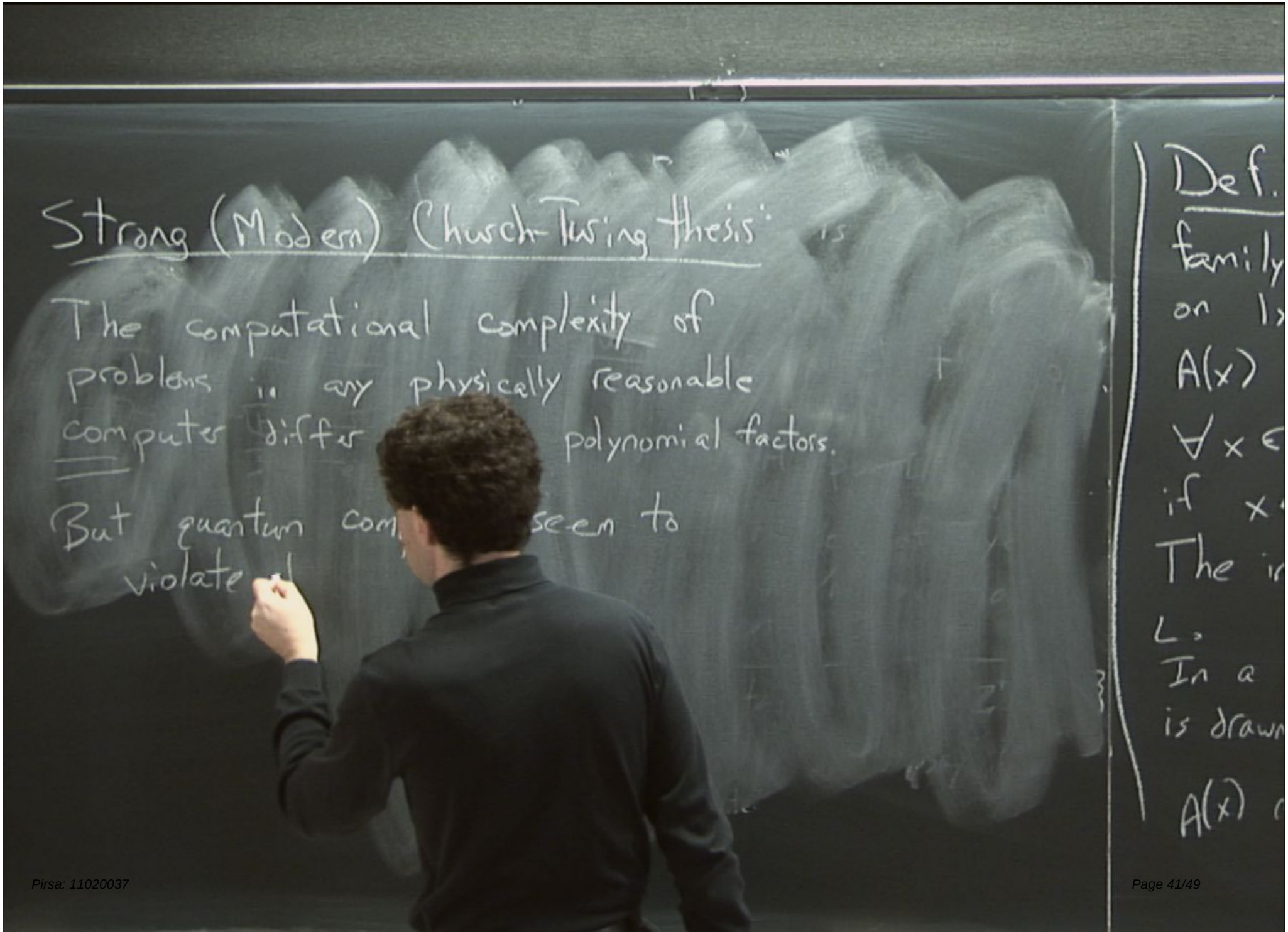

Note: Some languages are not decideable!
E.g. halting problem. - given input x
which is code for a computer program,
does x halt af
ite time?
Church-Turing thesis:
computable
functions is the sam
reasonable computer.
This is still true for a
Can compute output

