Lecture 16
Synthesis for Inductive Learning

Nadia Polikarpova

Motivation

Traditional Machine Learning

- Learn a function from a set of examples
- Optimization-based search (fast)
- Models:
  - too inexpressive (linear function / BDD): limited to simple problems
  - too expressive (NN): uninterpretable and data-hungry

Inductive Synthesis

- Learn a function from a set of examples
- Combinatorial search (slow)
- Models are programs from a DSL:
  - as expressive as we like!

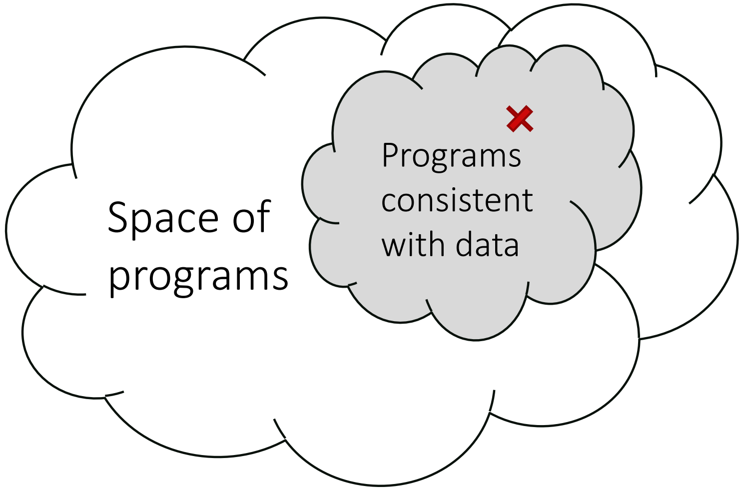
We want to model complex phenomena
but also understand, verify, extrapolate the model,
and learn from few data-points?

Program Induction

```
L ::= sort(L)  |  L[N..N]
    |  L + L  |  [N]  |  x
N ::= find(L,N)  |  0
```

$$\underset{p \in DSL}{\operatorname{argmax}}\, 1[p \text{ satisfies } D]$$

```
[0] → [0]
[5,1] → [1,5,0]
```



Need to find the most likely program given the data!

Bayes Rule

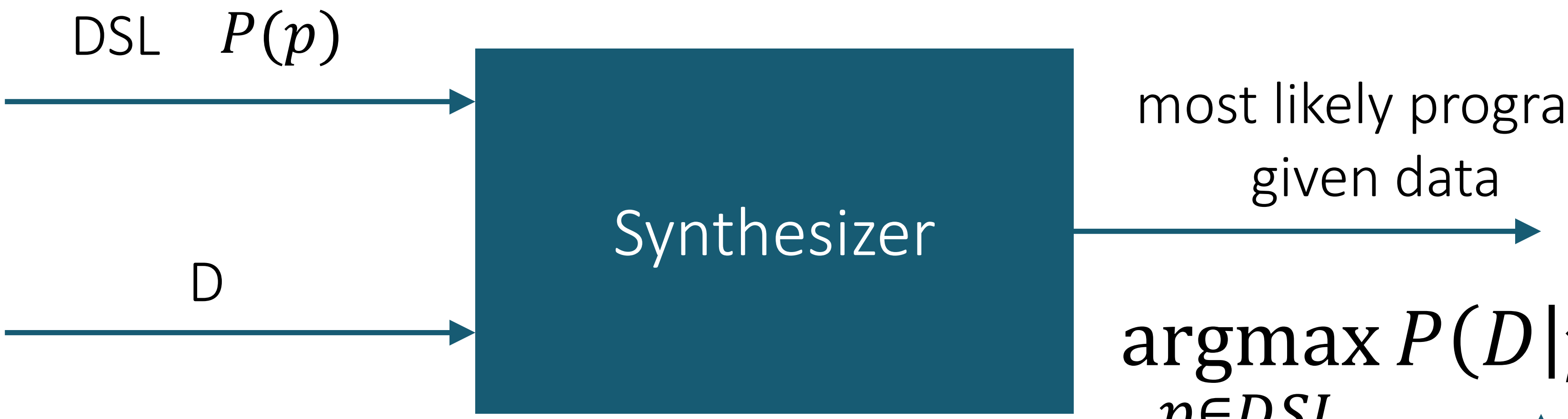
Likelihood of data given program

Prior on programs

$$P(p|D) = \frac{P(D|p)P(p)}{P(D)}$$

Does not depend on the program: ignore

Bayesian Program Induction



size
PCFG
PHOG

$$\underset{p \in DSL}{\operatorname{argmax}} P(D|p)P(p)$$

$$1[p \text{ satisfies } D]$$

or more complex in the presence of noise,
strong sampling assumptions, etc

(Constrained) optimization

$$\text{prog}(D) = \underset{p \in DSL}{\operatorname{argmin}}\, cost(p) + error(p, D)$$

How do we solve this constraint?

Solving constrained optimization

$$\text{prog}(D) = \underset{p \in DSL}{\text{argmin}}\, cost(p) + error(p, D)$$

Enumerative a-la Euphony

- enumerate in order of $cost(p)$, check if $error(p, D)$ is 0 or ∞

Iterative SAT/SMT

- solve $\exists p.\, cost(p) + error(p, D) \leq C$
- if solution found, decrease C; otherwise increase C

MaxSAT / MaxSMT

- SAT/SMT + optimization objective

Examples

Graphics programs

Linguistics

Control

Examples

Graphics programs

- Ellis, Ritchie, Solar-Lezama, Tenenbaum: Learning to Infer Graphics Programs from Hand-Drawn Images. NeurIPS’18
- Tian, Luo, Sun, Ellis, Freeman, Tenenbaum, Wu: Learning to Infer and Execute 3D Shape Programs. ICLR'19

Linguistics

Control

Graphics programs: contributions

Cool application!

New combination of neural nets and program synthesis

Probability-guided constraint-based synthesis

Learning a search policy to guide synthesis

Graphics programs: limitations

Does not scale to many shapes and programs of depth > 3

How do you come up with search policy parameters?

NN only works for carefully drawn images

Graphics programs: questions

Behavioral constraints? Structural constraints? Search strategy?

- Sets of graphics primitives (generated from hand-drawn images)
- DSL (Table 2)
- Constraint based + Levin search

Levin search vs Euphony

- Probabilistic grammar vs search policy
- Used differently to guide search

Examples

Graphics programs

Linguistics

- Ellis, Solar-Lezama, Tenenbaum: Unsupervised Learning by Program Synthesis. NeurIPS’15
- Barke, Kunkel, Polikarpova, Meinhardt, Bakovic, Bergen: Constraint-based Learning of Phonological Processes EMNLP'19

Control

English verbs PAST Tense

zip is phonetically [zɪp]
beg is phonetically [bɛg]

[zɪpt] (zipped)

[bɛgd] (begged)

English verbs PAST Tense

zip is phonetically [zɪp]
beg is phonetically [bɛg]

/zɪp + d/ → [zɪpt]

(zipped)

/bɛg + d/ → [bɛgd]

(begged)

English verbs PAST Tense

zip is phonetically [zɪp]
beg is phonetically [bɛg]

/zɪp + d/ → [zɪpt]

/bɛg + d/ → [bɛgd]

/d/ → [t] if it occurs after voiceless sounds

RESEARCH PROBLEM

zip is phonetically [zɪp]
beg is phonetically [bɛg]

/zɪp + d/ → [zɪpt]

/bɛg + d/ → [bɛgd]

Automatic Inference of Phonological rules

/d/ → [t] if it occurs after voiceless sounds

WORD FORMS

/zɪp + d/ → [zɪpt]

/bɛg + d/ → [bɛgd]

/stem + suffix/ → [surface form]

WORD FORMS

/zɪp + d/ → [zɪpt]

/bɛg + d/ → [bɛgd]

/stem + suffix/ → [surface form]

