HOW LONG WILL MY SYNTHESIZER RUN FOR?

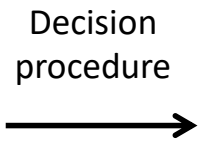
Loris D'Antoni

University of Wisconsin Madison

Joint work with Aws Albarghouthi and Samuel Drews

The good old days of synthesis

Friendly logic



Friendly program fragment

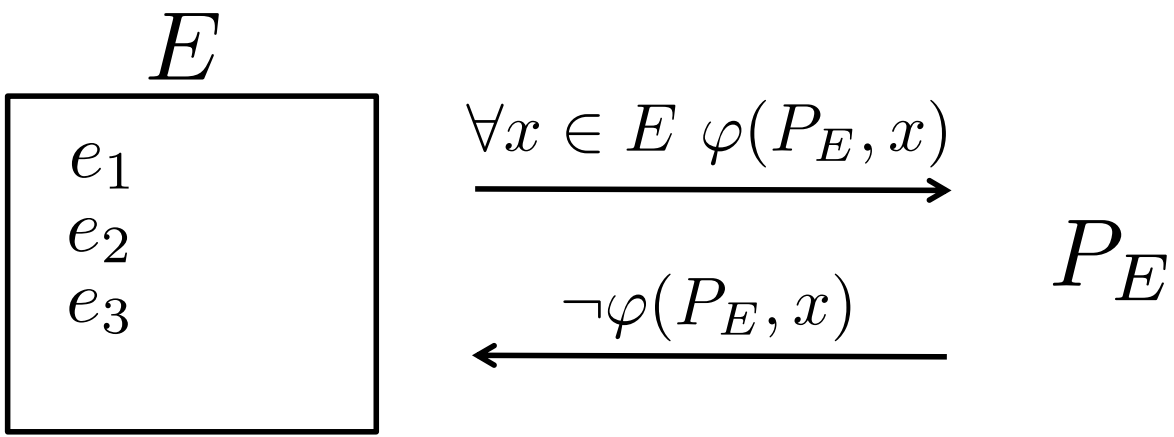
Synthesis: Dreams $\Longrightarrow$ Programs

ZOHAR MANNA AND RICHARD WALDINGER

$$\exists P \; \forall x \; \varphi(P, x)$$

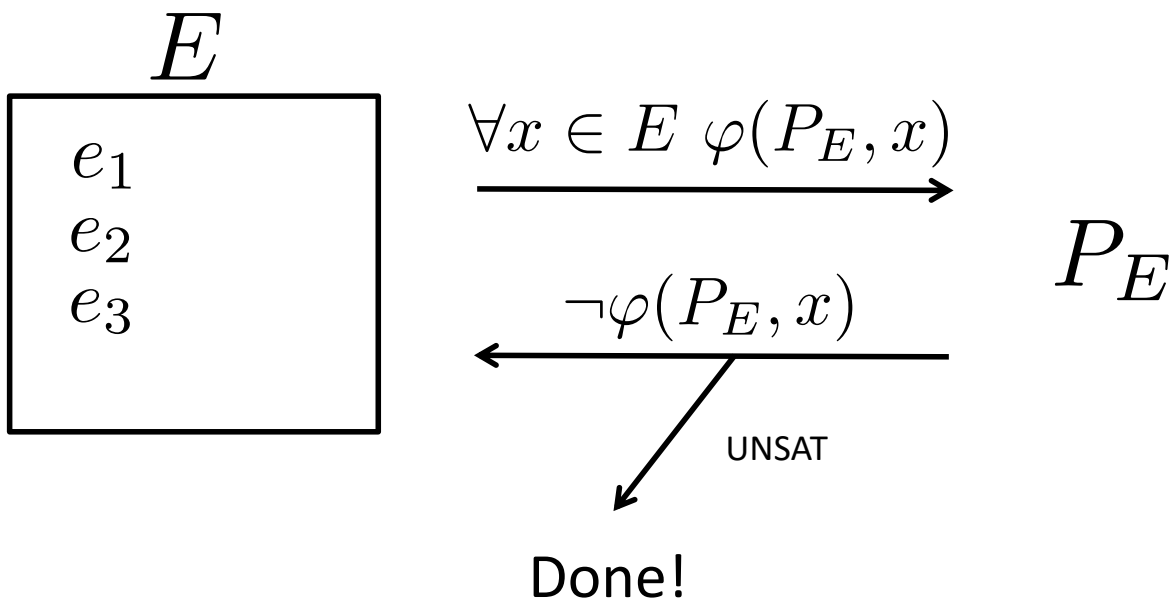
Inductive synthesis

$$\exists P \in \mathcal{S}\ \forall x\ \varphi(P, x)$$



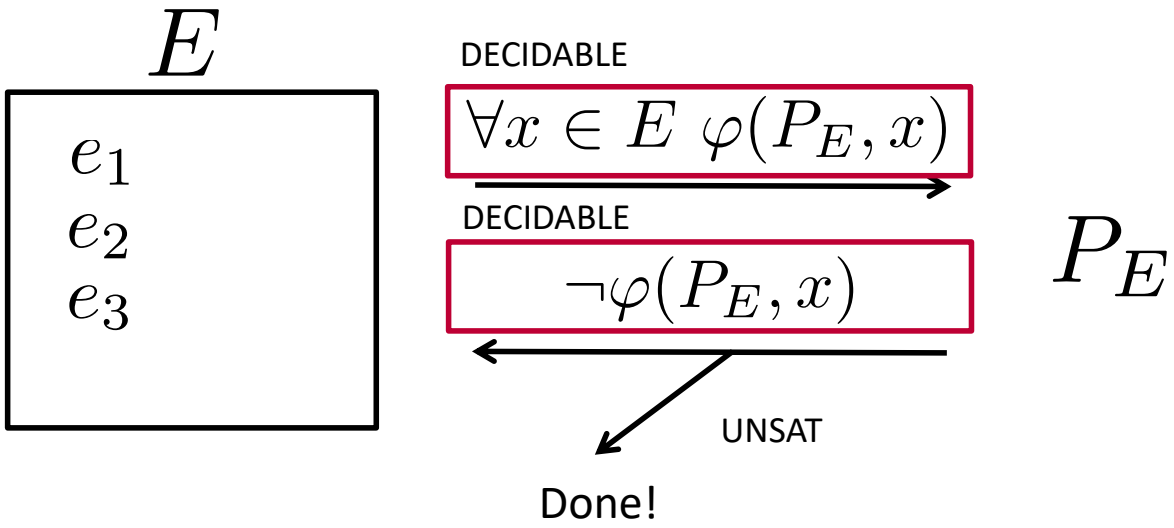
Inductive synthesis

$$\exists P \in \mathcal{S}\ \forall x\ \varphi(P, x)$$



Inductive synthesis

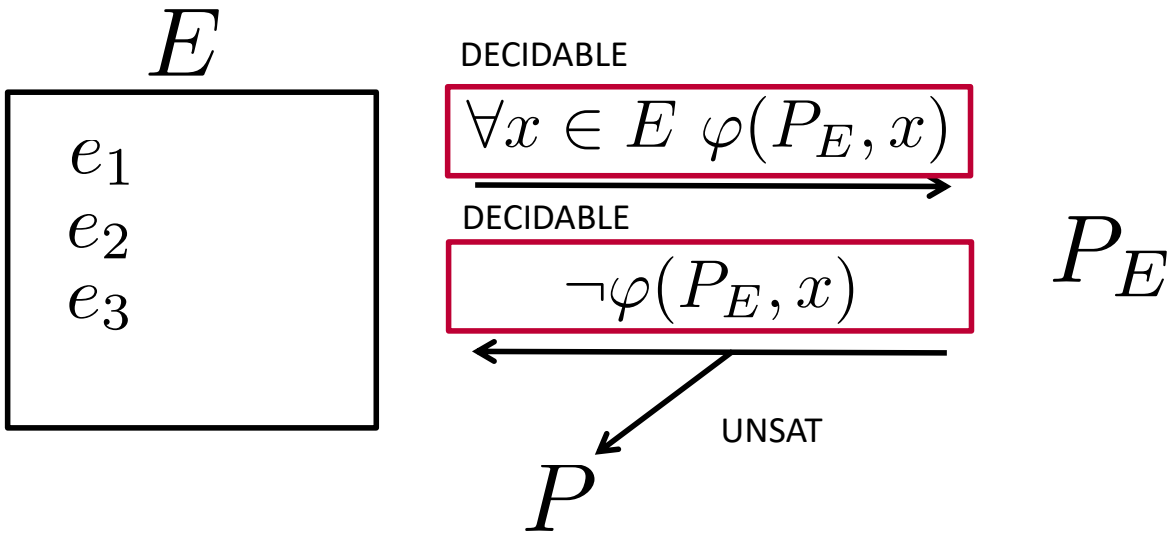
$$\exists P \in \mathcal{S}\ \forall x\ \varphi(P, x)$$



Does this algorithm always terminate?

Inductive synthesis

$$\exists P \in \mathcal{S}\ \forall x\ \varphi(P, x)$$



Does this algorithm always terminate
if a solution to the synthesis problem exists?

In general NO $\exists P{\in}\mathcal{S}\ \forall x\ \varphi(P,x)$

$$\mathcal{S} = \{P(x){=}c \mid c{\geq}0\}$$

$$\exists P{\in}\mathcal{S}\ \forall x\ P(x){>}x$$

No matter what the example set is, there is a program in the search space consistent with it

$$E = \{1,4,7,8\} \qquad P_E(x) = 8$$

Synthesis: Dreams $\Longrightarrow$ Programs

ZOHAR MANNA AND RICHARD WALDINGER

Will we keep dreaming forever?

Will my synthesizer ever terminate?

How long will my synthesizer run for?

This talk

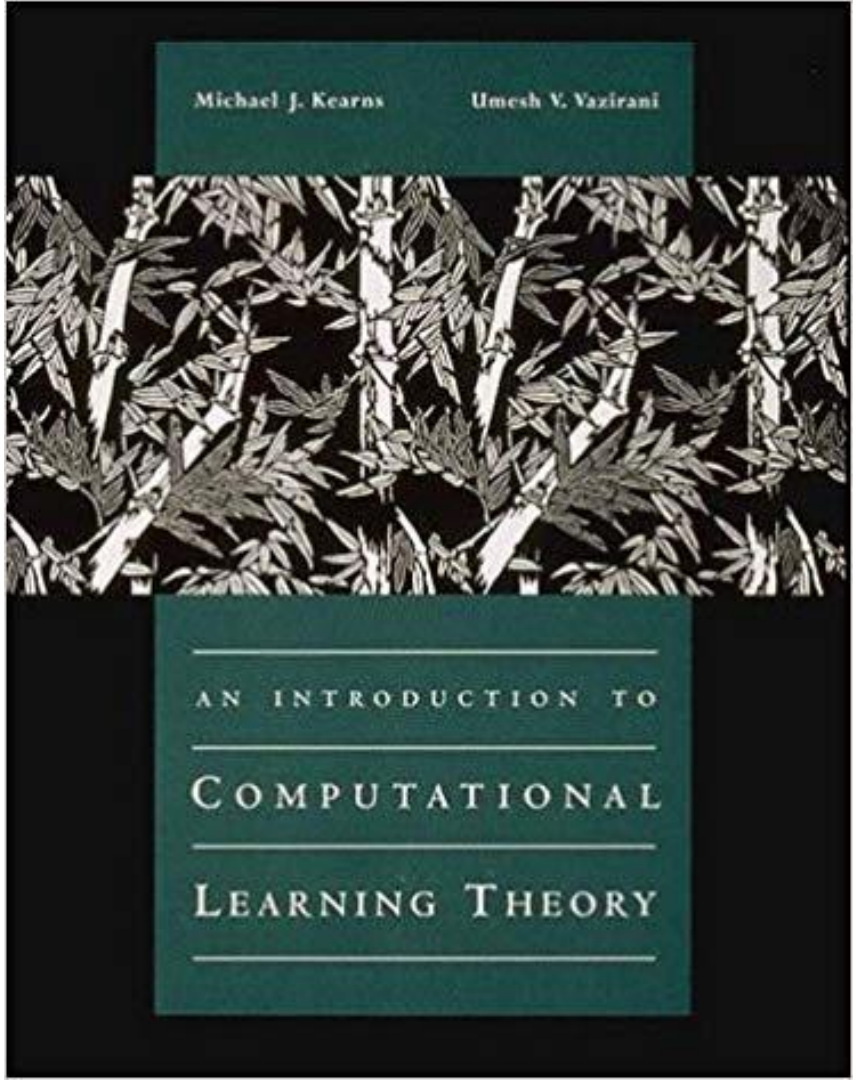
Problem: How long will my synthesizer run for?

A connection to inductive learning

DIGITS: Distribution Guided Inductive Synthesis

IS THIS REALLY A NEW QUESTION?

When can functions be learned?



Computational learning theory in a nutshell

Probably Approximately Correct (PAC) learning

Certain types of functions

Can be learned approximately

Using polynomially many IO examples

What is so different about program synthesis?

Inductive learning

Target function I'm trying to learn

$f(x) = 2x+2$

Labeled input-output examples

(1,4), (2,6)

Inductive synthesis

Target relation

$f(x)>x$

Input examples but no outputs!

(1,??), (2,??)

What is so different about program synthesis?

Inductive learning	Inductive synthesis
Certain types of functions	Search space is a mess
Can be learned approximately	We can’t be sort of right
Using polynomially many IO examples	We don’t have labeled examples

How long will my synthesizer run for?