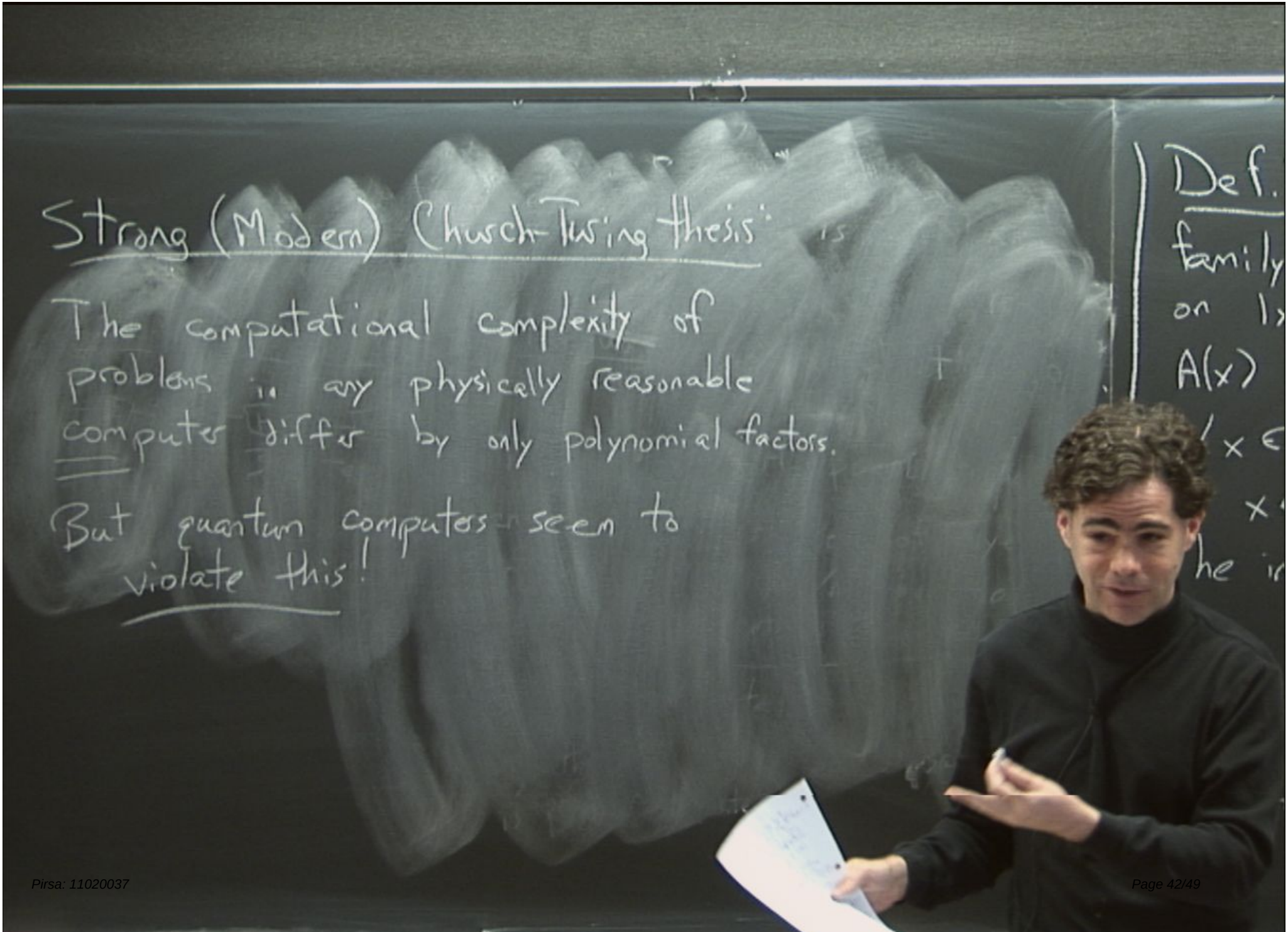

Note: Some languages are not decideable!
E.g. halting problem. - given input x
which is code for a computer program,
does x halt after finite time?
Church-Turing thesis: Set of computable
functions is the same for any physically
reasonable computer.
This is still true for a quantum computer:
Can compute output of QC by multiplying matrices.