WORD FORMS

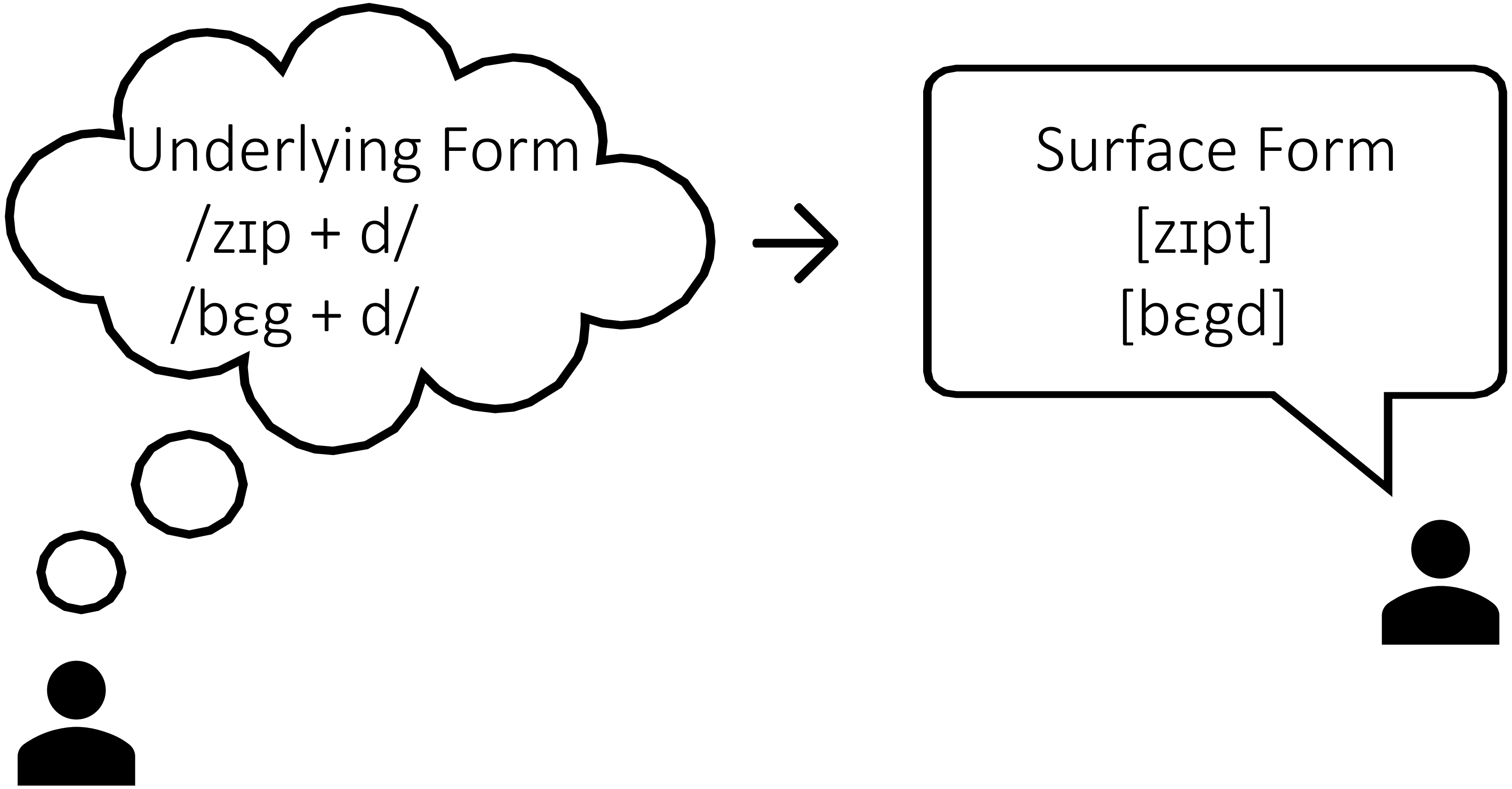
/zɪp + d/ → [zɪpt]

/bɛg + d/ → [bɛgd]

/underlying form/ → [surface form]

PHONOLOGICAL PROCESS

Goal – Infer function from the underlying form to surface form.



Phonological rewrite rules

$$A \rightarrow B \,/\, L \_ R$$

Any sound that matches A and occurs between sounds that match left context L and right context R will be rewritten to B.

Phonological rewrite rules

Surface forms	A → B / L _ R
[zɪpt]	/d/ → [t] / [p] _ ø
[bɛgd]	No change
[zɪps]	/z/ → [s] / [p] _ ø
[bɛgz]	No change

Phonological rewrite rules

Surface forms	A → B / L _ R
[zɪpt]	/d/ → [t] / [p] _ ø
[zɪps]	/z/ → [s] / [p] _ ø

/d/, /z/ → [t] , [s] / [p] _ ø

Phonological rewrite rules

Surface forms	A → B / L _ R
[zɪpt]	/d/ → [t] / [p] _ ø
[zɪps]	/z/ → [s] / [p] _ ø

/d/, /z/ → [t] , [s] / [p] _ ø

voiceless

Phonological rewrite rules

Surface forms	A → B / L _ R
[zɪpt]	/d/ → [t] / [p] _ ø
[zɪps]	/z/ → [s] / [p] _ ø

/d/, /z/ → [t] , [s] / [p] _ ø

[-voice]

Phonological rewrite rules

<u>Surface forms</u> A → B / L _ R

[zɪpt] /d/ → [t] / [p] _ ø

[zɪps] /z/ → [s] / [p] _ ø

/d/, /z/ → [t] , [s] / [p] _ ø

[-sonorant] → [-voice]

Phonological rewrite rules

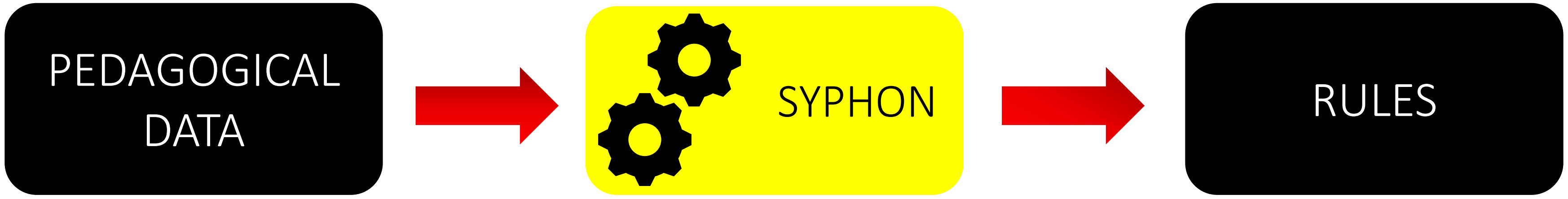
Surface forms	A → B / L _ R
[zɪpt]	/d/ → [t] / [p] _ ø
[zɪps]	/z/ → [s] / [p] _ ø

/d/, /z/ → [t] , [s] / [p] _ ø

[-sonorant] → [-voice] / [-voice] _ ø

SYNTHESIS OF PHONOLOGICAL RULES

Syphon



Syphon

PAST TENSE	PRESENT TENSE
zɪpt	zɪps
bɛgd	bɛgz
rod	roz
lɪvd	lɪvz
æskt	æsks

Syphon

PAST TENSE	PRESENT TENSE
zɪp + d	zɪp + z
bɛg + d	bɛg + z
ro + d	ro + z
lɪv + d	lɪv + z
æsk + d	æsk + z

PAST TENSE	PRESENT TENSE
zɪpt	zɪps
bɛgd	bɛgz
rod	roz
lɪvd	lɪvz
æskt	æsks

Syphon

PEDAGOGICAL DATA → SYPHON → RULES

[-sonorant] → [-voice] / [-voice] _

PAST TENSE	PRESENT TENSE
zɪp + d	zɪp + z
bɛg + d	bɛg + z
ro + d	ro + z
lɪv + d	lɪv + z
æsk + d	æsk + z

PAST TENSE	PRESENT TENSE
zɪpt	zɪps
bɛgd	bɛgz
rod	roz
lɪvd	lɪvz
æskt	æsks

Research goals

Interpretability - Inferred rules should be human readable

Data efficiency – Few shot learning

Interactivity - Inference at interactive speeds

Objective function

$$\mathrm{F}(R, U, X) = \begin{cases} \mathrm{length}(R) + \mathrm{fit}(R, U, X) & \text{if consistent}(R, U, X) \\ \infty & \text{otherwise} \end{cases}$$

R - Rules
U - Underlying forms
X - Surface forms

Objective function

Correctness constraint

$$\mathrm{F}(R, U, X) = \begin{cases} \mathrm{length}(R) + \mathrm{fit}(R, U, X) & \text{if consistent}(R, U, X) \\ \infty & \text{otherwise} \end{cases}$$

R - Rules
U - Underlying forms
X - Surface forms

Objective function

Simplicity constraint

Correctness constraint

$$\mathrm{F}(R, U, X) = \begin{cases} \mathrm{length}(R) + \mathrm{fit}(R, U, X) & \text{if consistent}(R, U, X) \\ \infty & \text{otherwise} \end{cases}$$