To answer this question, we need to address these problems

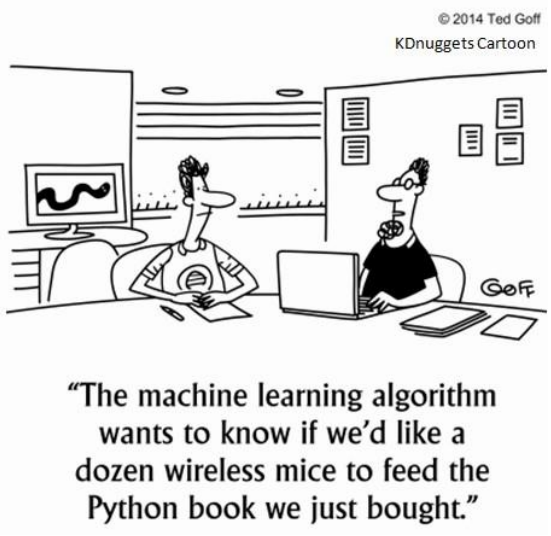
Can we develop a *computational synthesis theory*?

This talk

Problem: How long will my synthesizer run for?

A connection to inductive learning

DIGITS: Distribution Guided Inductive Synthesis



PROGRAM REPAIR UNDER UNCERTAINTY

REPAIRING PROGRAMS WITH PROBABILISTIC INPUTS

WITH: SAMUEL DREWS, AWS ALBARGHOUTHI [OOPSLA17, CAV17, CAV19]

Let’s focus on a concrete problem

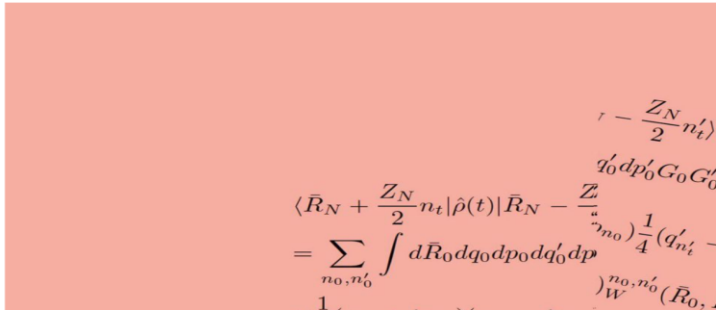
Everyone can now use machine learning

Machine learning library

Decision tree
SVM
DNN

Internet Of Things

Who do you blame when an algorithm gets you fired?



TheUpshot

HIDDEN BIAS

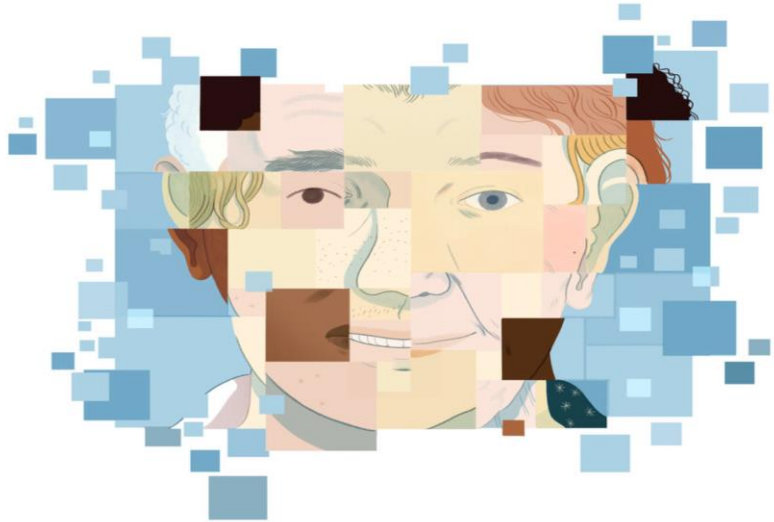
When Algorithms Discriminate

Claire Cain Miller @clairecm JULY 9, 2015

SundayReview | OPINION

Artificial Intelligence's White Guy Problem

By KATE CRAWFORD JUNE 25, 2016



Bianca Bagnarelli

Algorithmic fairness for programs (example)

$$\{v \sim \mathcal{M}\} \longrightarrow$$

Population model:
probability distribution of inputs

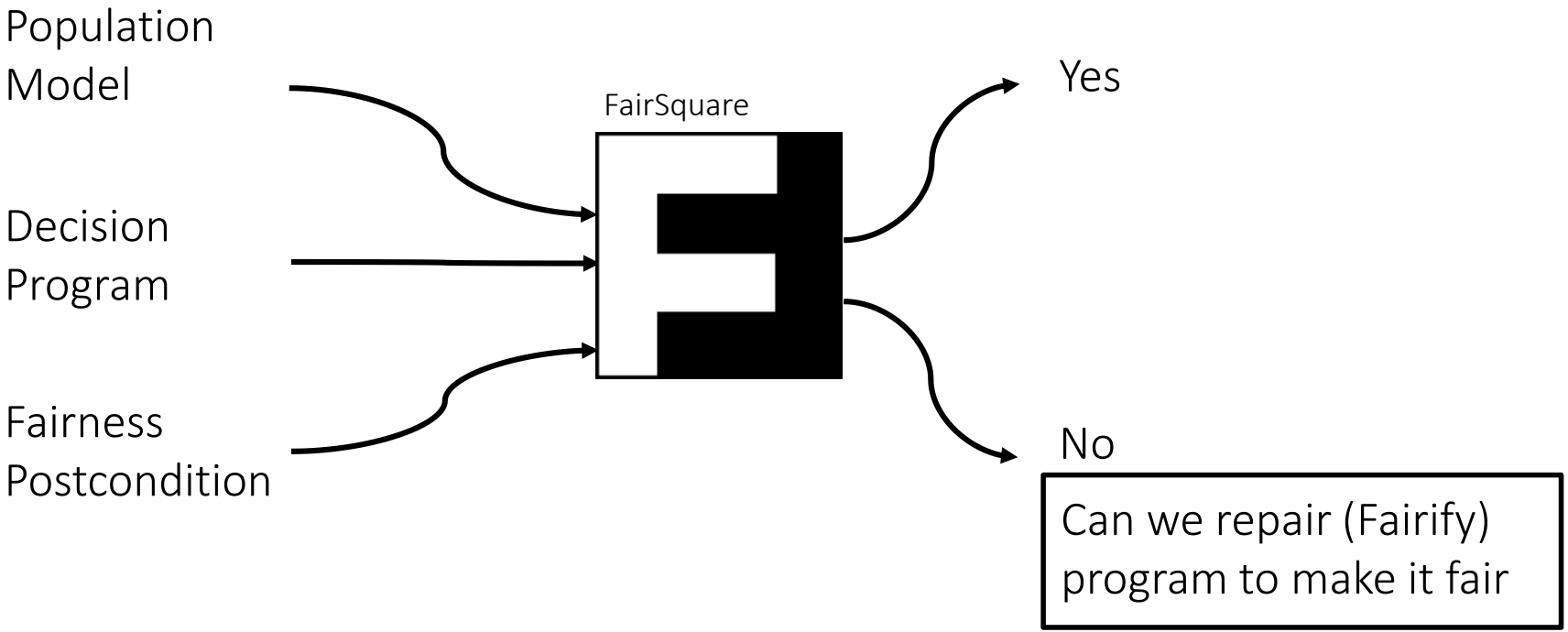
$$h \leftarrow \mathcal{D}(v) \longrightarrow$$

Decision making program:
returns true or false

$$\left\{ \frac{\Pr[h \mid v_s]}{\Pr[h \mid \neg v_s]} > 1 - \epsilon \right\} \longrightarrow$$

Probabilistic postcondition:
Probability of hiring given that candidate is woman should be close to probability of hiring given that is man

Verifying algorithmic fairness [OOPSLA17]



Probabilistic program repair: definition

Input

$$\{v \sim \mathcal{M}\}$$
$$h \leftarrow \mathcal{D}(v)$$
$$\{post\}$$

The postcondition **does not hold**

Output

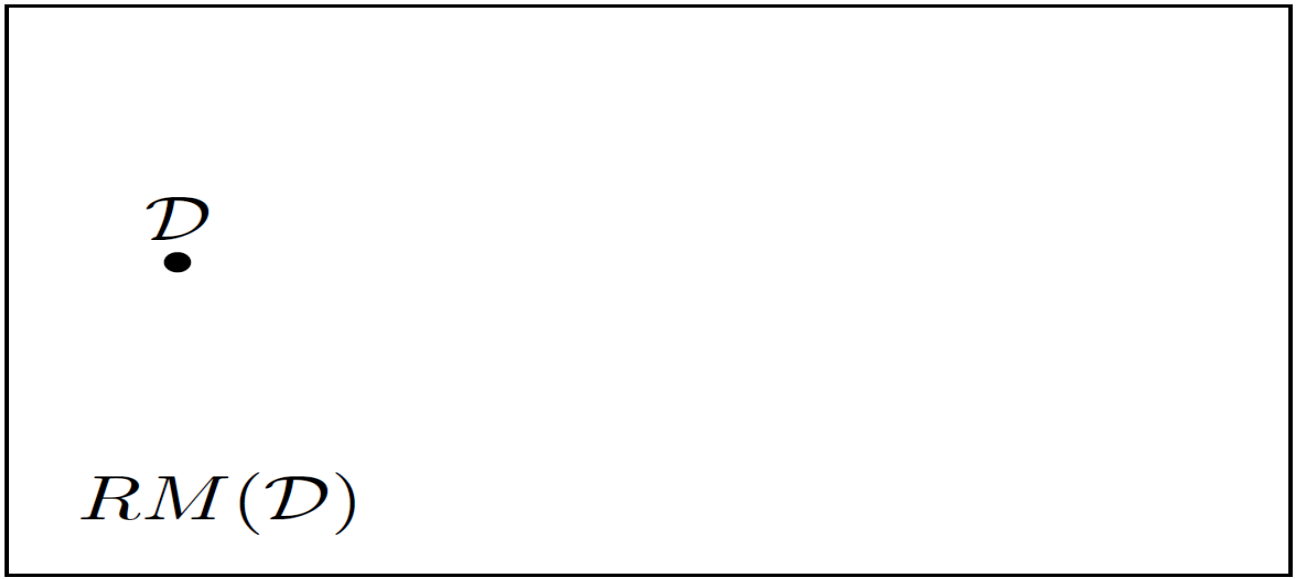
Program $\mathcal{D}'$ in repair model $RM(\mathcal{D})$

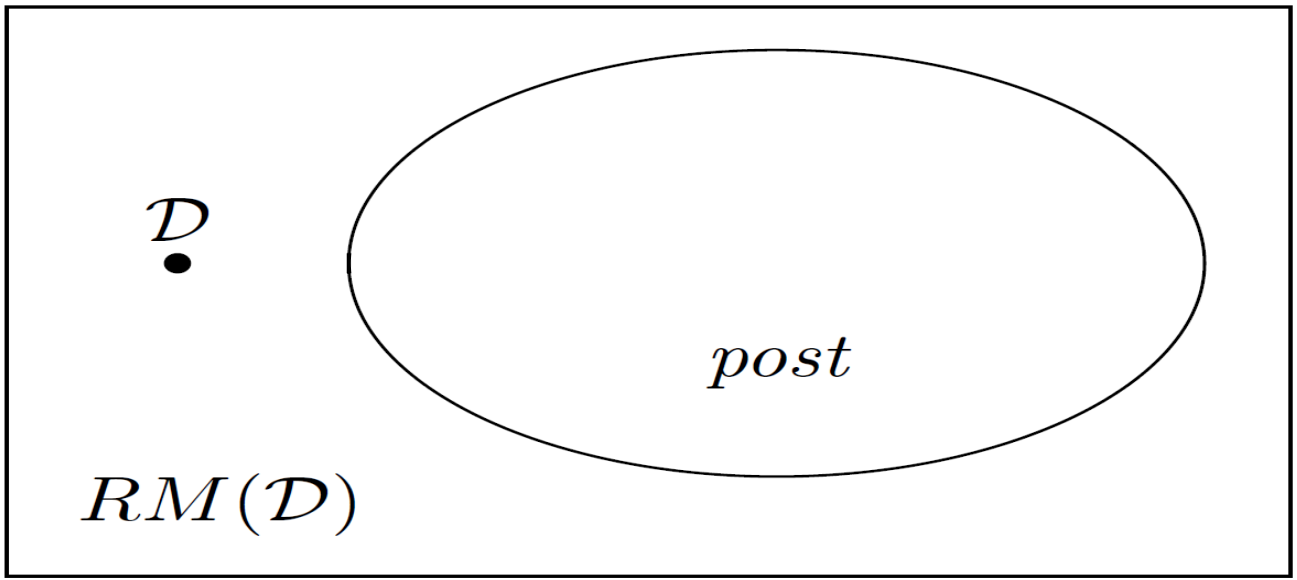
$$\{v \sim \mathcal{M}\}$$
$$h \leftarrow \mathcal{D}'(v)$$
$$\{post\}$$