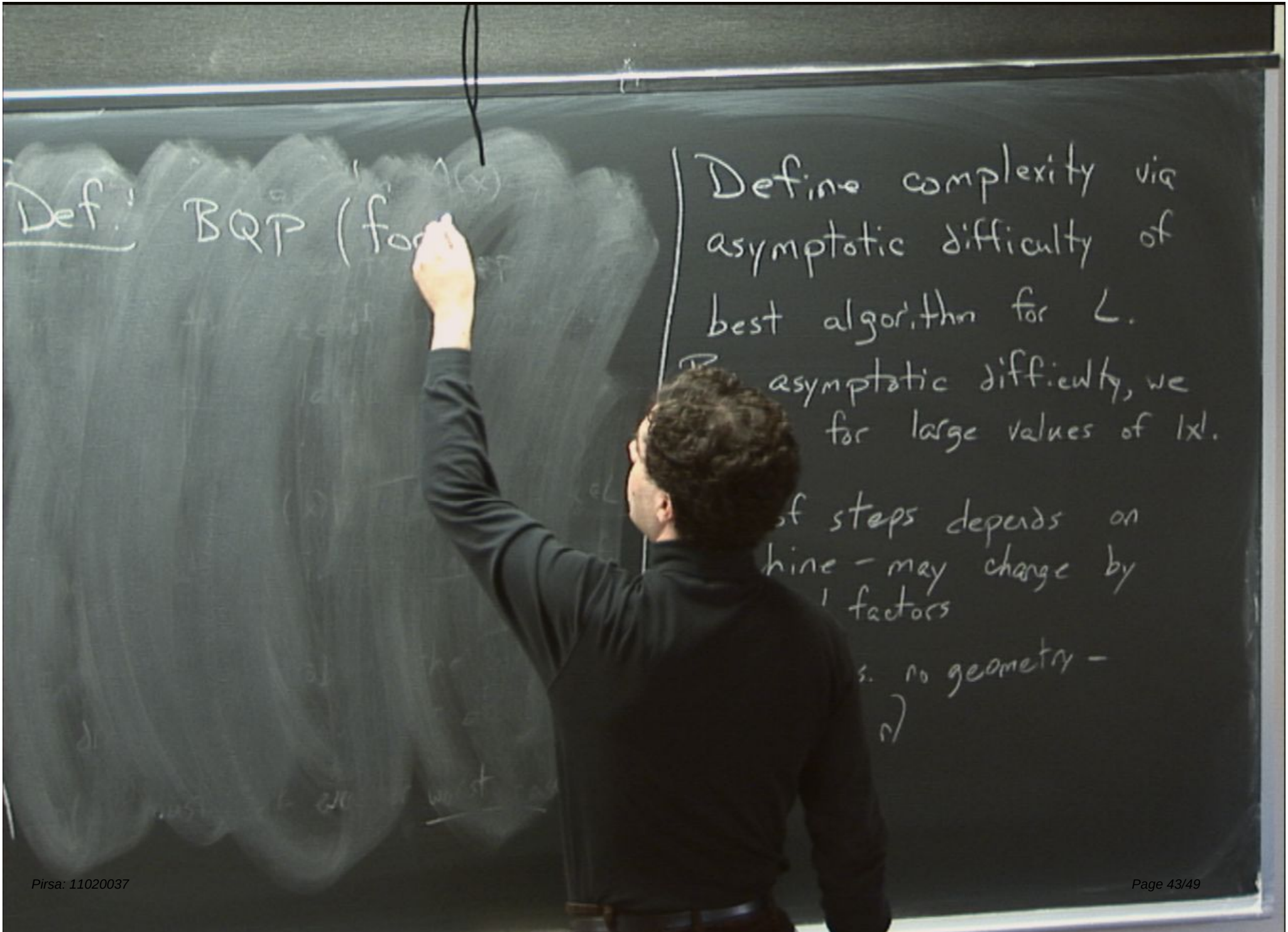

Strong (Modern) Church-Turing thesis:
Def.
family
on 1
A(x)
∀x ∈
if x
The ir
L.
In a
is drawn
A(x)