R - Rules
U - Underlying forms
X - Surface forms

Objective function

Simplicity constraint

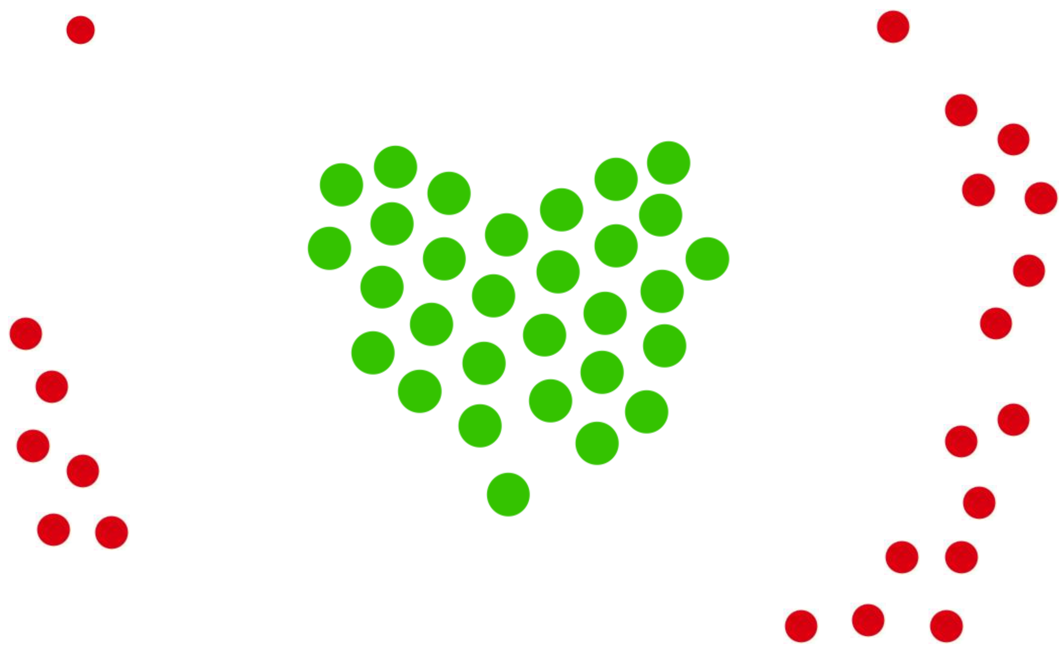
Correctness constraint

$$F(R, U, X) = \begin{cases} \text{length}(R) + \text{fit}(R, U, X) & \text{if consistent}(R, U, X) \\ \infty & \text{otherwise} \end{cases}$$