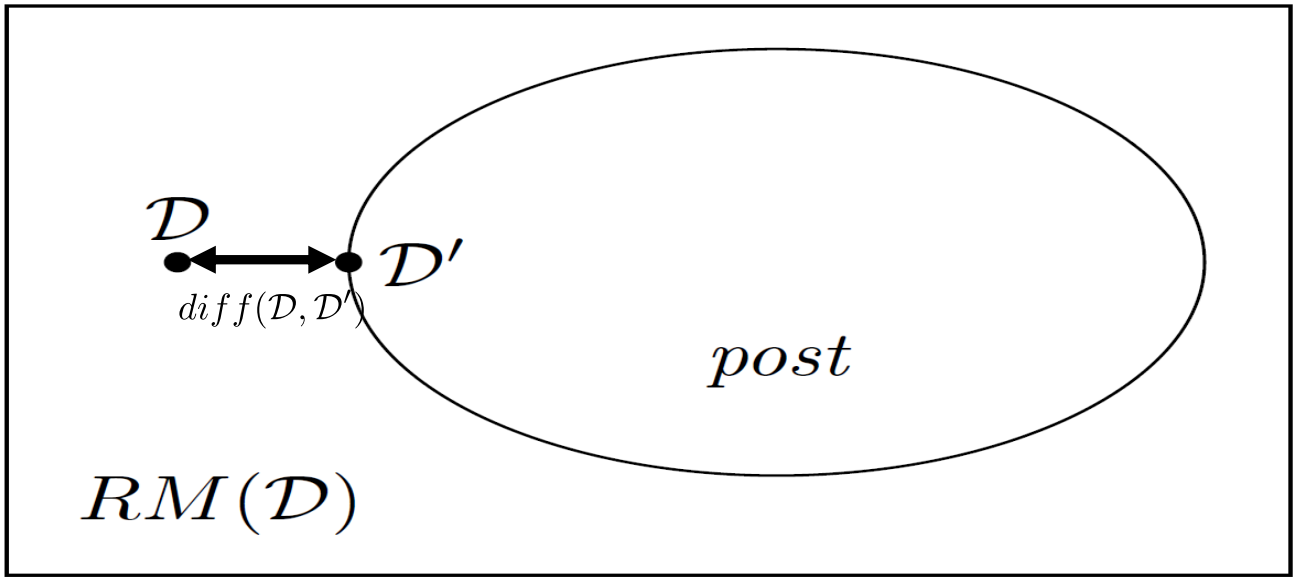
The postcondition **holds**

Difference between $\mathcal{D}$ and $\mathcal{D}'$ is minimal

$$diff(\mathcal{D}, \mathcal{D}') = Pr(\mathcal{D}(v) \neq \mathcal{D}'(v) \mid v \sim \mathcal{M})$$

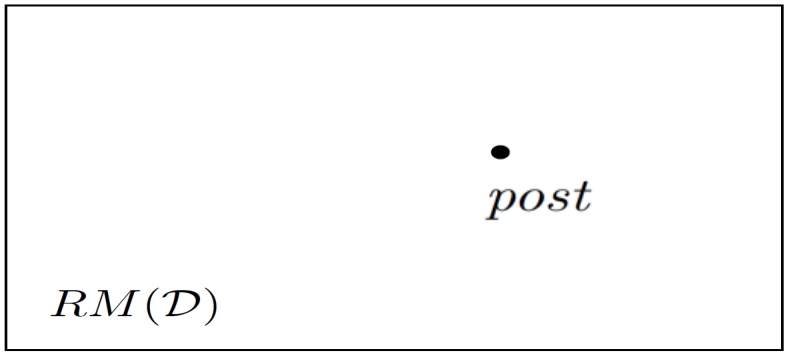

$\mathcal{D}$
$RM(\mathcal{D})$


$\mathcal{D}$
$post$
$RM(\mathcal{D})$

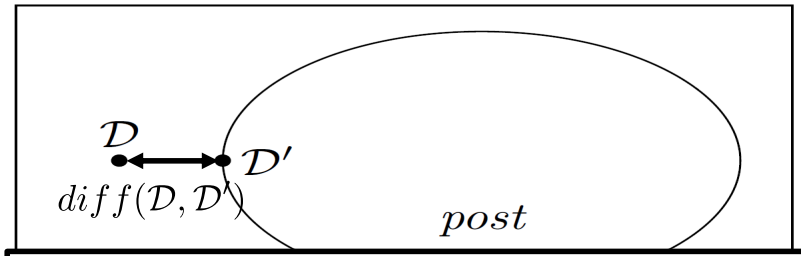

$\mathcal{D}$
$\mathcal{D}'$
$diff(\mathcal{D}, \mathcal{D}')$
$post$
$RM(\mathcal{D})$

What is so different about program synthesis? [reprised]

Learning



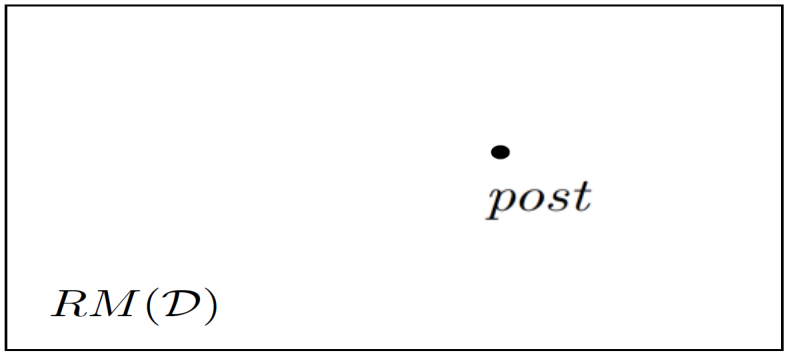
Synthesis



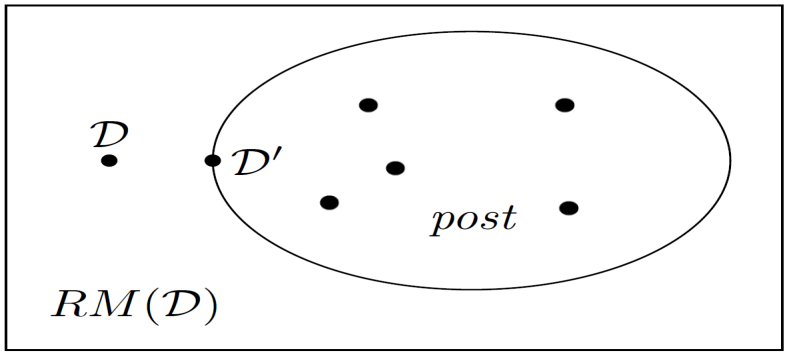
Challenge: no ground truth. What should the output of $\mathcal{D}'$ be on each input?

How do we get our input/output examples?

Learning



Synthesis



Can we approximately solve the synthesis problem using boundedly many examples?

DISTRIBUTION GUIDED INDUCTIVE SYNTHESIS

Distribution-Guided InducTive Synthesis (DIGITS)

I’m going to present a stupid algorithm

Show that it actually works

Inspired by the following

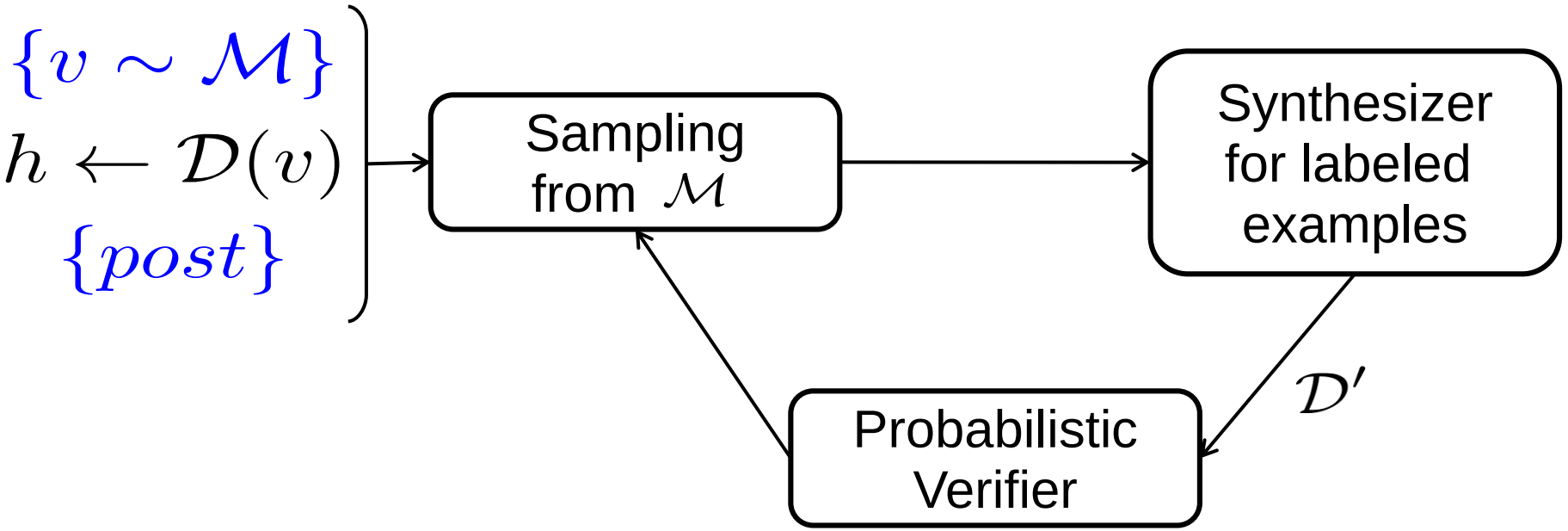
It works -> It ain’t stupid

I prove the algorithm isn’t that stupid and terminates using few examples [CAV17]

Tuesday: Sam will show you that, not only it’s not that stupid, it’s actually pretty smart!

Great complexity bound [Synthesis session at CAV19]

DIGITS Components



DIGITS Distribution directed inductive synthesis

Sample **n** inputs from $\mathcal{M}$

inp_1	inp_2	inp_3	...	inp_n
T	T	T	...	T
T	T	T	...	F
T	T	F	...	T
T	T	F	...	F
T	F	T	...	T
..	..	..	...	...

Synthesize one program per labeling

Check the postcondition

$\{v \sim \mathcal{M}\}\ \mathcal{D}_1\ \{post\}$

~~$\{v \sim \mathcal{M}\}\ \mathcal{D}_2\ \{post\}$~~

UNSAT

$\{v \sim \mathcal{M}\}\ \mathcal{D}_4\ \{post\}$

~~$\{v \sim \mathcal{M}\}\ \mathcal{D}_5\ \{post\}$~~

Output $\mathcal{D}_i$ that with minimal $diff(\mathcal{D}, \mathcal{D}_i)$

Synthesis using sketching
Verification and difference using FairSquare

$\mathcal{D}$

```
fun hire(min,urank)
    dec = 1 <= urank <= 10
    return dec
```

$\mathcal{D}$

```
fun hire(min,urank)
  dec = 1 <= urank <= 10
  return dec
```

$RM(\mathcal{D})$

```
fun hireRep(min,urank)
  dec = ●1 <= urank <= ●2
  return dec
```

```
fun hireRep(min,urank)
    dec = ●1 <= urank <= ●2
    return dec
```

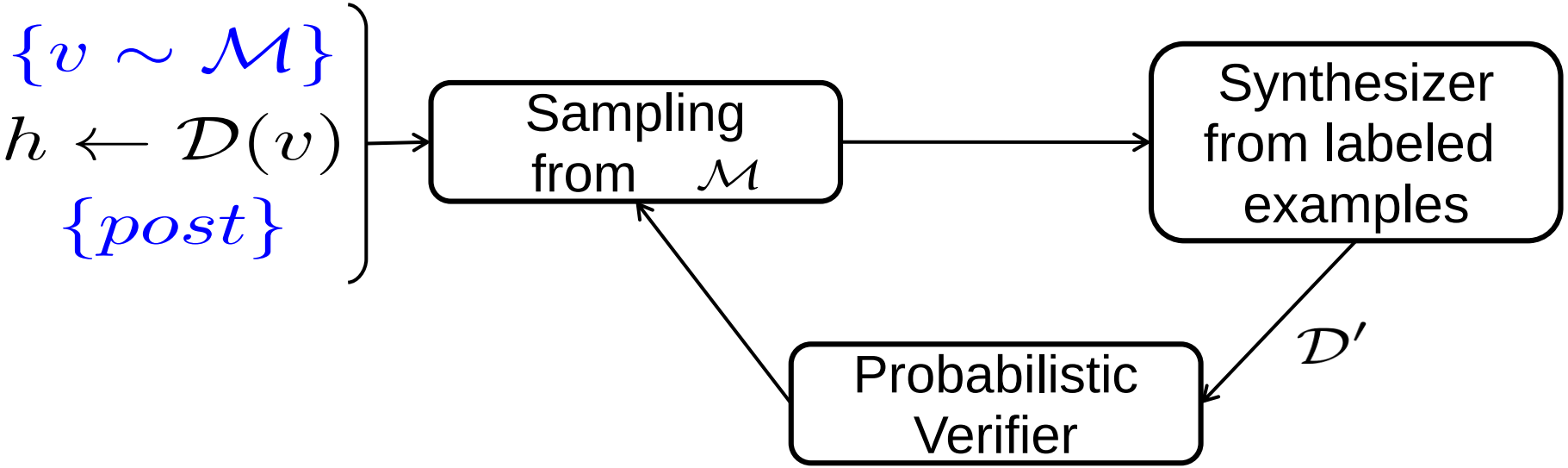
$$\varphi = ((H_1 \leq urank \leq H_2) == dec)$$

$$((H_1 \leq 12 \leq H_2) == true)$$
$$((H_1 \leq 17 \leq H_2) == false)$$

$$((H_1 \leq 12 \leq H_2) == true)$$
$$((H_1 \leq 17 \leq H_2) == false)$$

$$H_1 = 10, H_2 = 15$$

```
fun hireRep1(min,urank)
  dec = 10 <= urank <= 15
  return dec
```



How long will my synthesizer run for?

How many samples to find a solution?