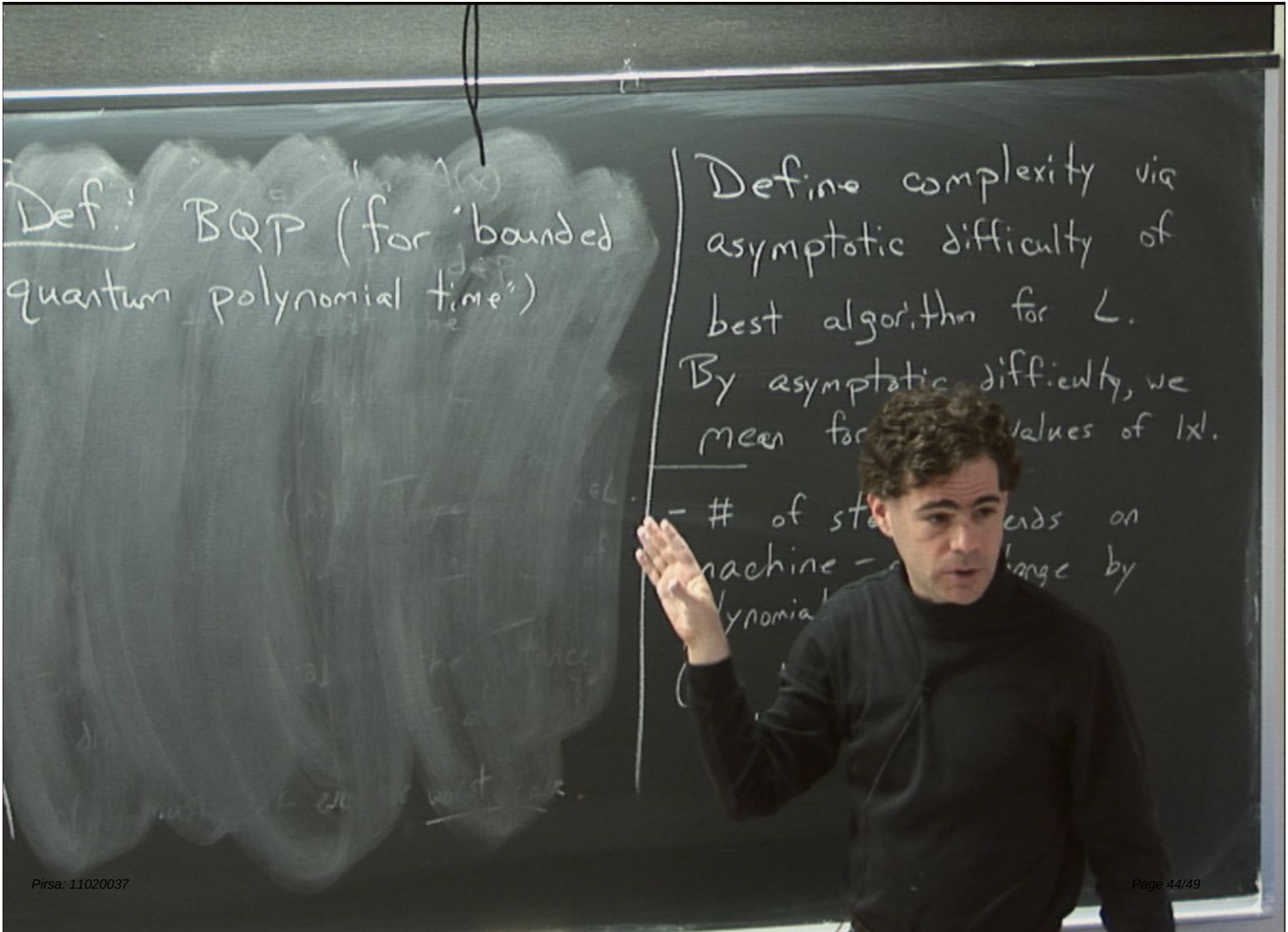

Strong (Modern) Church-Turing thesis:
The computational complexity of
Def.
family
on 1
A(x)
∀ x ∈
if x
The ir
L.
In a
is drawn
A(x)