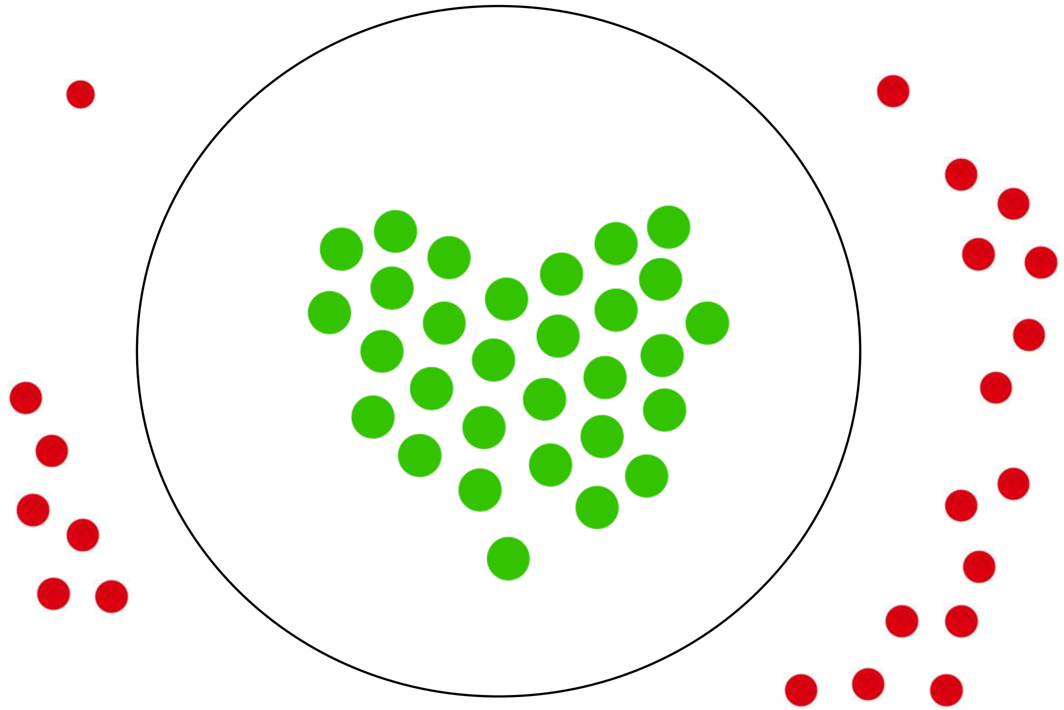
Specificity constraint

R - Rules
U - Underlying forms
X - Surface forms

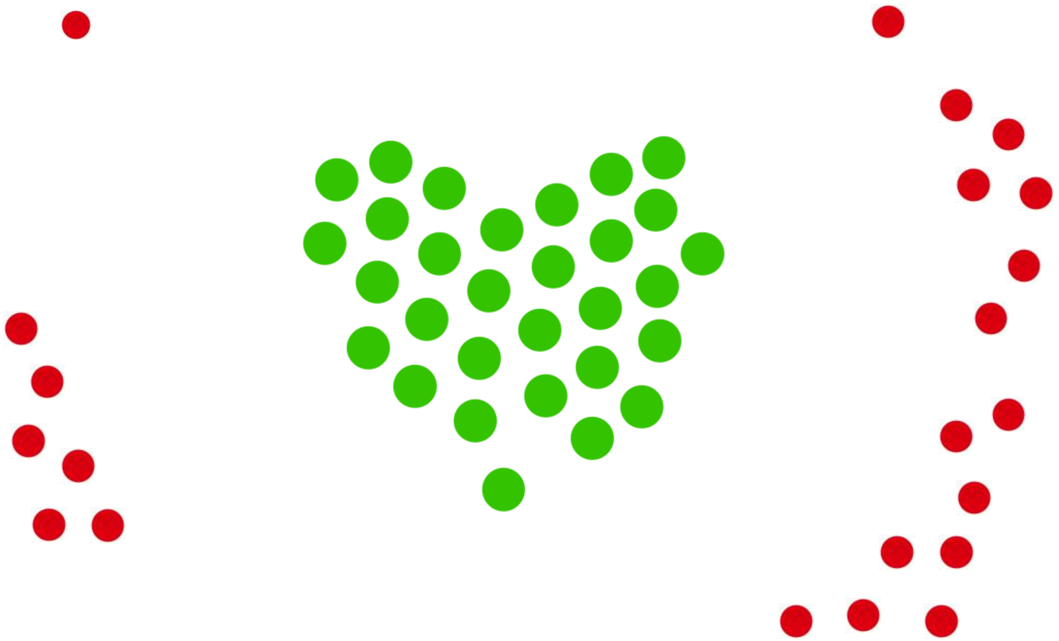
Objective function SIMPLICITY



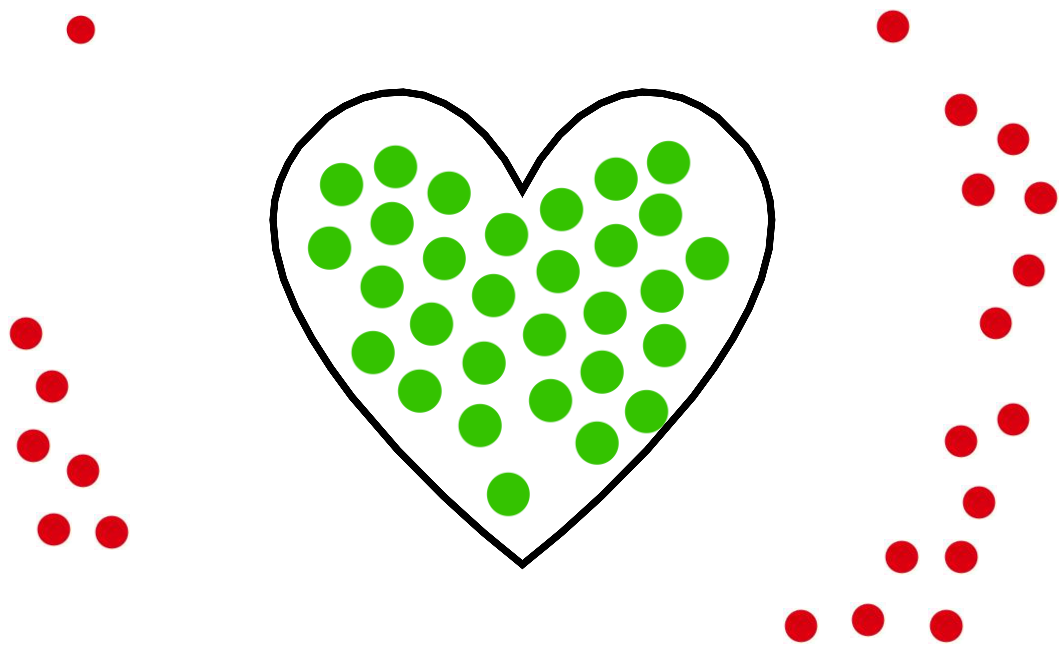
Objective function SIMPLICITY



Objective function specificity



Objective function specificity



Constraint based program synthesis

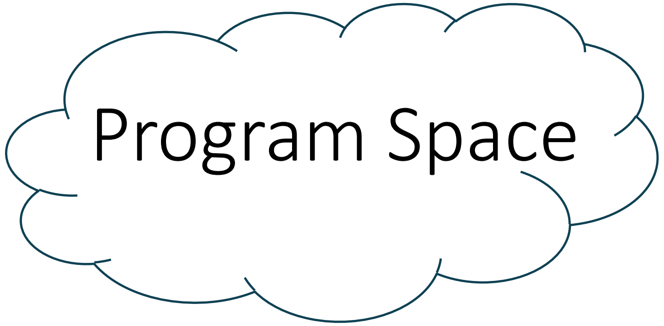
Represent rule $A \rightarrow B$ / L _ R as a program