Distribution-Guided InducTive Synthesis (DIGITS)

I'm going to present a stupid algorithm

Show that it actually works

Inspired by the following

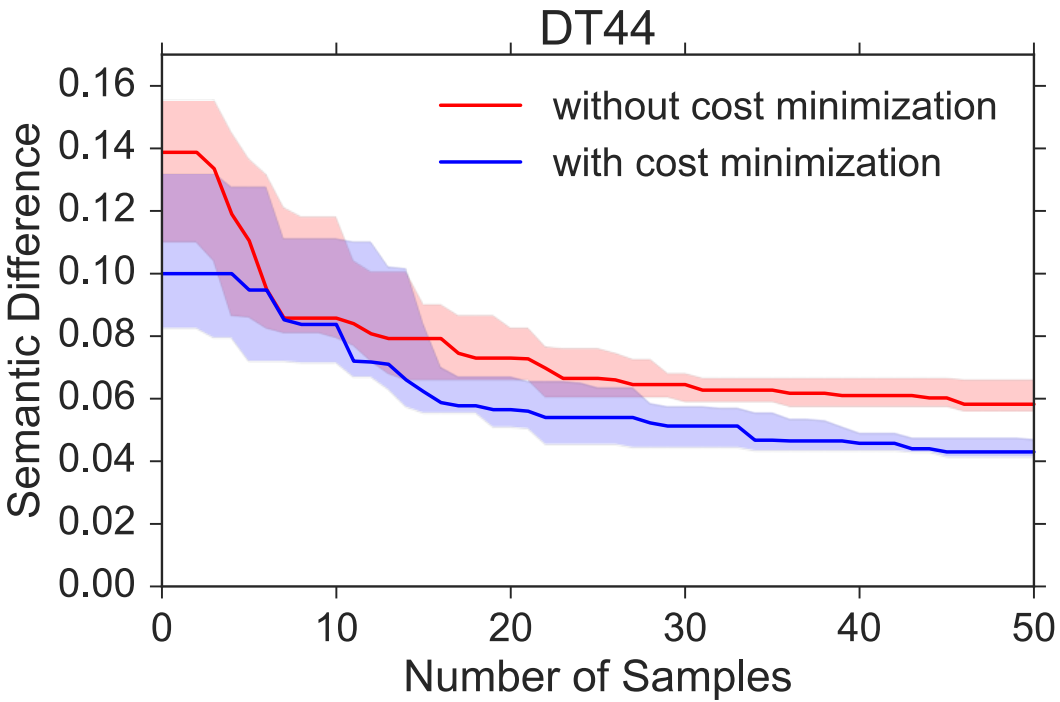
It works -> It ain't stupid

I prove that it isn't that stupid and terminates using few examples [CAV17]

Tuesday: Sam will show you that, not only it's not that stupid, it's actually pretty smart!

Great complexity bound [Synthesis session at CAV19]

Convergence on repairing a decision tree



Distribution-Guided InducTive Synthesis (DIGITS)

I'm going to present a stupid algorithm

Show that it actually works

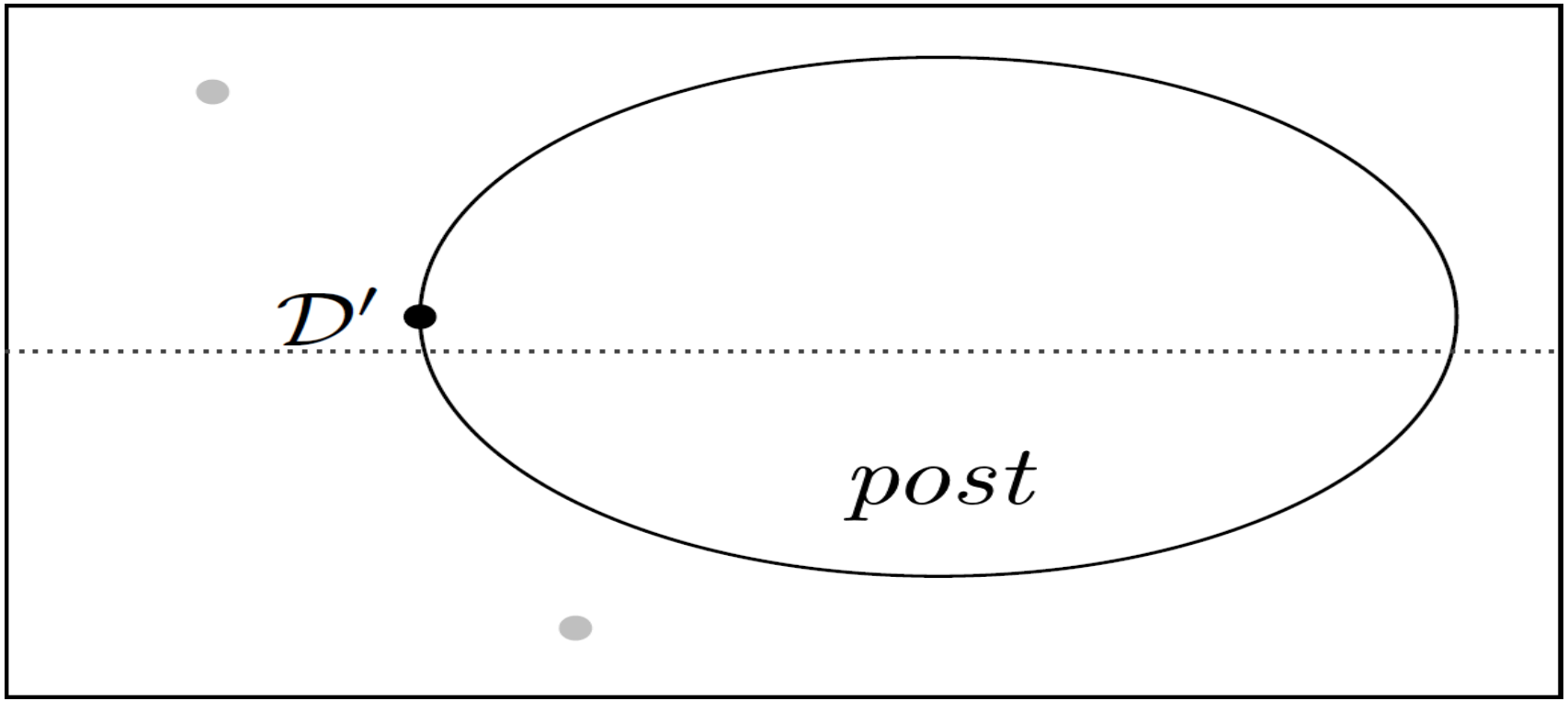
Inspired by the following

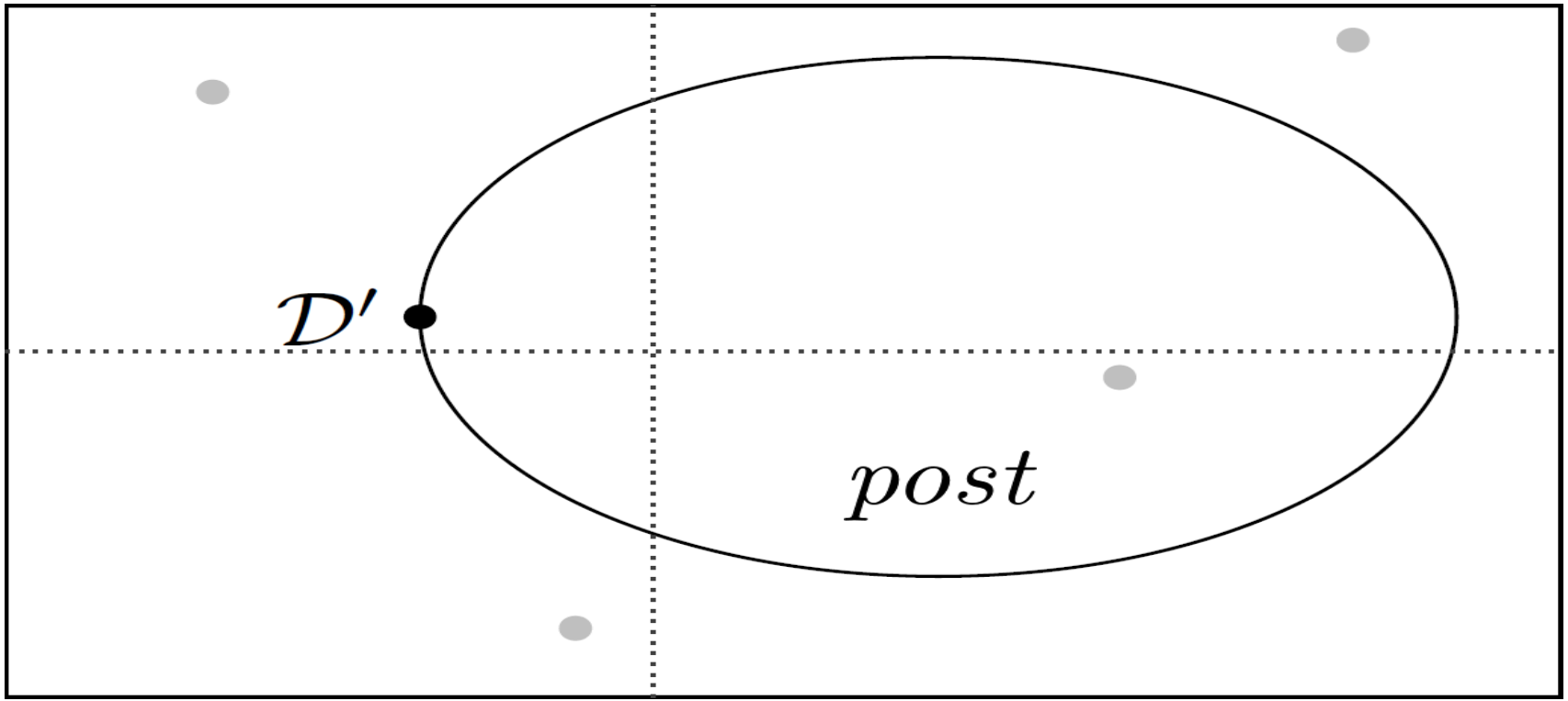
It works -> It ain't stupid

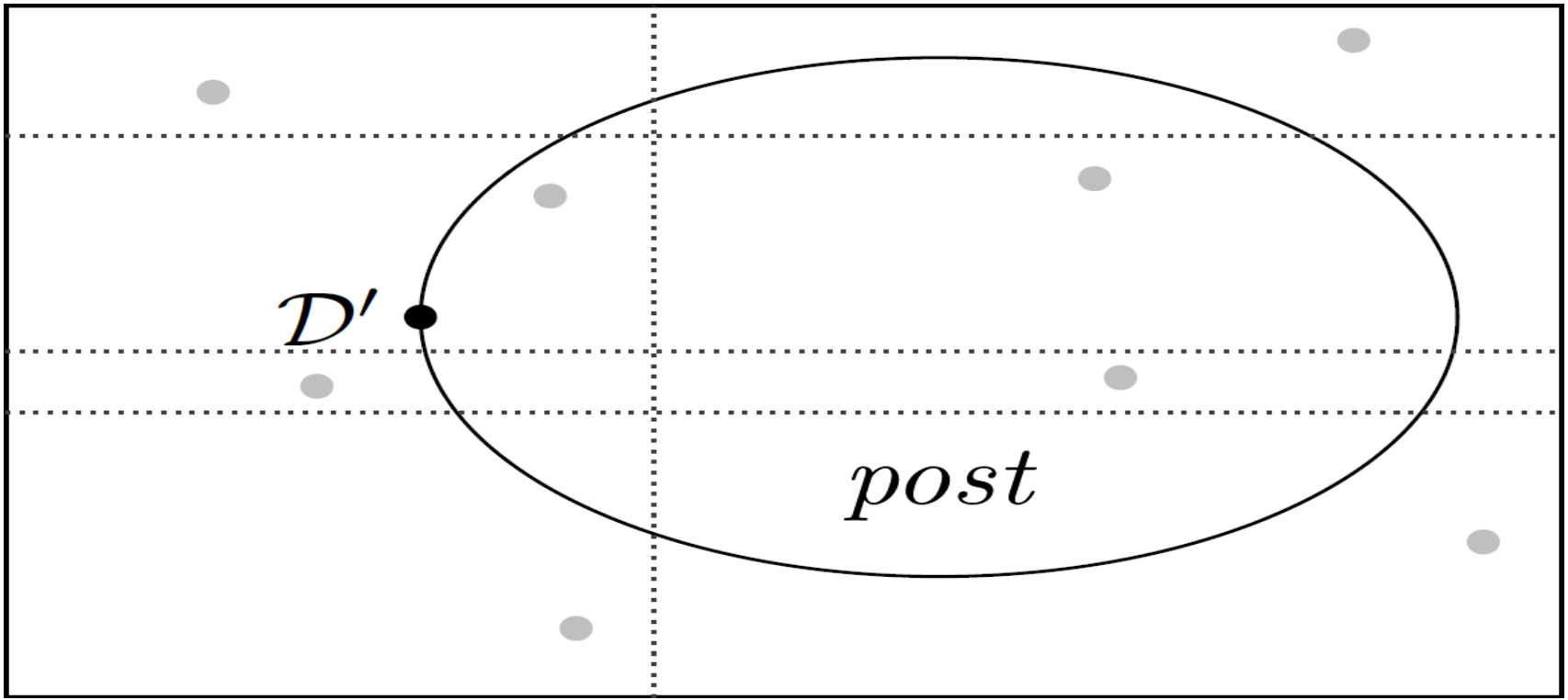
I prove that it isn't that stupid and terminates using few examples [CAV17]