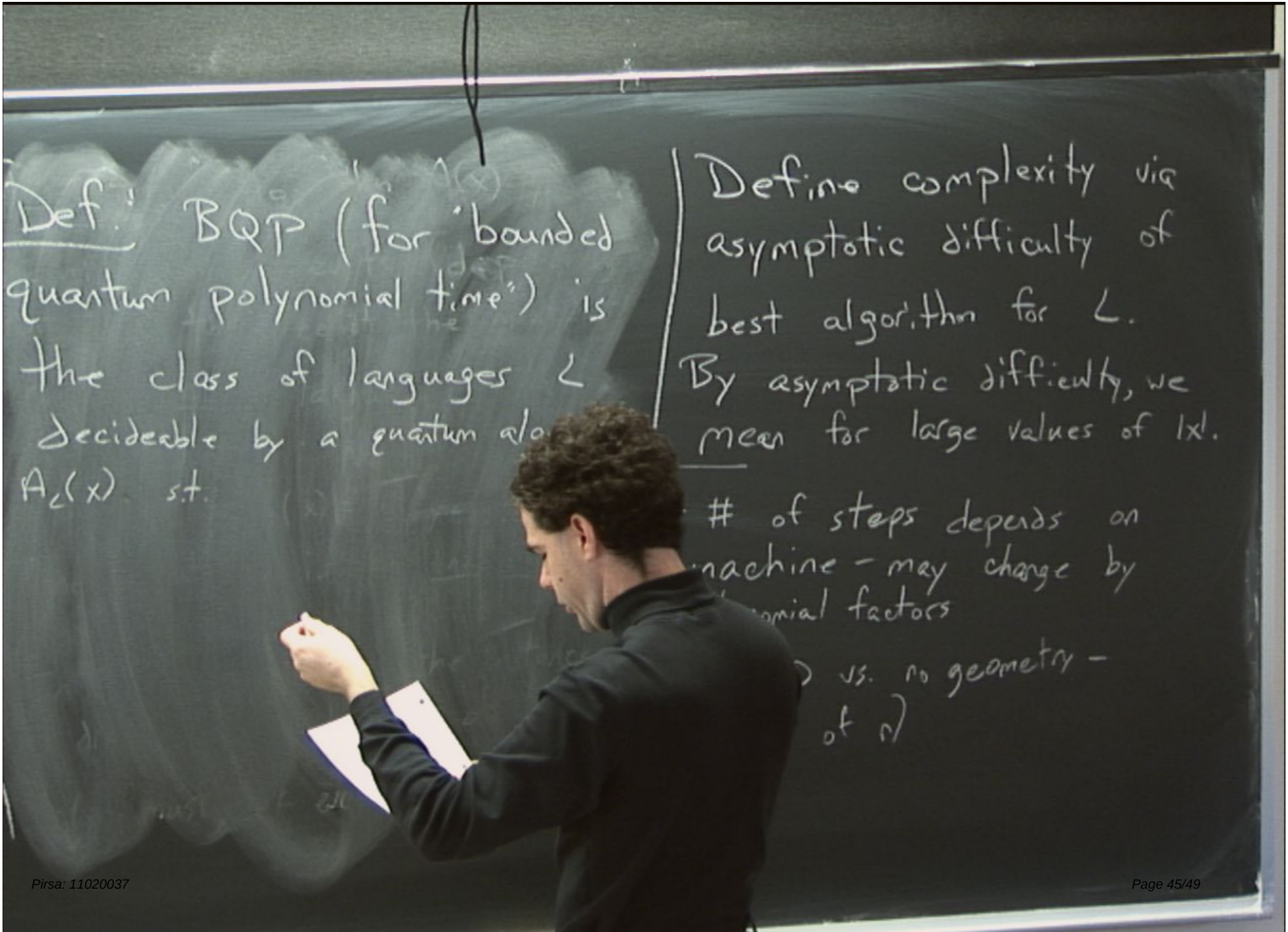

Strong (Modern) Church-Turing thesis:
The computational complexity of
problems in any physically reasonable
computer differ by only polynomial factors.
Def.
family
on 1
A(x)
∀x ∈
if x
The ir
In a
is drawn
A(x)