Program Space

Constraint based program synthesis

Represent rule A → B / L _ R as a program

$$\mathrm{F}(R, U, X)$$



Constraint based program synthesis

Represent rule A → B / L _ R as a program

$$\mathrm{F}(R, U, X)$$

Program Space

Consistent program

Constraint based program synthesis

Represent rule A → B / L _ R as a program

$$\mathrm{F}(R, U, X)$$

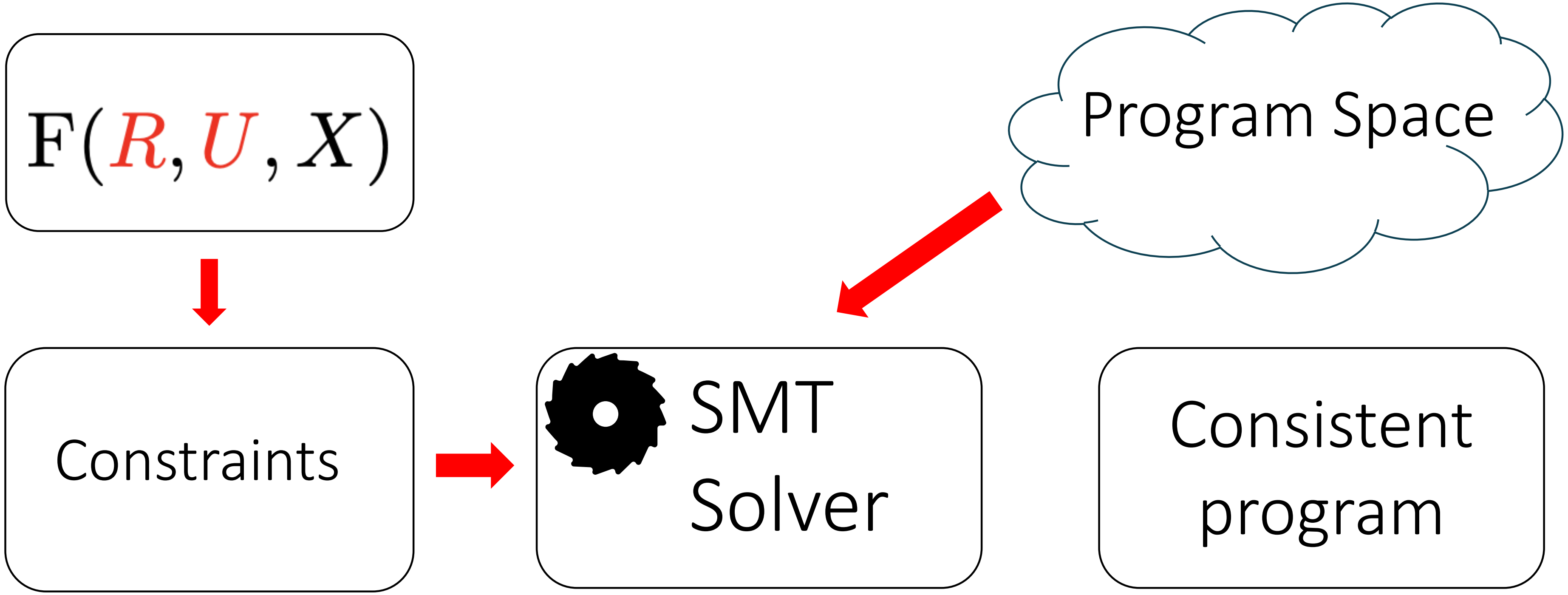
Constraints

Program Space

Consistent program

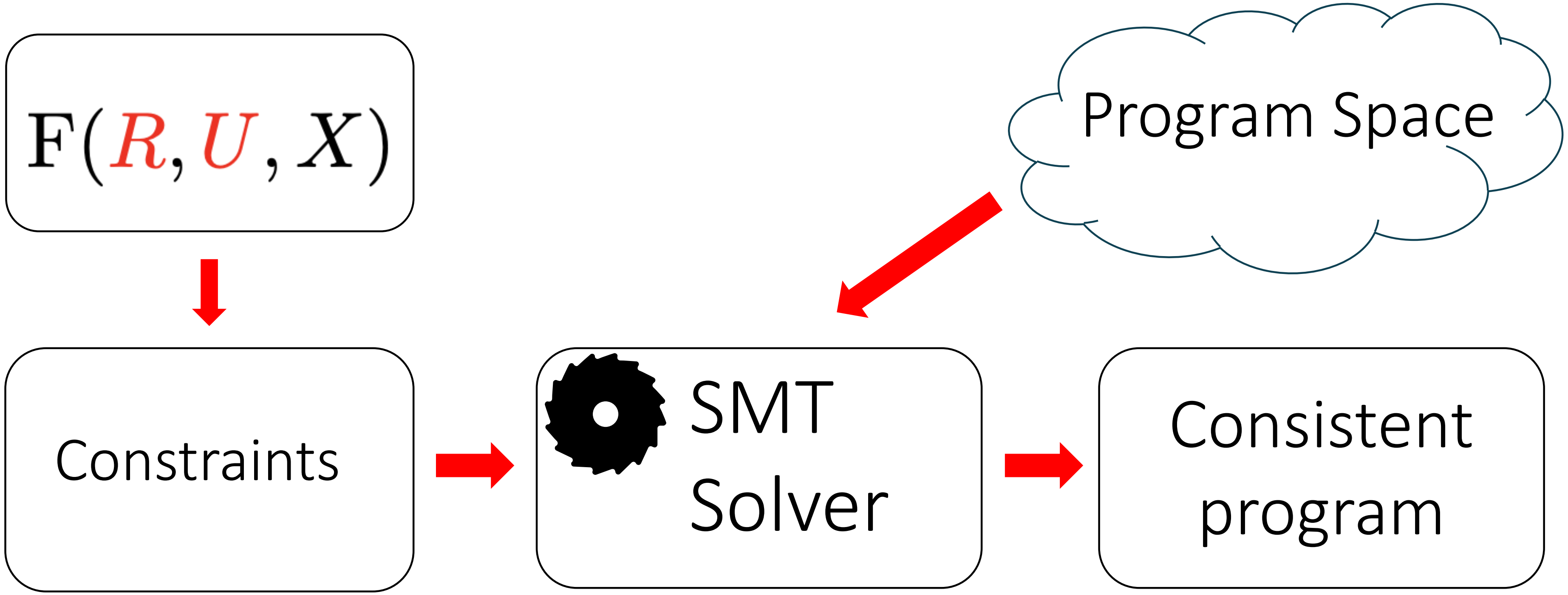
Constraint based program synthesis

Represent rule A → B / L _ R as a program

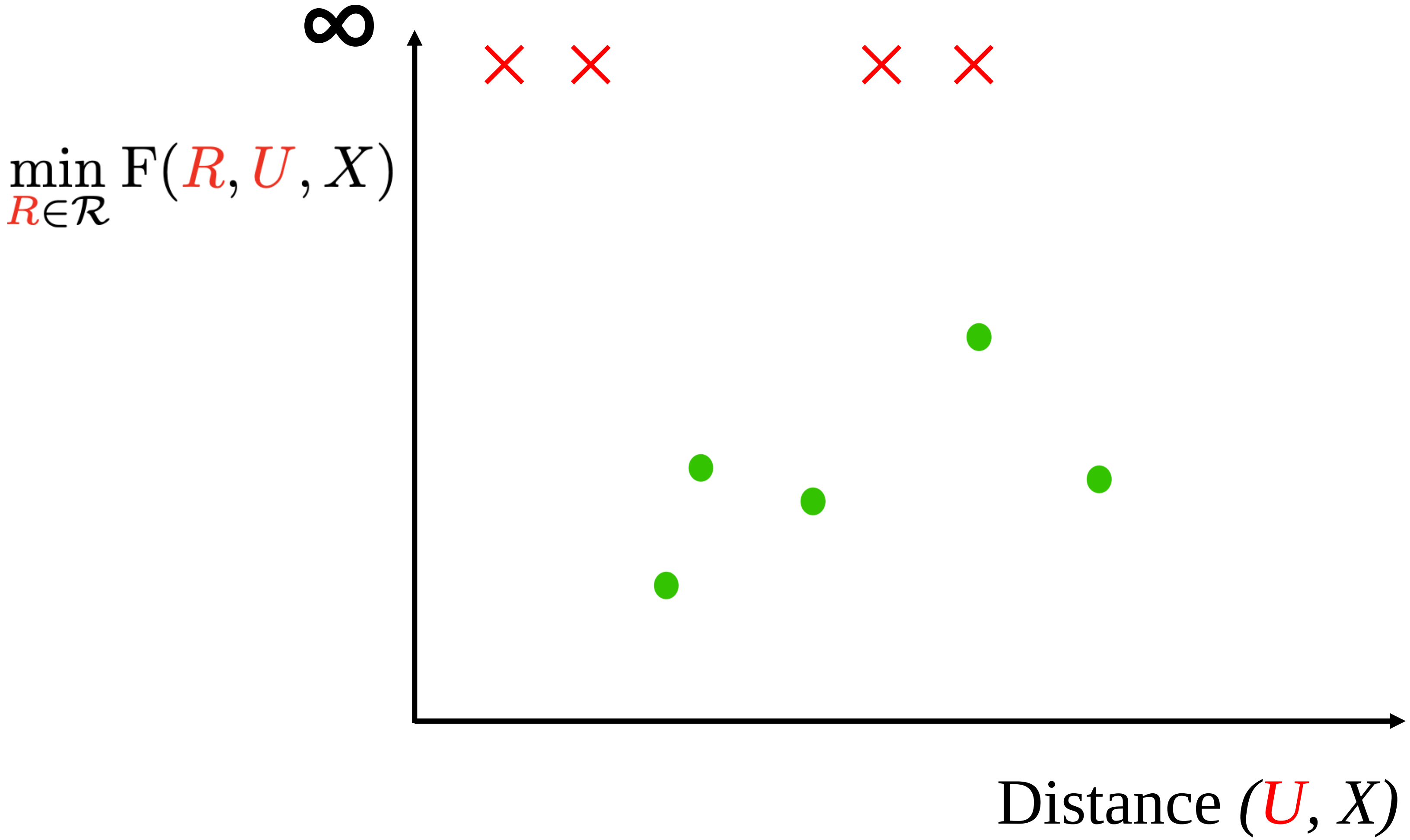


Constraint based program synthesis

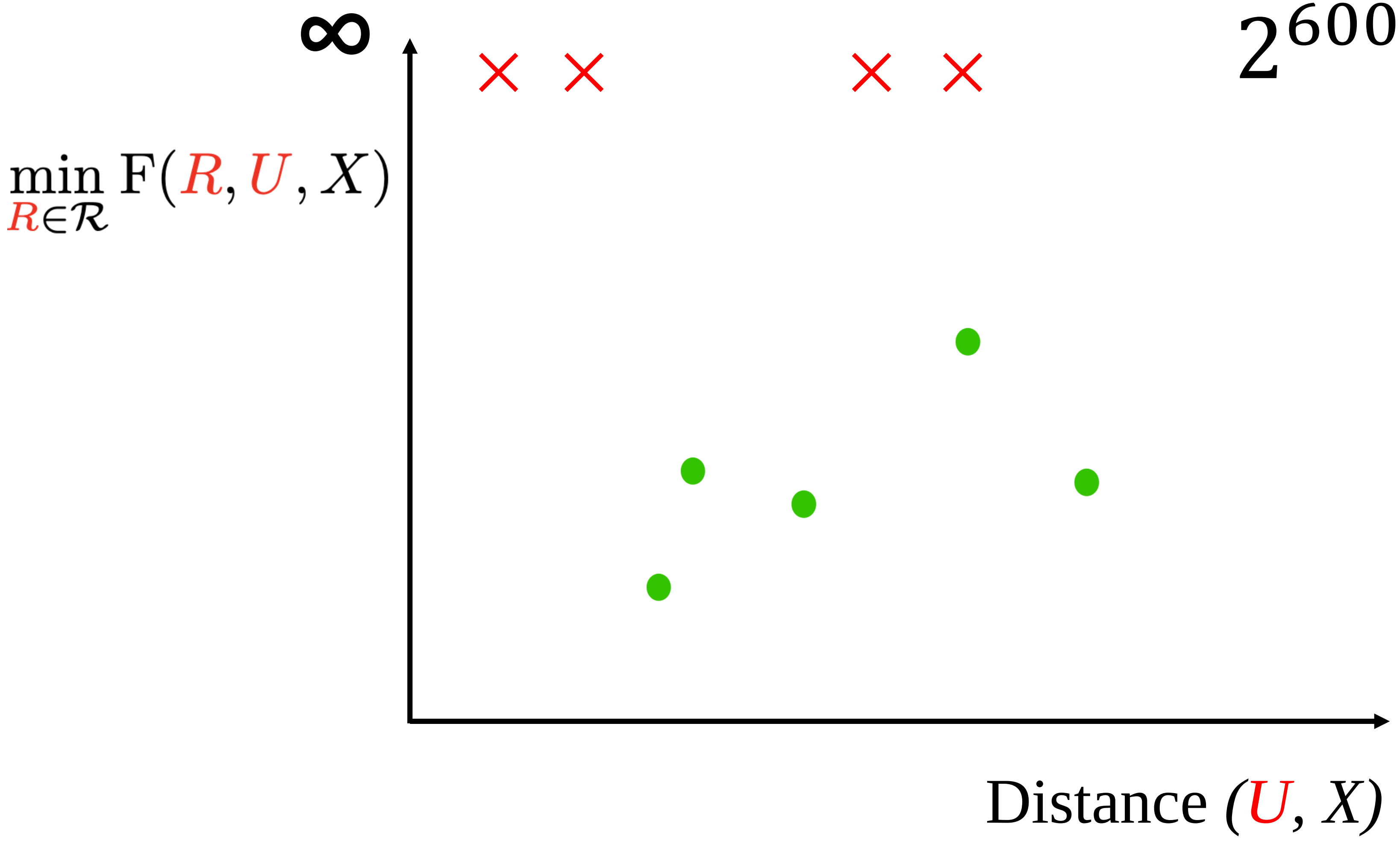
Represent rule A → B / L _ R as a program