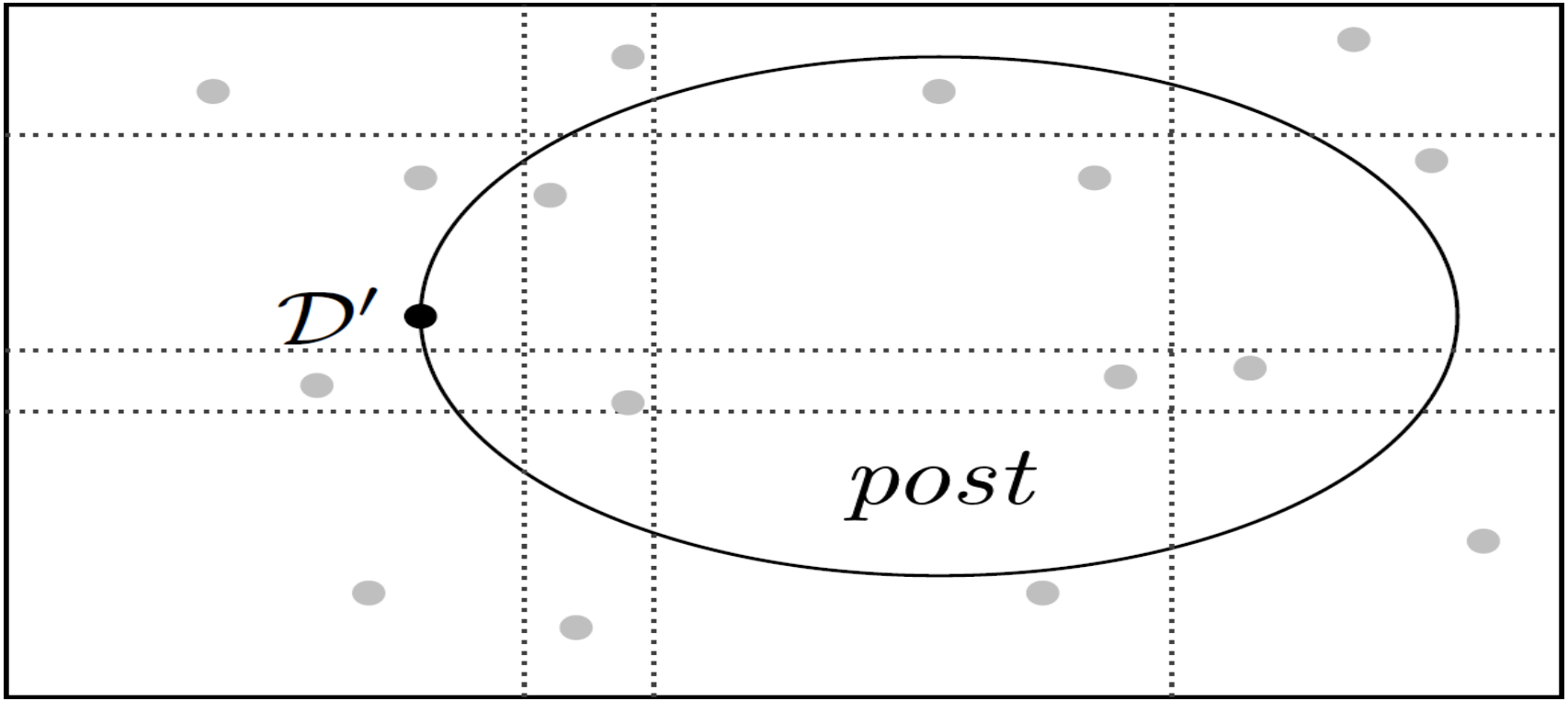
Tuesday: Sam will show you that, not only it's not that stupid, it's actually pretty smart!

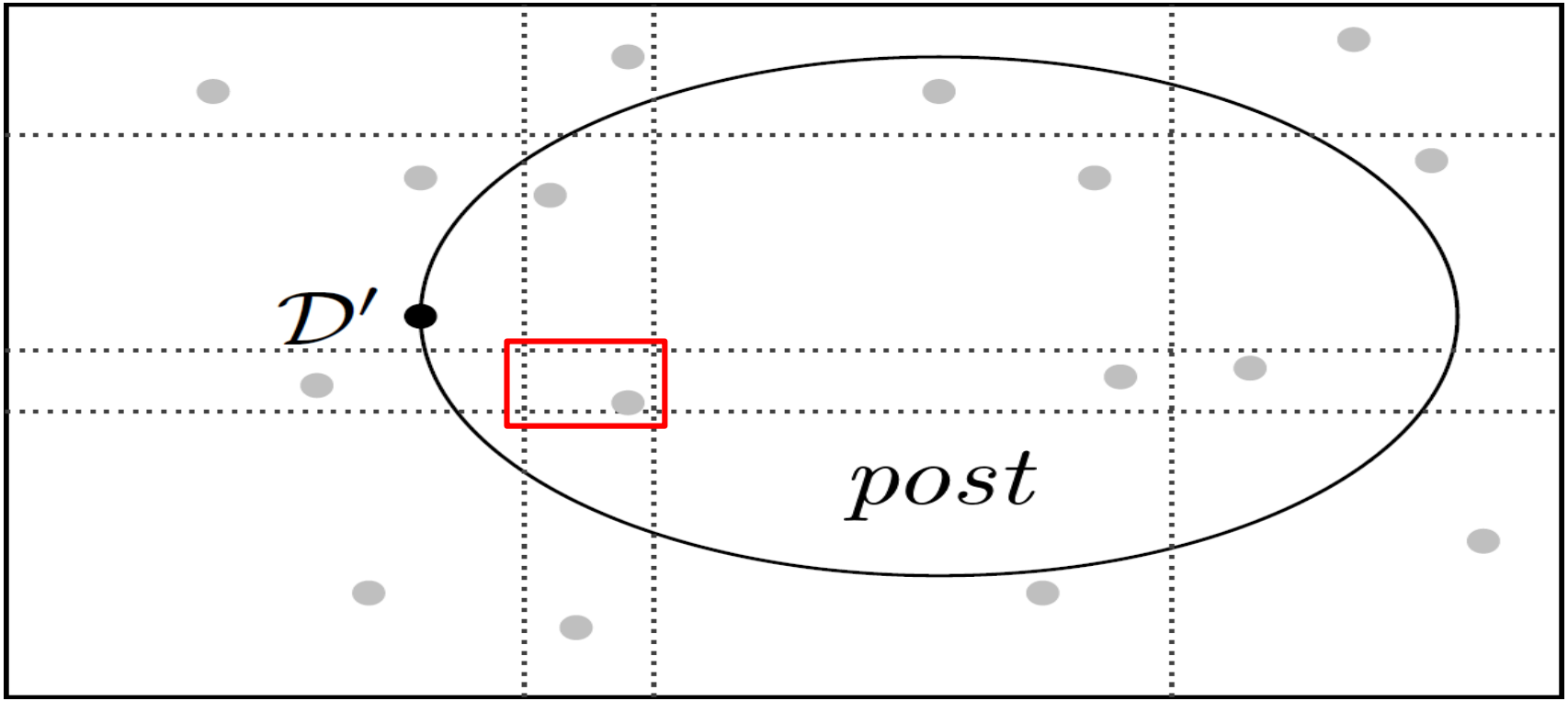
Great complexity bound [Synthesis session at CAV19]


$\mathcal{D}'$
$post$


$\mathcal{D}'$
$post$


$\mathcal{D}'$
$post$


$\mathcal{D}'$
$post$


$\mathcal{D}'$
$post$

VC dimension of a set of programs

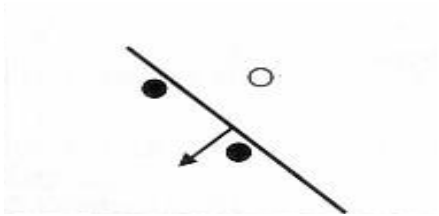


image: V. Kecman, 2001

VC dimension of a set of programs

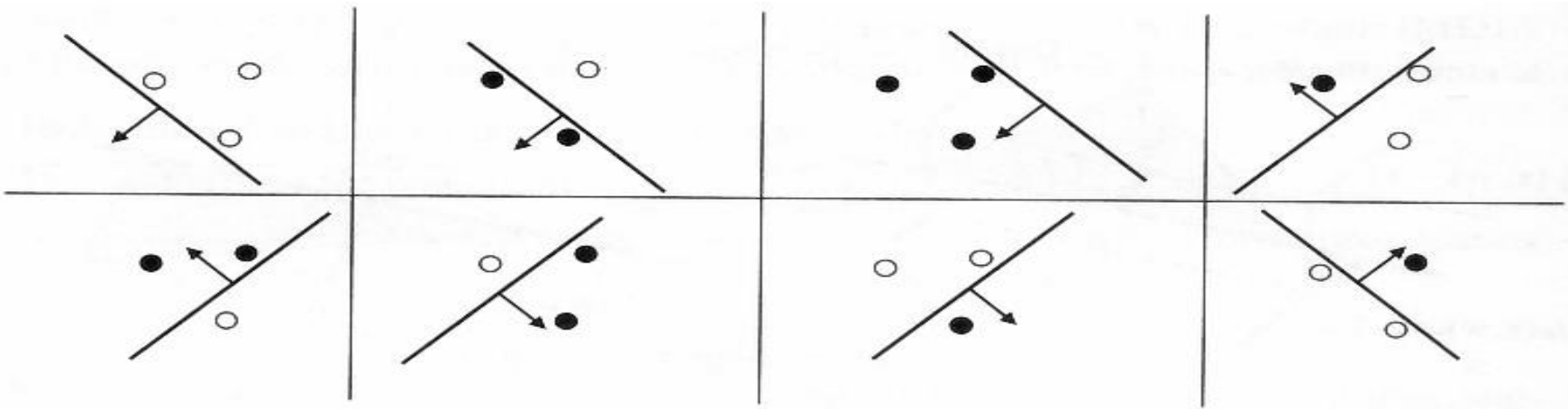


image: V. Kecman, 2001

VC dimension of a set of programs

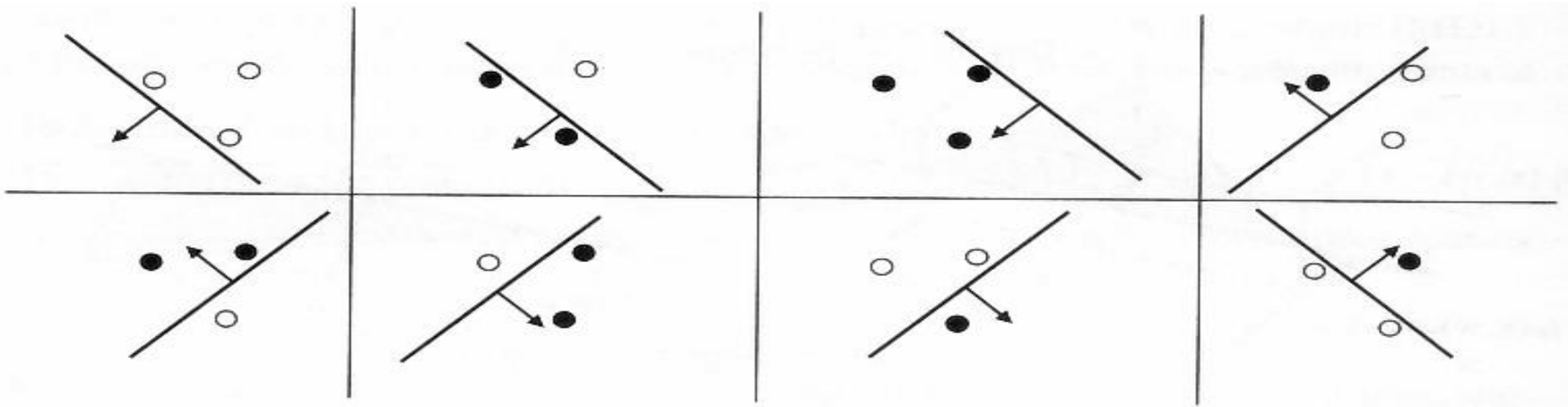
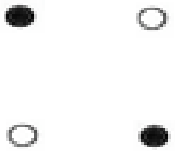


image: V. Kecman, 2001



No line can separate these four points

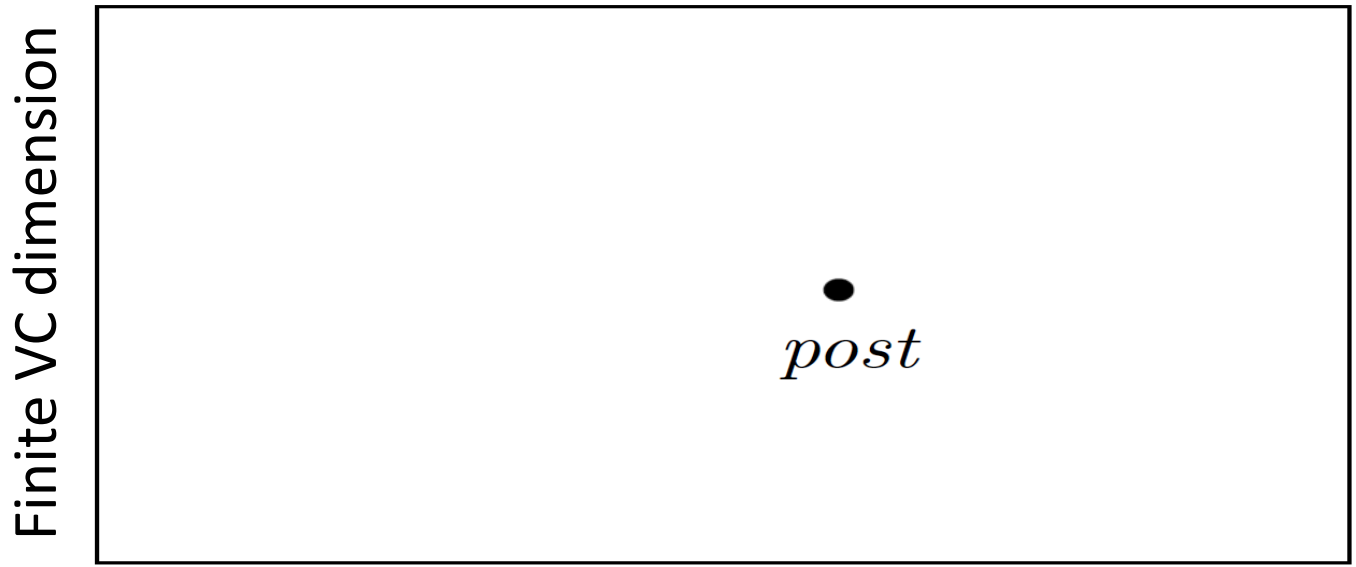
```
fun hireRep(min,urank)
    dec = ●1 <= urank <= ●2
    return dec
```