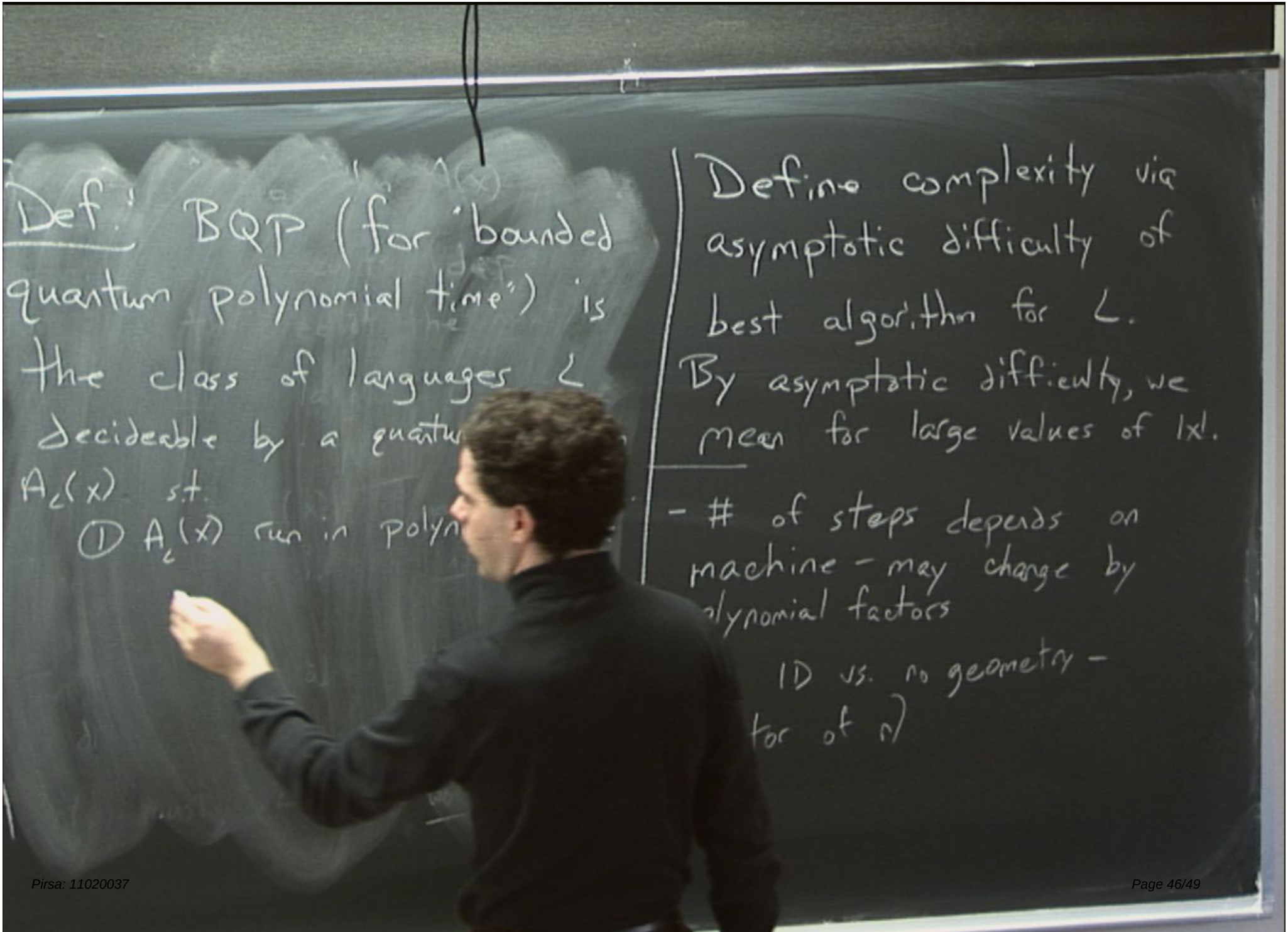

Strong (Modern) Church-Turing thesis:
The computational complexity of
problems in any physically reasonable
computer differ polynomial factors.
But quantum com seen to
violate
Def.
family
on 1
A(x)
∀ x ∈
if x
The i
L.
In a
is drawn
A(x)