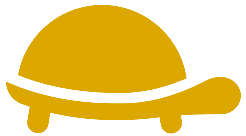


HYPOTHESIS SPACE



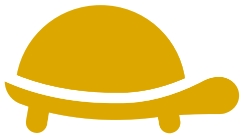
HYPOTHESIS SPACE



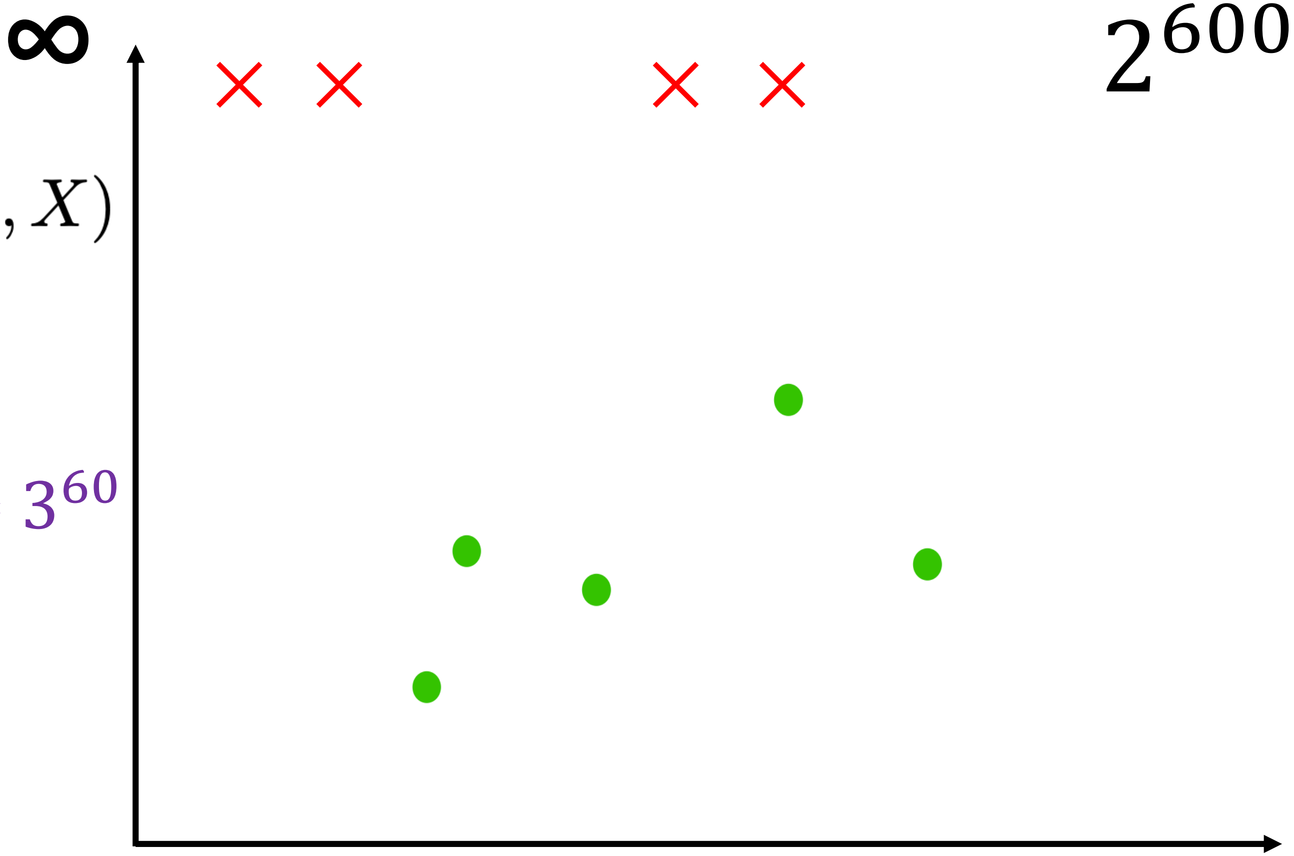


2^{600}

HYPOTHESIS SPACE



2^{600}



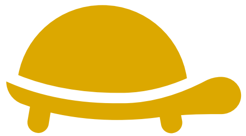
$$\min_{R \in \mathcal{R}} \mathrm{F}(R, U, X)$$

$$|R| = 3^{180} * 3^{60}$$

$$|U| = 90^{34}$$

Distance (U, X)