$$\varphi = ((H_1 \leq urank \leq H_2) == dec)$$

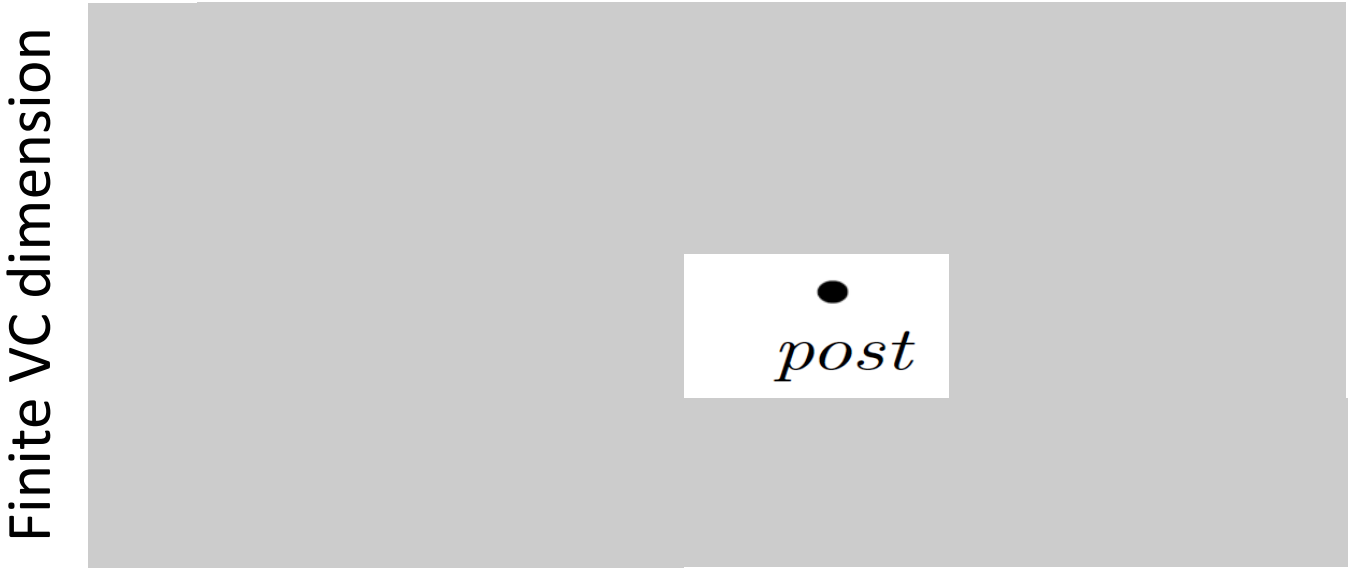
VC Dimension is 2

urank = 2	true
urank = 10	false
urank = 20	true

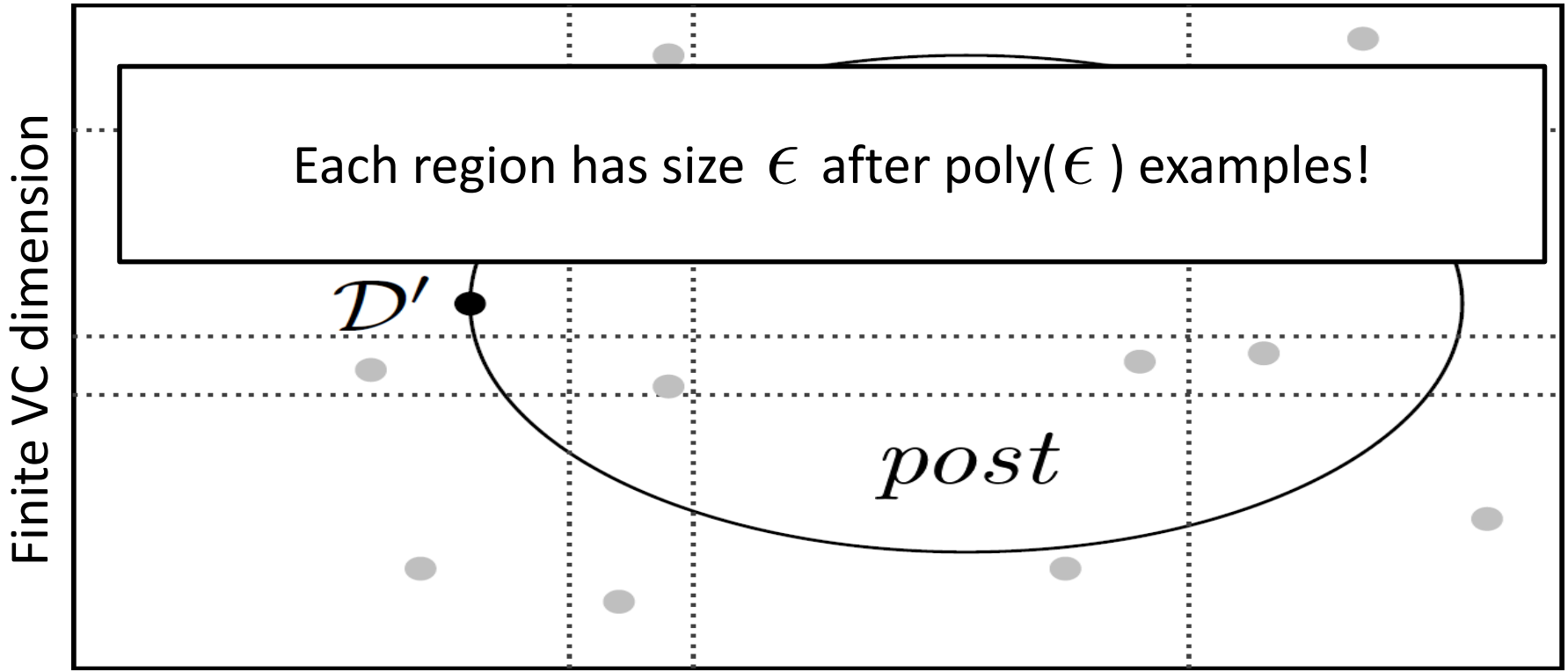
Remember PAC learning?



Remember PAC learning?



The white square has size ϵ after poly(ϵ) examples


Finite VC dimension
Each region has size ϵ after poly(ϵ) examples!
$\mathcal{D}'$
$post$

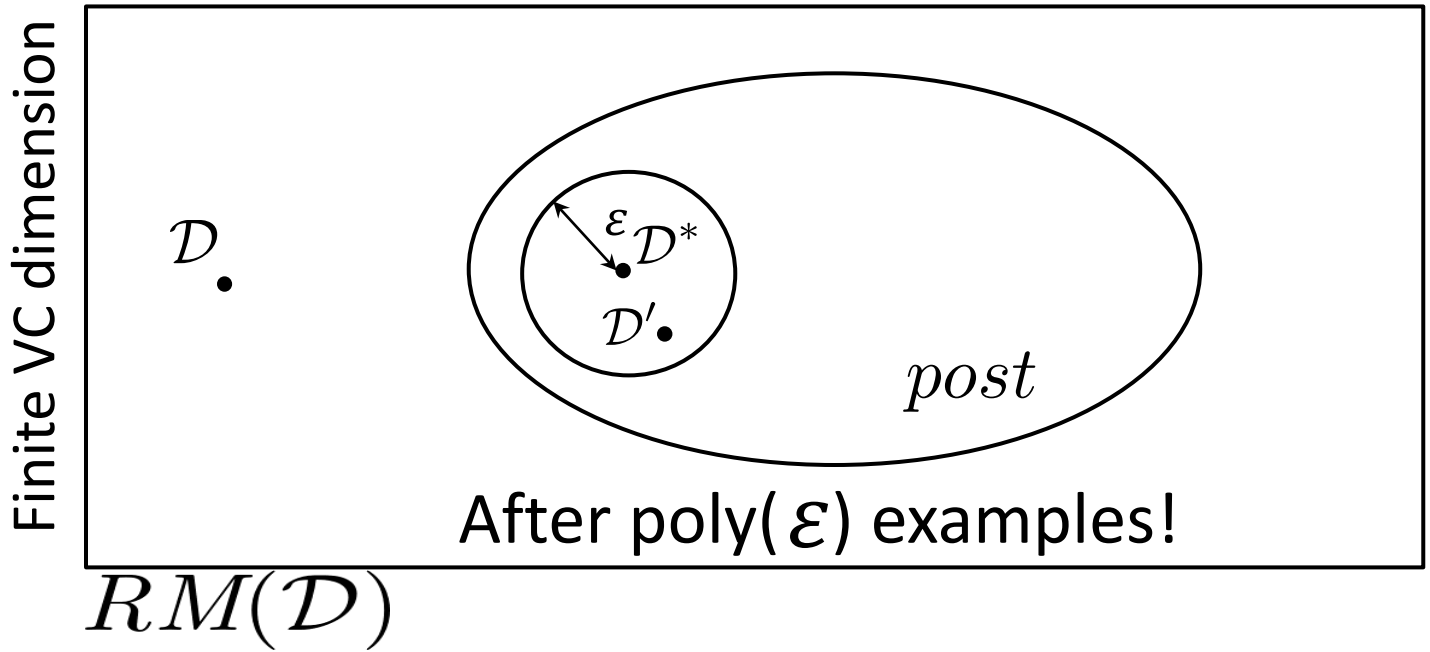
What is so different about program synthesis?

Inductive learning	Inductive synthesis
Certain types of functions	~~Search space is a mess~~
Can be learned approximately	~~We can't be sort of right~~
Using polynomially many IO examples	~~We don't have labeled examples~~

How long will my synthesizer run for?

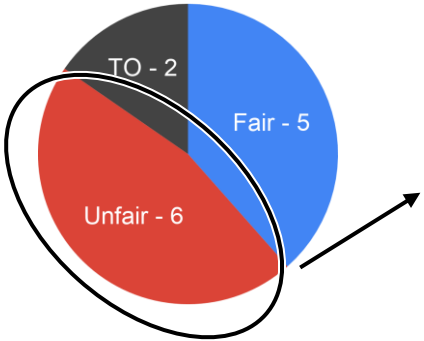
To answer this question, we need to address these problems

Theorem: Convergence of Digits



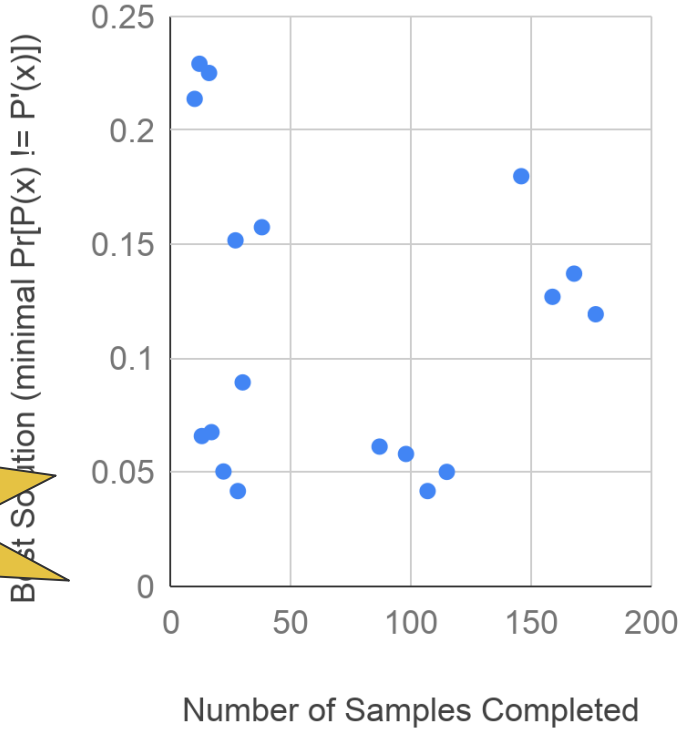
DOES IT WORK?

DIGITS Case Study



Manually replace some constants with "holes":
18 Repair Problems

All Repaired!