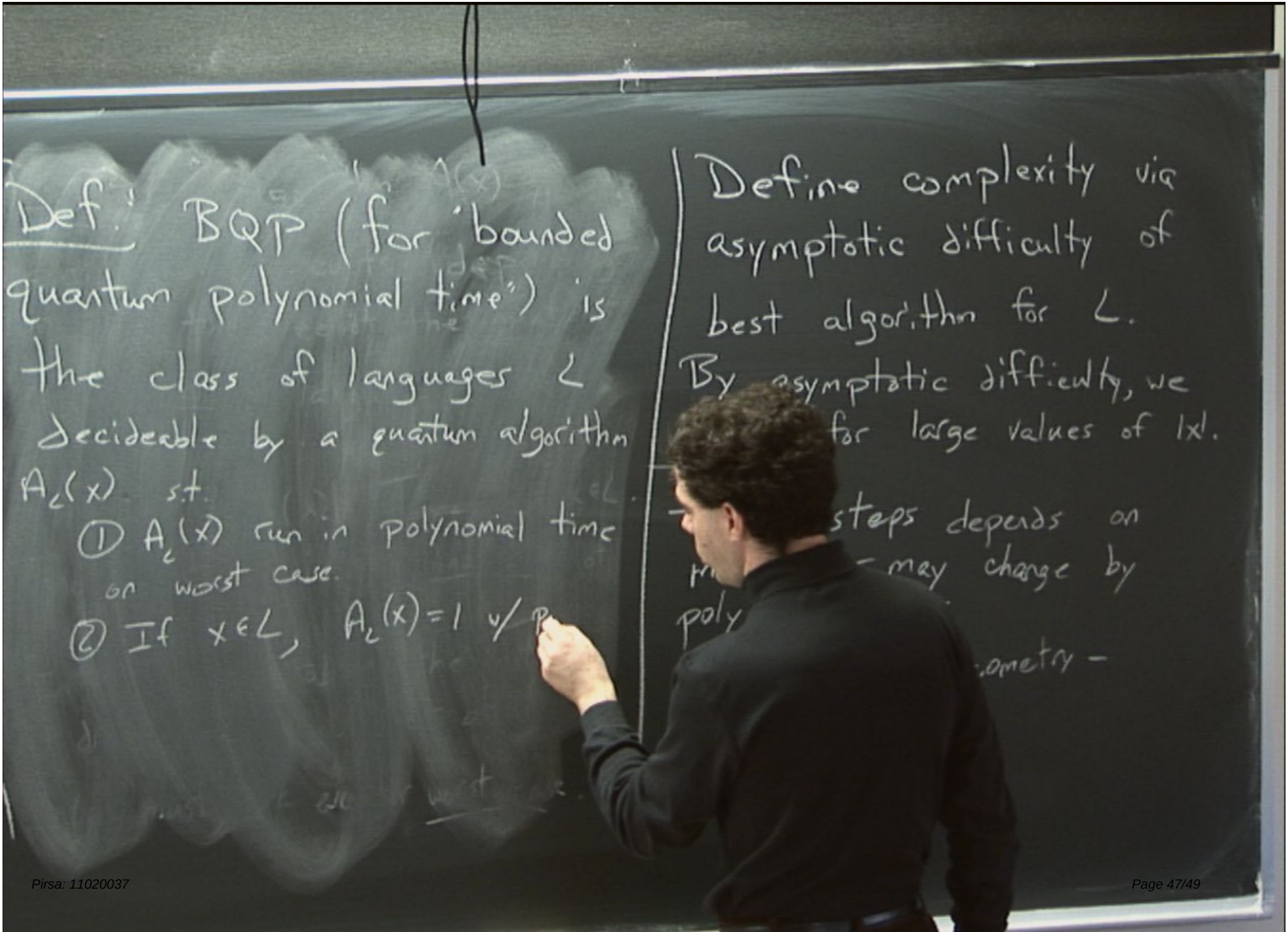

Strong (Modern) Church-Turing thesis:
The computational complexity of problems in any physically reasonable computer differ by only polynomial factors.
But quantum computers seem to violate this!
Def.
family
on
A(x)


Def: BQP (fo
Define complexity via
asymptotic difficulty of
best algorithm for L.
asymptotic difficulty, we
for large values of |x|.
f steps depends on
hine - may change by
factors
s. no geometry -
n)


Def: BQP (for "bounded quantum polynomial time")
Define complexity via asymptotic difficulty of best algorithm for L.
By asymptotic difficulty, we mean
- # of st
ads on
achine -
ange by
ynomia


Def: BQP (for 'bounded quantum polynomial time') is the class of languages L decidable by a quantum alg
$A_L(x)$ s.t.
Define complexity via asymptotic difficulty of best algorithm for L.
By asymptotic difficulty, we mean for large values of |x|.
# of steps depends on machine - may change by
vs. no geometry -
of n)


Def: BQP (for "bounded quantum polynomial time") is the class of languages L decidable by a quantu
$A_L(x)$ s.t.
① $A_L(x)$ run in polyn
Define complexity via asymptotic difficulty of best algorithm for L.
By asymptotic difficulty, we mean for large values of |x|.
- # of steps depends on machine - may change by
olynomial factors
1D vs. no geometry -
tor of n)


Def: BQP (for 'bounded quantum polynomial time') is the class of languages L decideable by a quantum algorithm $A_L(x)$ s.t.
① $A_L(x)$ run in polynomial time on worst case.
② If $x \in L$, $A_L(x) = 1$ w/
Define complexity via asymptotic difficulty of best algorithm for L.
By asymptotic difficulty, we for large values of |x|.
steps depends on
may change by
poly
ometry -

Def: BQP (for "bounded quantum polynomial time") is the class of languages L decidable by a quantum algorithm $A_L(x)$ s.t.

1. $A_L(x)$ run in polynomial time on worst case.
2. If $x \in L$, $A_L(x) = 1$ w/ prob. $\geq \frac{2}{3}$
3. If $x \notin L$, $A_L(x) = 0$ w/ prob. $\geq \frac{2}{3}$

($A_L(x)$ outputs a random variable w/ this value)