Global baseline

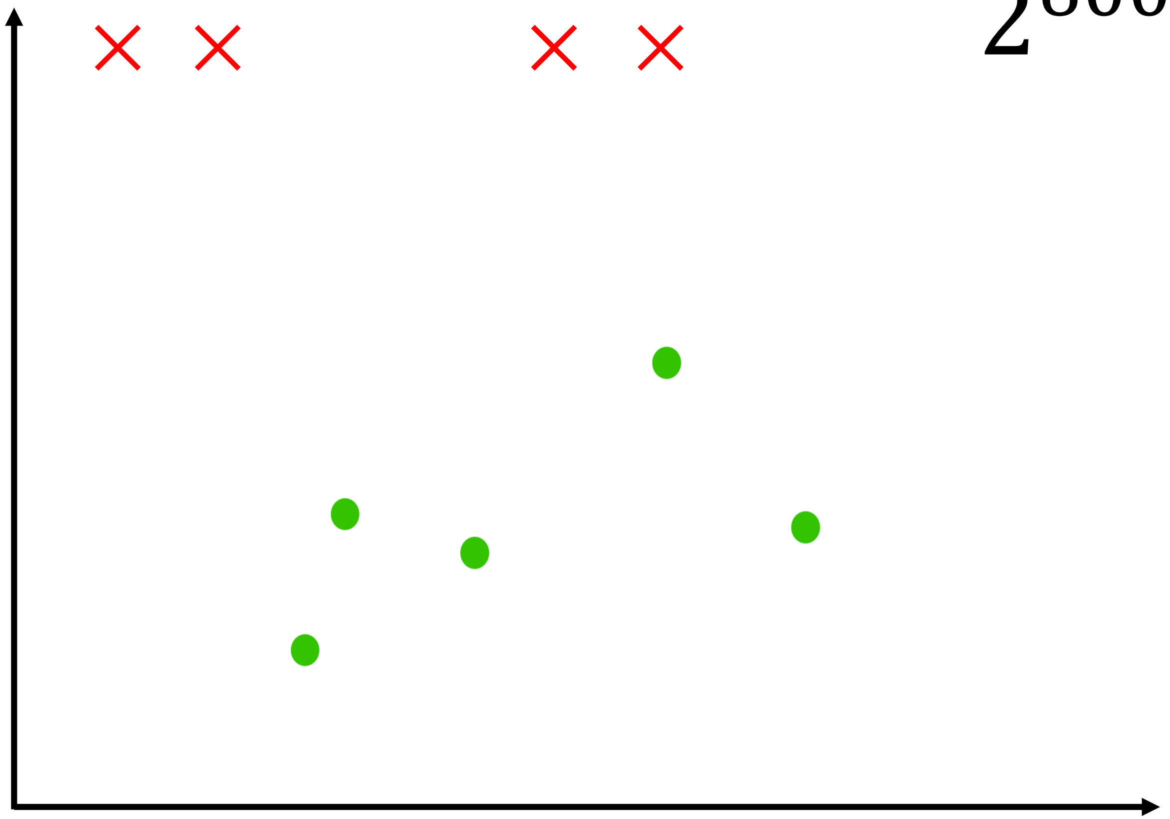


2^{600}

∞

$\min_{R \in \mathcal{R}} \mathrm{F}(R, U, X)$

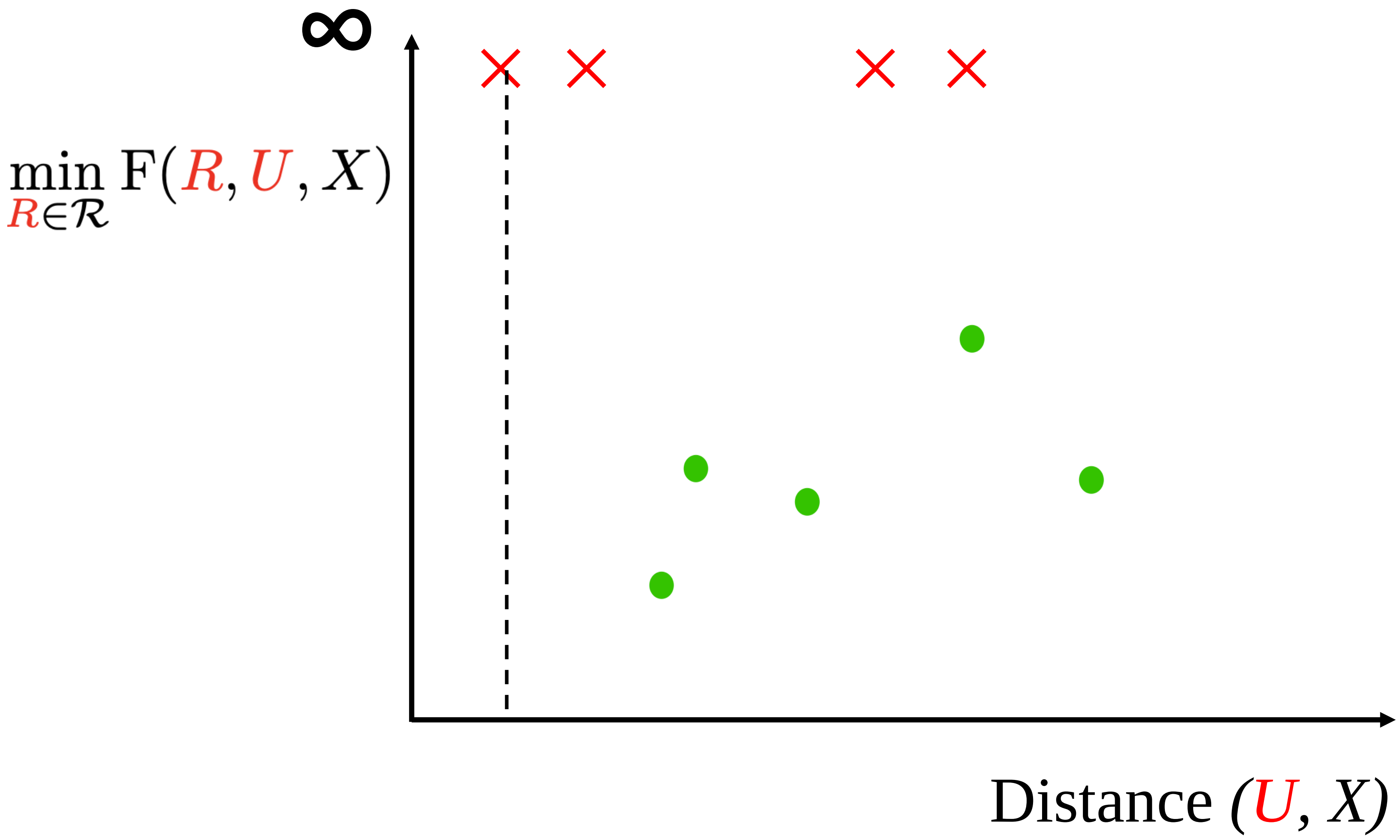
$|R| = 3^{180} * 3^{60}$



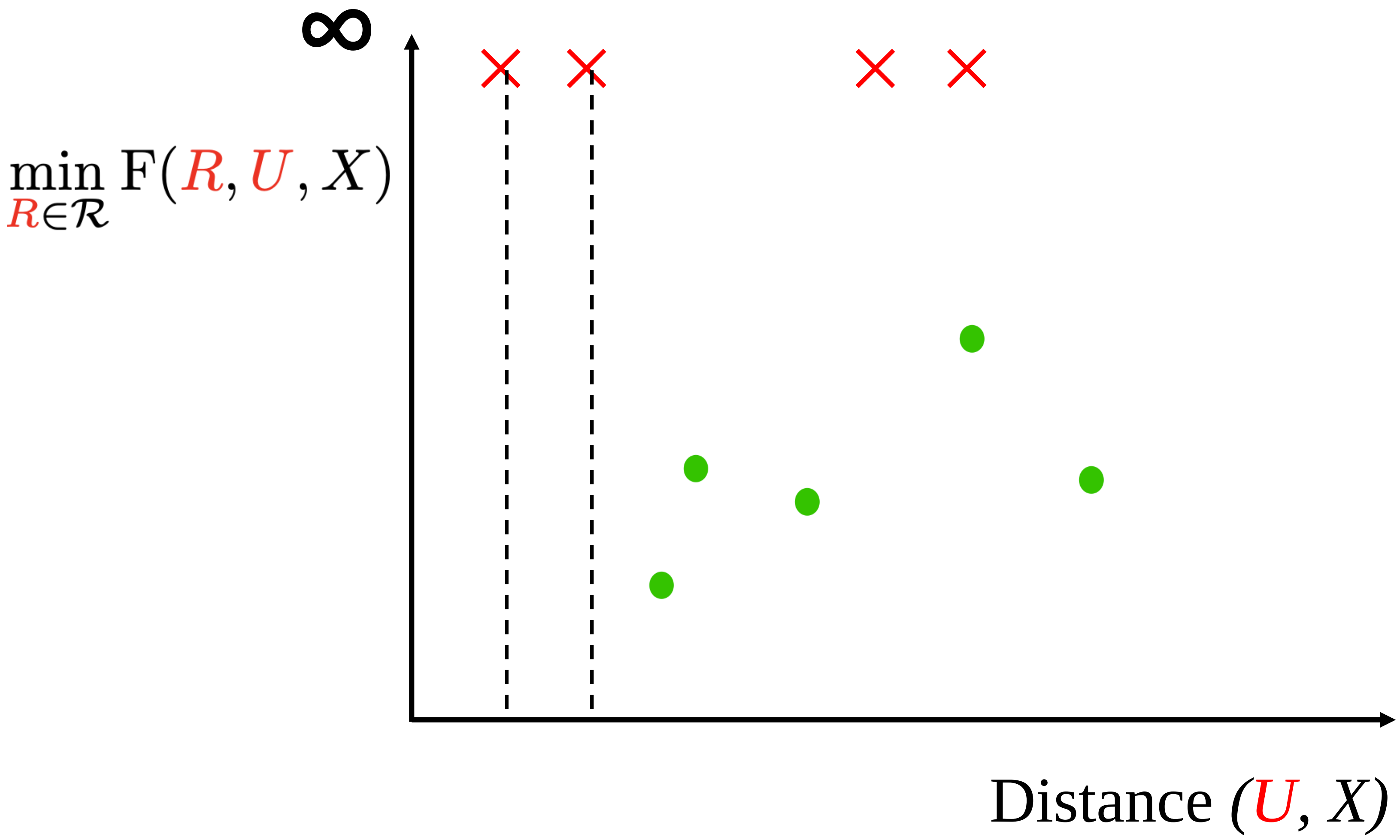
$|U| = 90^{34}$

Distance (U, X)