Distribution-Guided InducTive Synthesis (DIGITS)

I'm going to present a stupid algorithm

Show that it actually works

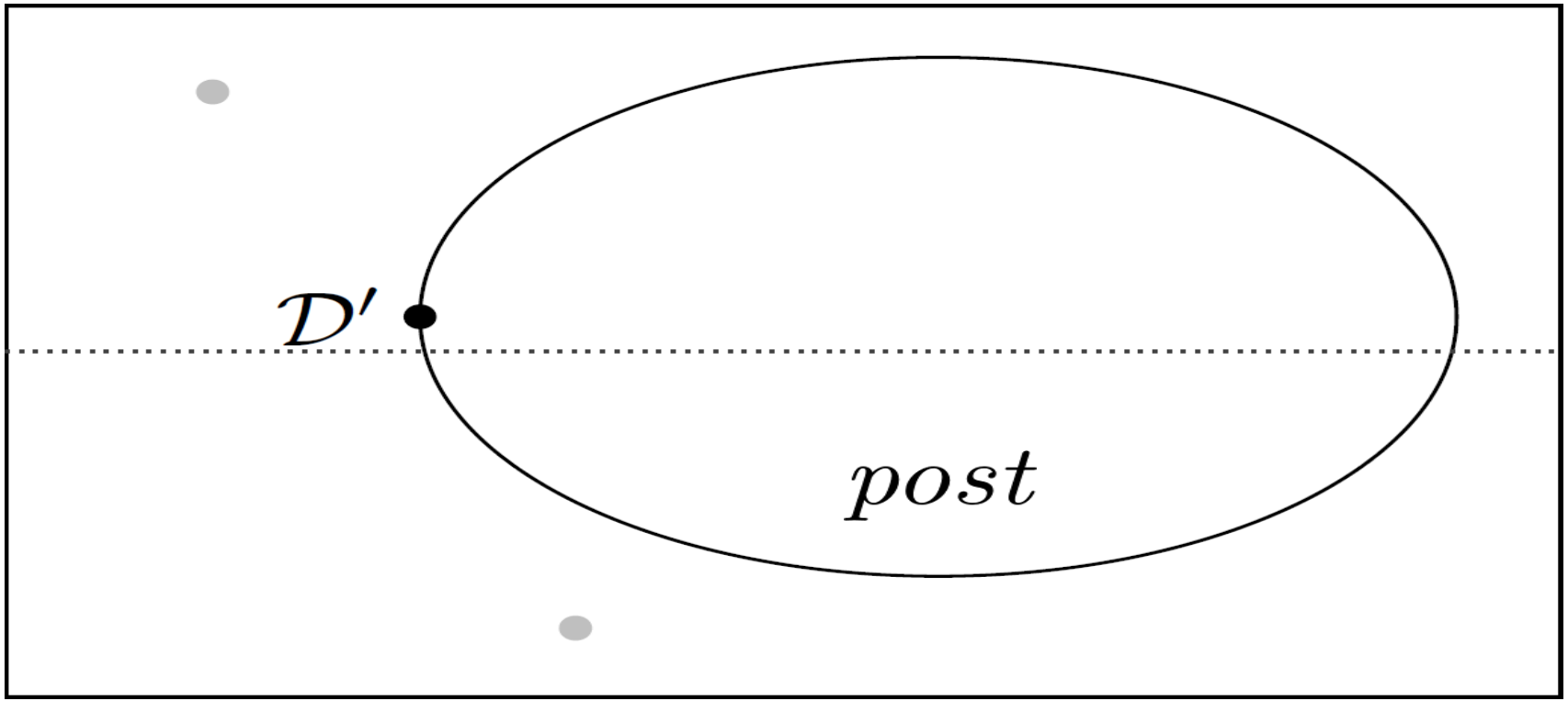
Inspired by the following

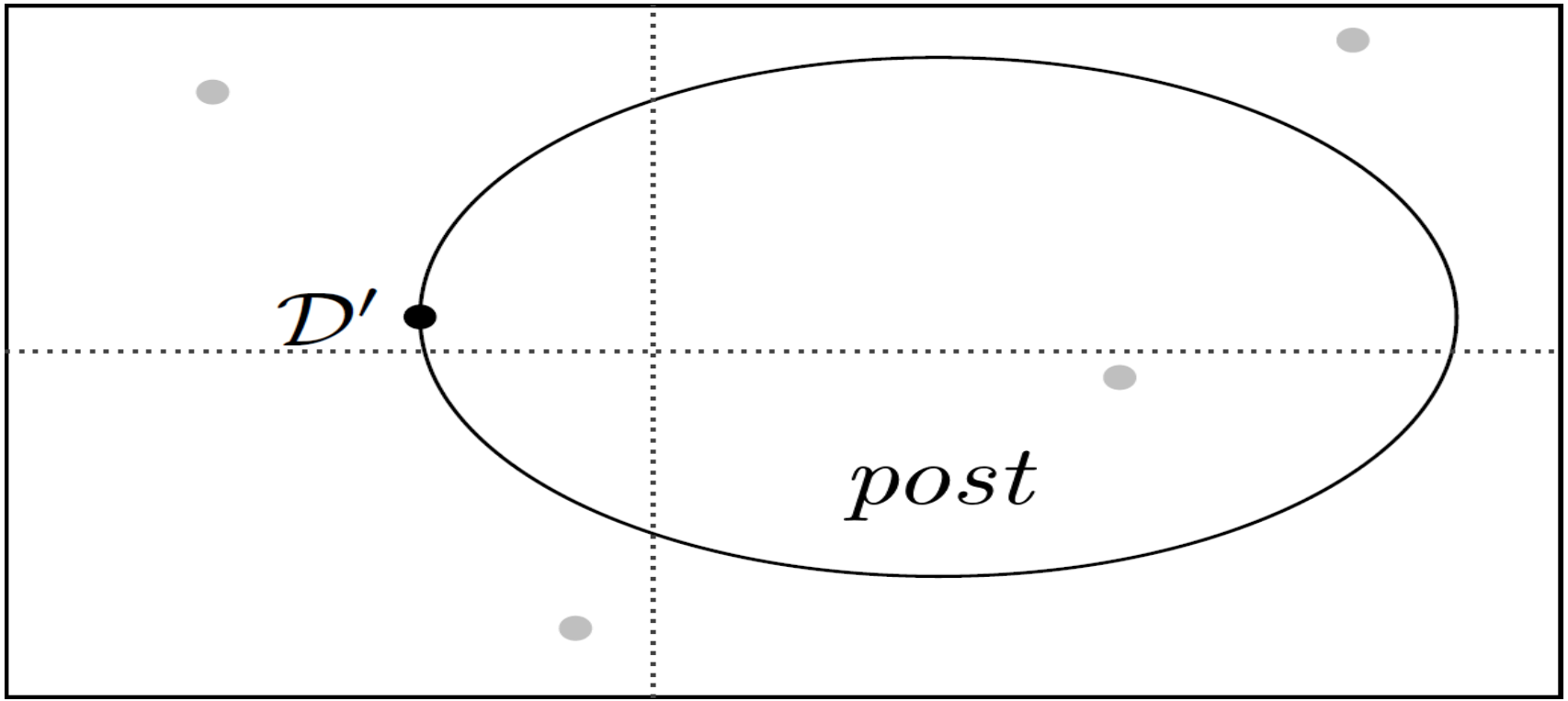
It works -> It ain't stupid

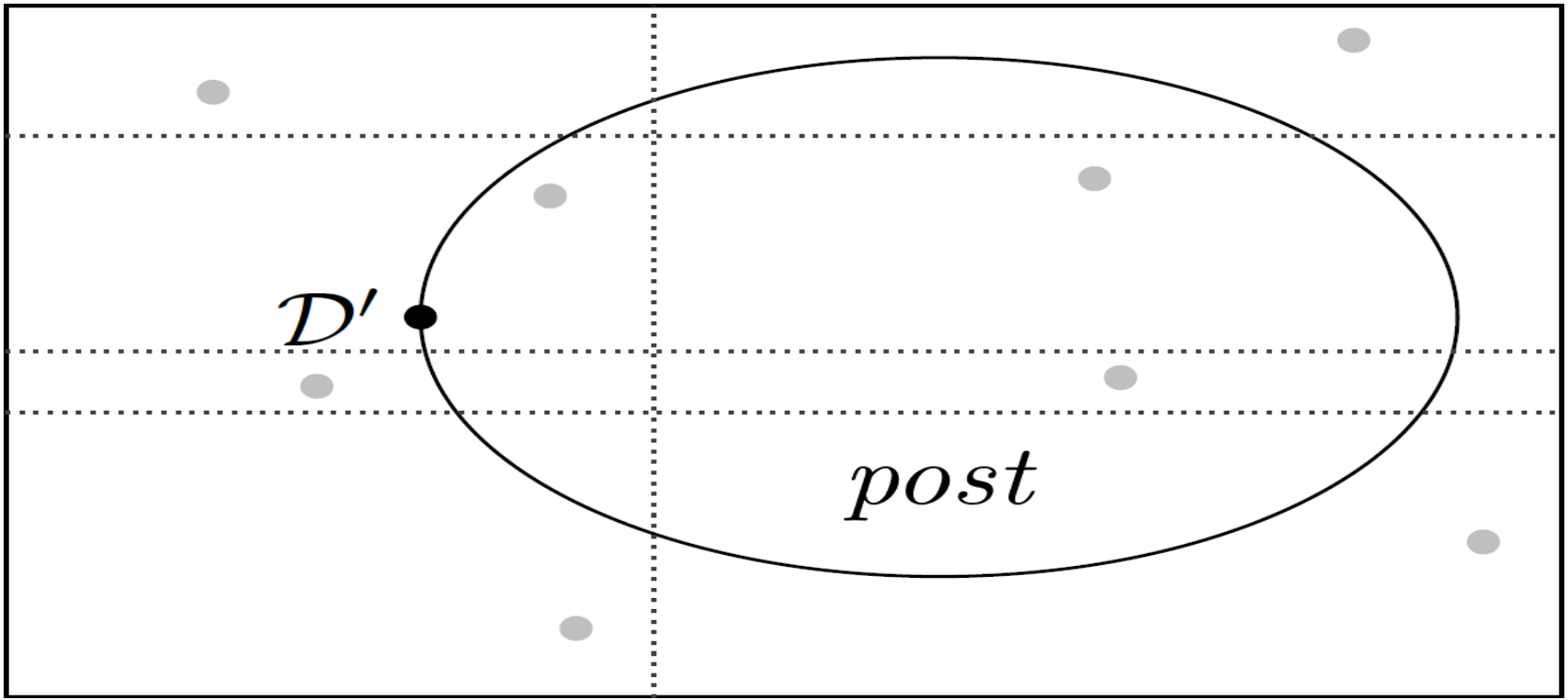
I prove that it isn't that stupid and terminates using few examples [CAV17]

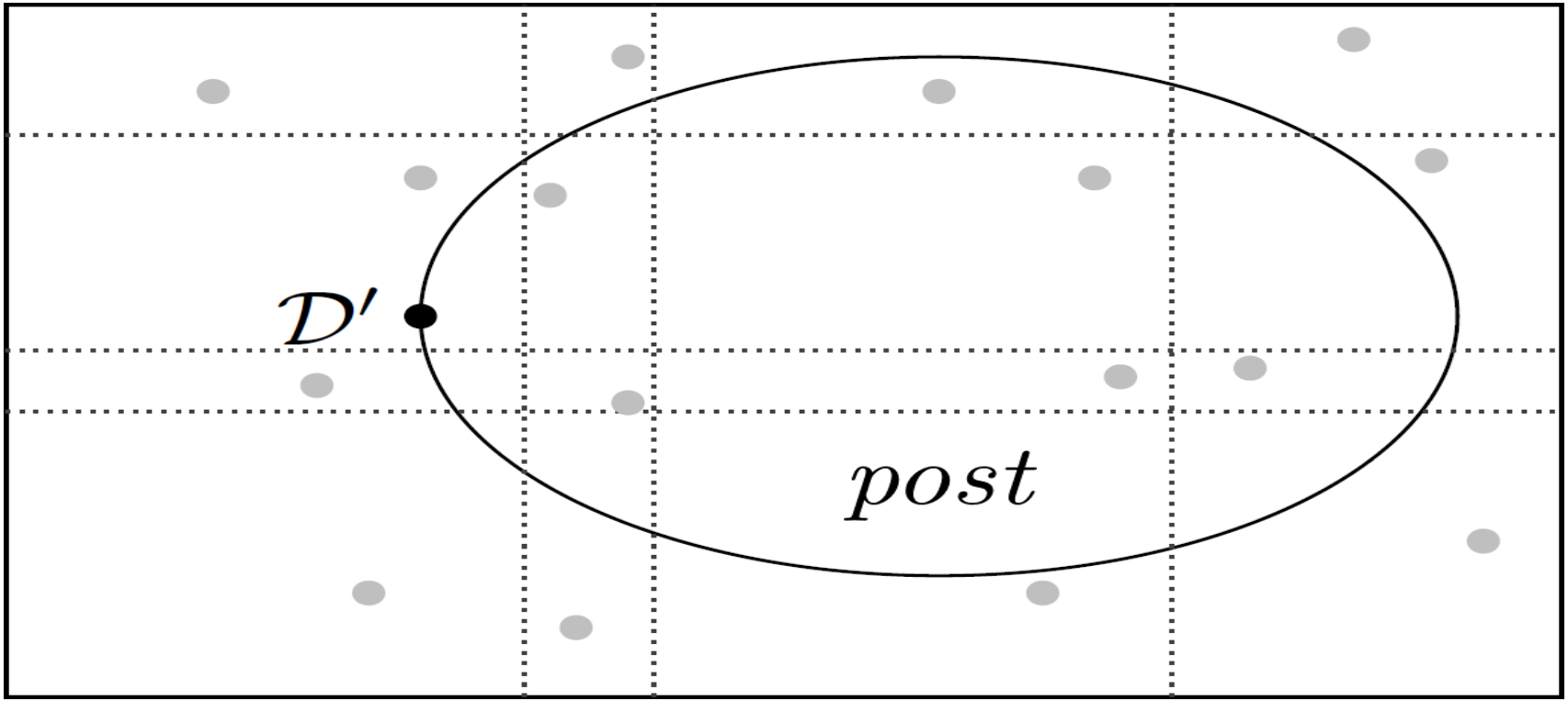
Tuesday: Sam will show you that, not only it's not that stupid, it's actually pretty smart!