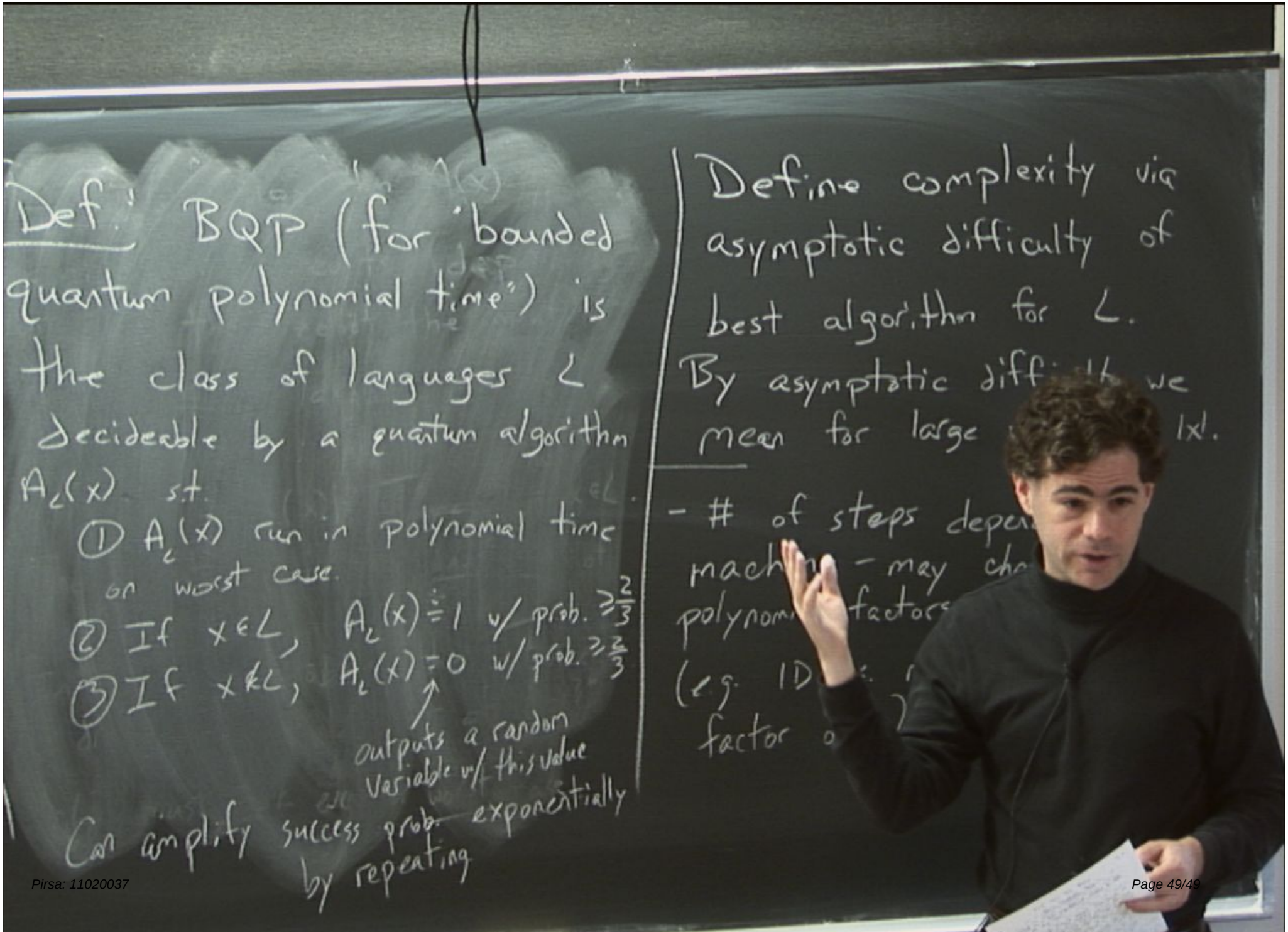
Can amplify success prob. exponentially by repeating

Define complexity via asymptotic difficulty of best algorithm for L.

By asymptotic difficulty, we mean for lar[illegible] of $|x|$.

- # of steps [illegible] on machine - may [illegible] by polynomial fact[illegible]

(e.g. ID vs. [illegible] factor of n)


Def: BQP (for "bounded quantum polynomial time") is the class of languages L decideable by a quantum algorithm $A_L(x)$ s.t.
① $A_L(x)$ run in polynomial time on worst case.
② If $x \in L$, $A_L(x) = 1$ w/ prob. $\geq \frac{2}{3}$
③ If $x \notin L$, $A_L(x) = 0$ w/ prob. $\geq \frac{2}{3}$
outputs a random variable w/ this value
Can amplify success prob. exponentially by repeating
Define complexity via asymptotic difficulty of best algorithm for L.
By asymptotic diff... we mean for large ... $|x|$.
- # of steps depe... machi... - may ch... polynom... factors
(e.g. ID... factor o...