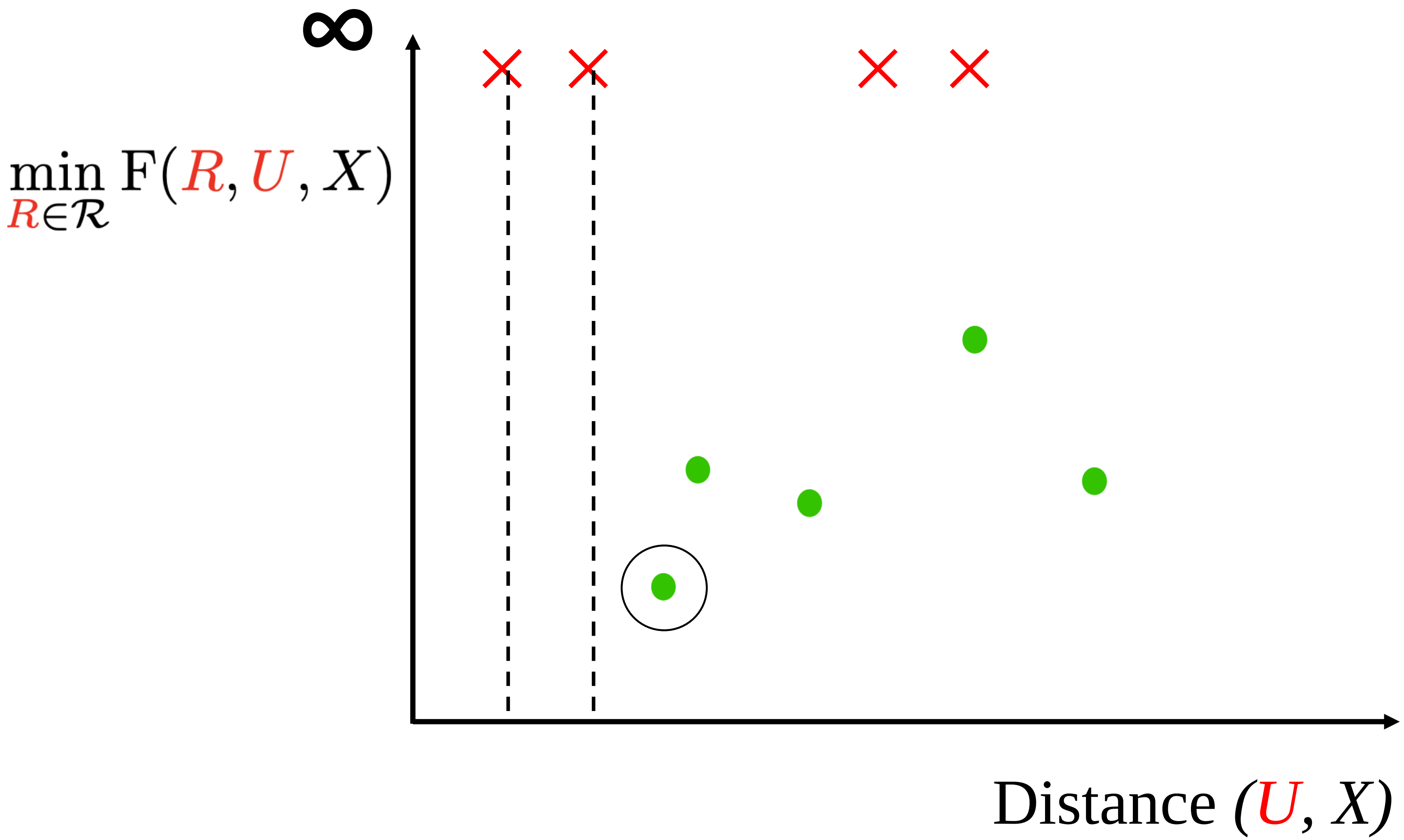
Our solution



Our solution



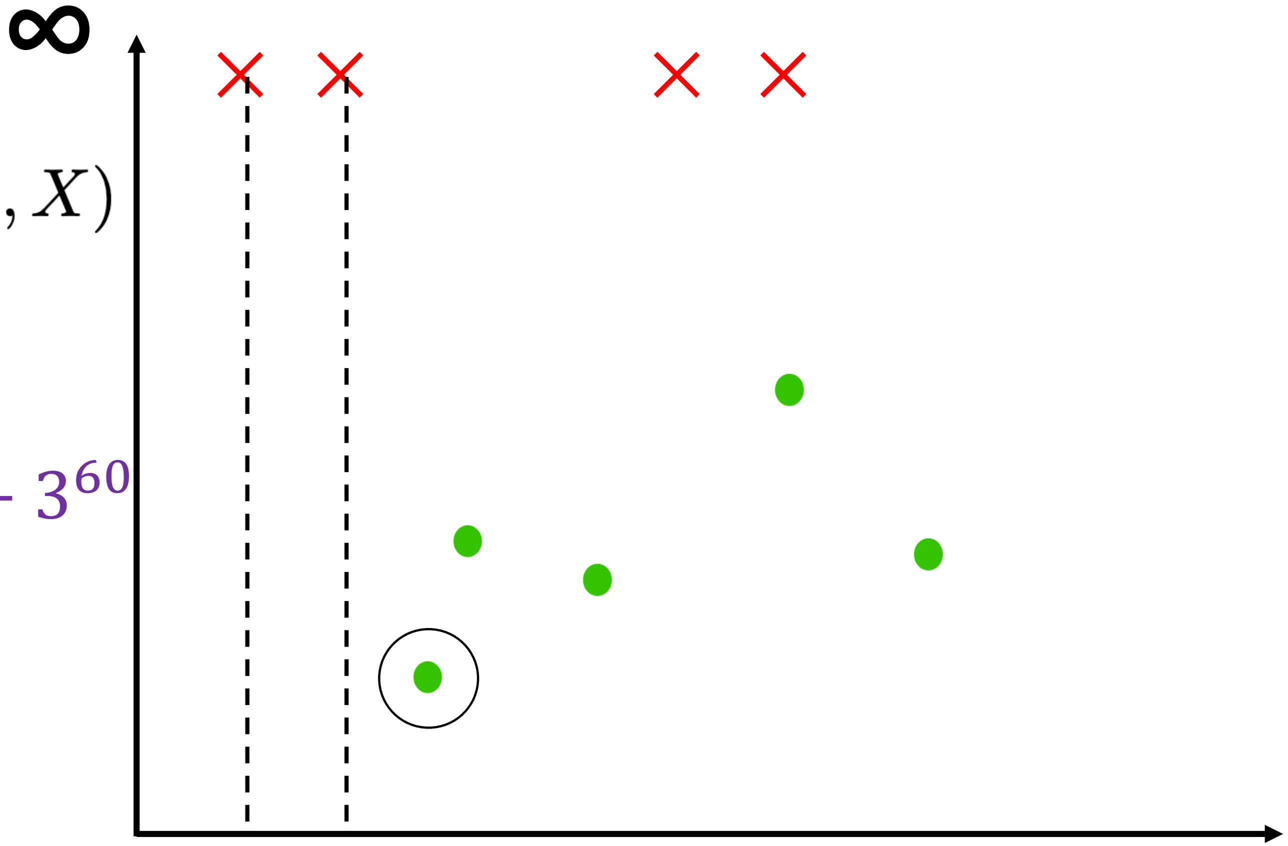
Our solution



Our solution

$$\min_{R \in \mathcal{R}} \mathrm{F}(R, U, X)$$

$$|R| = 3^{180} + 3^{60}$$



$|U| = 90^{34}$

Distance (U, X)

Our contributions

Decomposition of the rule learning problem

- Underlying form inference
- Change inference
- Condition inference

Efficient MaxSMT encoding

Experimental data

Textbook Problems : 34 (~20 Datapoints)

Lexical Datasets : 2 (~6000 Datapoints)

32 Languages

Evaluation metrics

Learn rule set from 20, 50 and 100 data points

	Accuracy	Rule Match	
		Precision	Recall
Flap 20	76	50	31
Flap 50	93	86	86
Flap 100	100	100	100
Verb 20	86	48	83
Verb 50	88	52	92
Verb 100	95	62	100

Evaluation metrics

Learn rule set from 20, 50 and 100 data points

Accuracy evaluated on held out data points

	Accuracy	Rule Match	
		Precision	Recall
Flap 20	76	50	31
Flap 50	93	86	86
Flap 100	100	100	100
Verb 20	86	48	83
Verb 50	88	52	92
Verb 100	95	62	100

Evaluation metrics

Learn rule set from 20, 50 and 100 data points

Accuracy evaluated on held out data points

Syntactic comparison of rule set against the gold standard rules

	Accuracy	Rule Match	
		Precision	Recall
Flap 20	76	50	31
Flap 50	93	86	86
Flap 100	100	100	100
Verb 20	86	48	83
Verb 50	88	52	92
Verb 100	95	62	100

LEXICAL DATASETS

Learn rule set from 20, 50 and 100 data points

Accuracy evaluated on held out data points

Syntactic comparison of rule set against the gold standard rules

	Accuracy	Rule Match	
		Precision	Recall
Flap 20	76	50	31
Flap 50	93	86	86
Flap 100	100	100	100
Verb 20	86	48	83
Verb 50	88	52	92
Verb 100	95	62	100

Evaluation textbook problems

Classes of textbook problems
of different complexity

	Accuracy	Rule Match	
		Precision	Recall
10 MAT	100	70	77
20 ALT	100	66	71
4 SUP	100	63	71
10 TEST	100	54	61

Evaluation textbook problems

Classes of textbook problems
of different complexity

	Accuracy	Rule Match	
		Precision	Recall
10 MAT	100	70	77
20 ALT	100	66	71
4 SUP	100	63	71
10 TEST	100	54	61

Held out test problems

Evaluation textbook problems

Classes of textbook problems
of different complexity

	Accuracy	Rule Match	
		Precision	Recall
10 MAT	100	70	77
20 ALT	100	66	71
4 SUP	100	63	71
10 TEST	100	54	61

Held out test problems

INFERENCE TIME SPEEDUP

SYPHON = BASELINE / 10^2

	Inference Time (secs)		
	SYPHON	Baseline	Speedup
MAT	30.0	3100	124.6
ALT	10.7	N/A	N/A
SUP	5.3	6333	378.3
TEST	8.3	N/A	N/A

Examples

Graphics programs

Linguistics

Control

- Verma, Murali, Singh, Kohli, Chaudhuri. Programmatically Interpretable Reinforcement Learning ICML'18

Programmatically Interpretable Reinforcement Learning

Learn a policy to drive a car in a TORCS simulator

Programmatic vs neural:

- Neural is fast on the track where it trained
- Bur programmatic drives smoother and doesn’t crash on tracks where it didn’t train!