Great complexity bound [Synthesis session at CAV19]

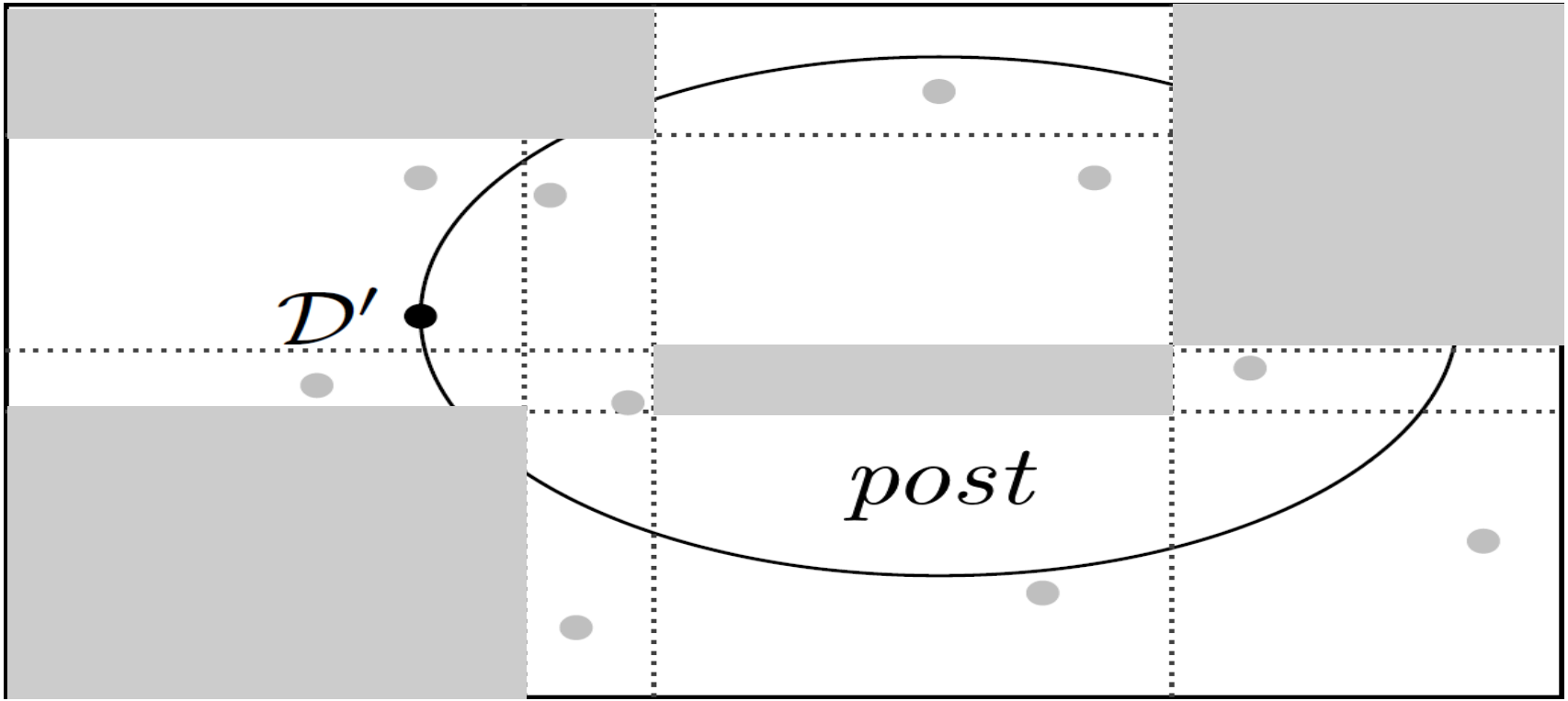

$\mathcal{D}'$
$post$


$\mathcal{D}'$
$post$

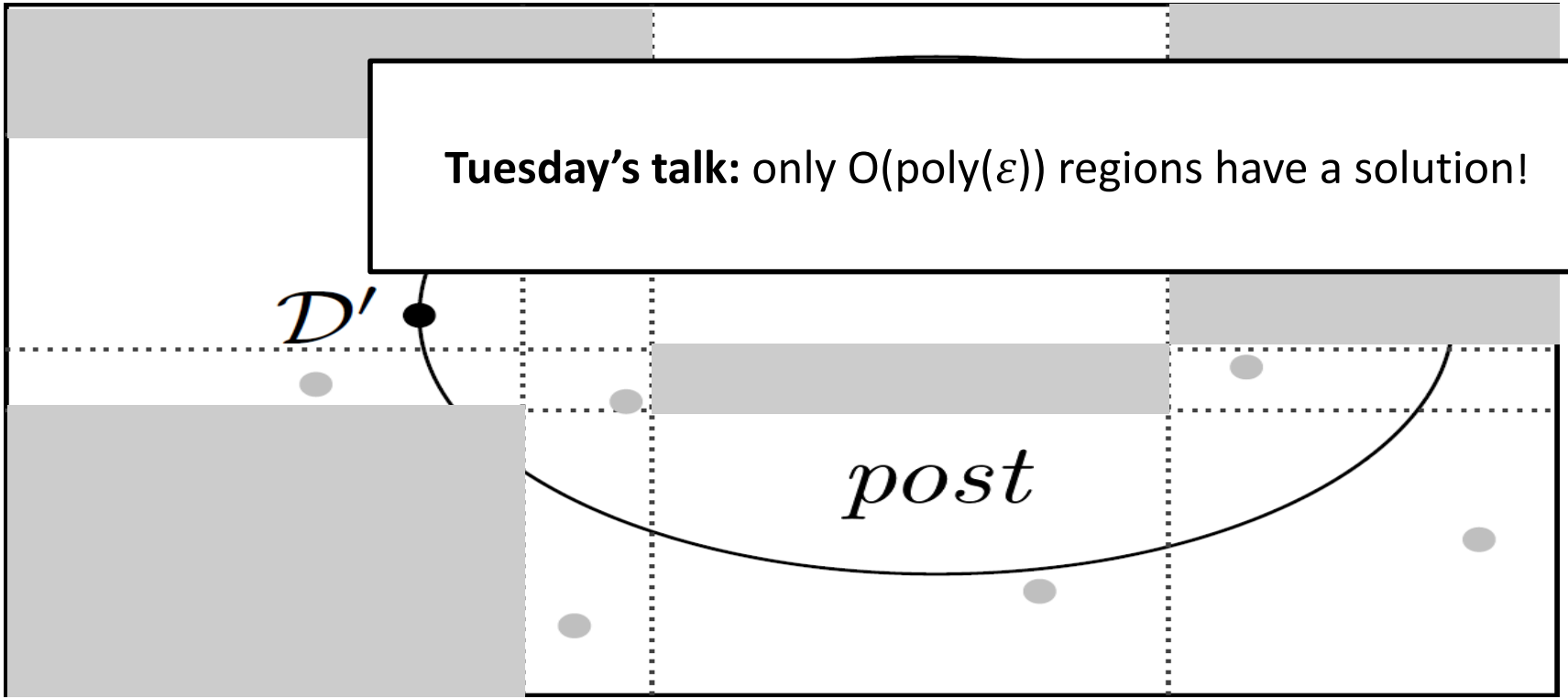

$\mathcal{D}'$
$post$



Number of dots in this plot is exponential in the number of examples

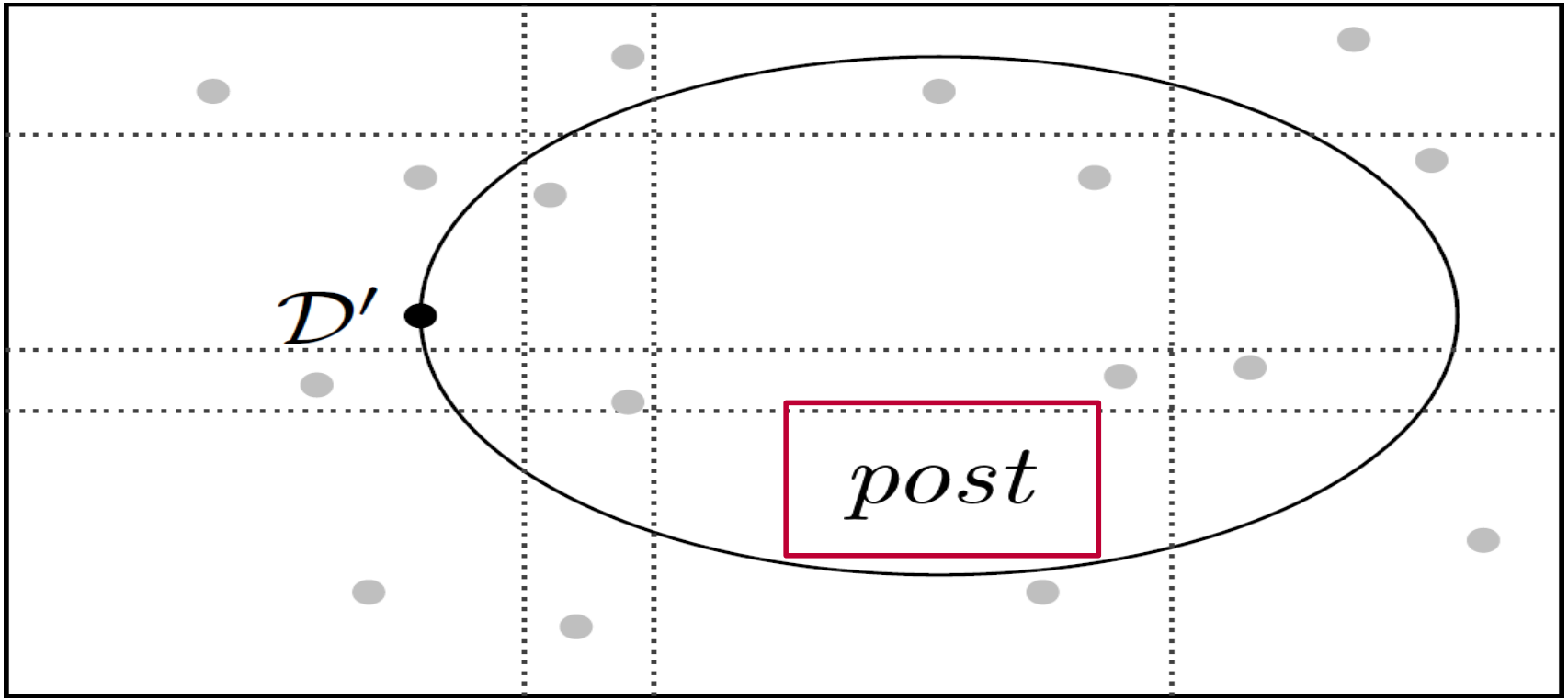


But maybe not all these regions admit a solution…



But maybe not all these regions admit a solution…

Can we speed up the search? [Sam on Tuesday]



Can we handle more expressive programs?

$$\{v \sim \mathcal{M}\}$$

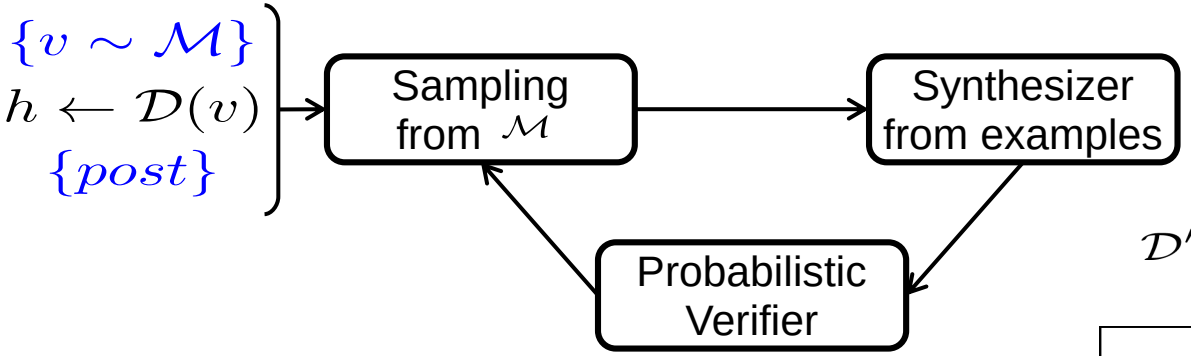
$$h \leftarrow \mathcal{D}(v)$$

→ Decision making program: returns true or false

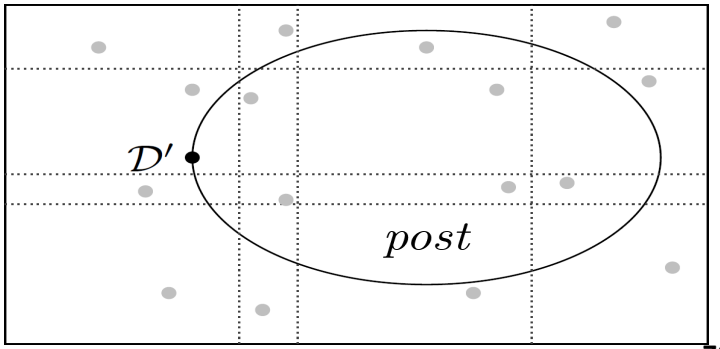
$$\left\{ \frac{\Pr[h \mid v_s]}{\Pr[h \mid \neg v_s]} > 1 - \epsilon \right\}$$

What about programs that compute values?

CONCLUSION


$\{v \sim \mathcal{M}\}$
$h \leftarrow \mathcal{D}(v)$
$\{post\}$
Sampling from $\mathcal{M}$
Synthesizer from examples
Probabilistic Verifier

$\mathcal{D}'$


$\mathcal{D}'$
$post$