Fanoos: Multi-Resolution, Multi-Strength, Interactive Explanations for Learned Systems

David Bayani , Stefan Mitsch
dcbayani@alumni.cmu.edu
smitsch@cs.cmu.edu

This material is based upon work supported by the United States Air Force and DARPA under Contract No. FA8750-18-C-0092. Any opinions, findings and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the United States Air Force and DARPA.

VMCAI
Jan. 17, 2022

Acronym to Know: XAI

- “Explainable AI” : XAI
- Our specific focus here: Explanations for Machine-Learned Systems

Overview of Why Want XAI

Overview of Why Want XAI

- Critical for many end-users. Ex: Doctors

Overview of Why Want XAI

- Critical for many end-users. Ex: Doctors
- Explicit Legal Requirements: EU’s GDPR

Overview of Why Want XAI

- Critical for many end-users. Ex: Doctors
- Explicit Legal Requirements: EU’s GDPR
- For AI Scientists and Engineers: need to understand, debug, and tweak systems.

Overview of Why Want XAI

- Critical for many end-users. Ex: Doctors
- Explicit Legal Requirements: EU’s GDPR
- For AI Scientists and Engineers: need to understand, debug, and tweak systems.
- As a result: exponential growth in recent XAI publications ([1])

Variety in XAI Stances and Approaches

- Large variety in what counts as an explanations and, subsequently, methods

Variety in XAI Stances and Approaches

- Large variety in what counts as an explanations and, subsequently, methods
  - See, for instance: “Mythos of Model Interpretability” ([10])

Variety in XAI Stances and Approaches

- Large variety in what counts as an explanations and, subsequently, methods
  - See, for instance: “Mythos of Model Interpretability” ([10])
  - Being loose here: if I say “interpretability”, “explainability”, or “transparency”, etc., I probably mean the same thing

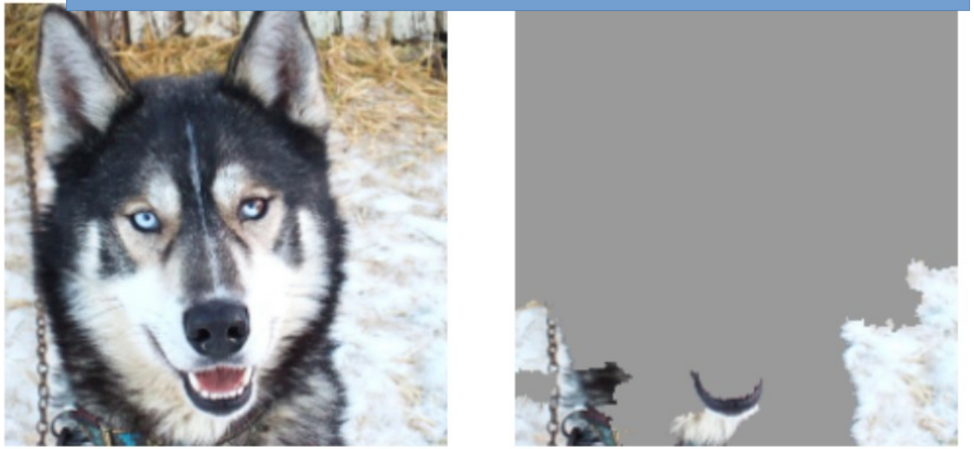
Variety in XAI Cnt.: Example Diffs.

Medium /
Style of
Explanation

Variety in XAI Cnt.: Example Diffs

LIME ([12])

Medium / Style of Explanation



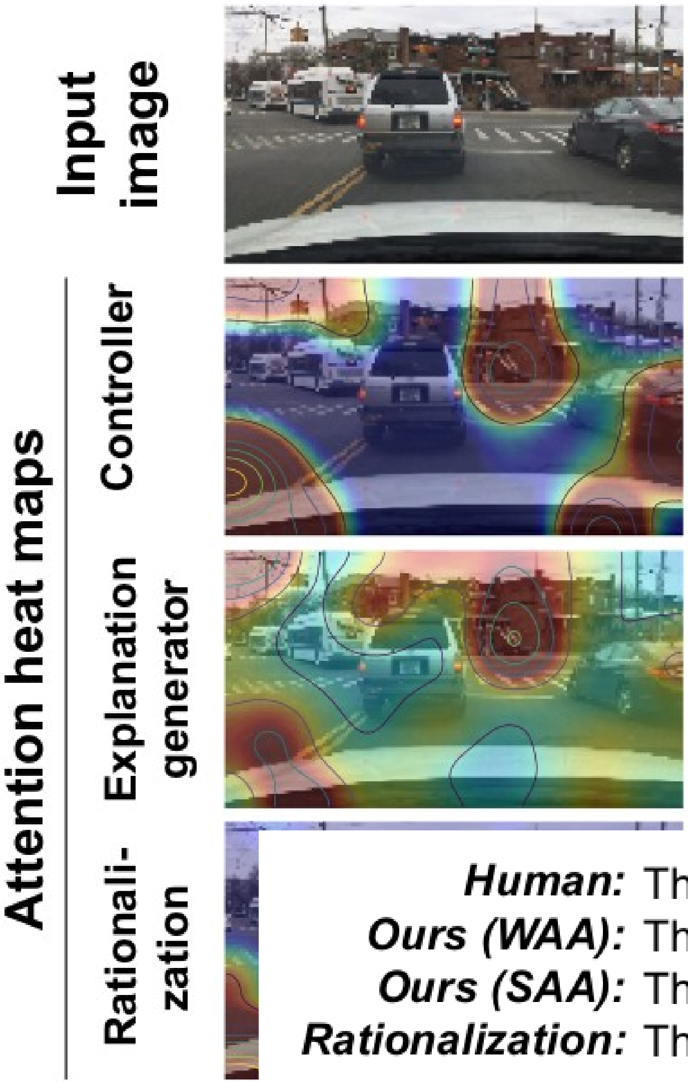
(a) Husky classified as wolf

(b) Explanation

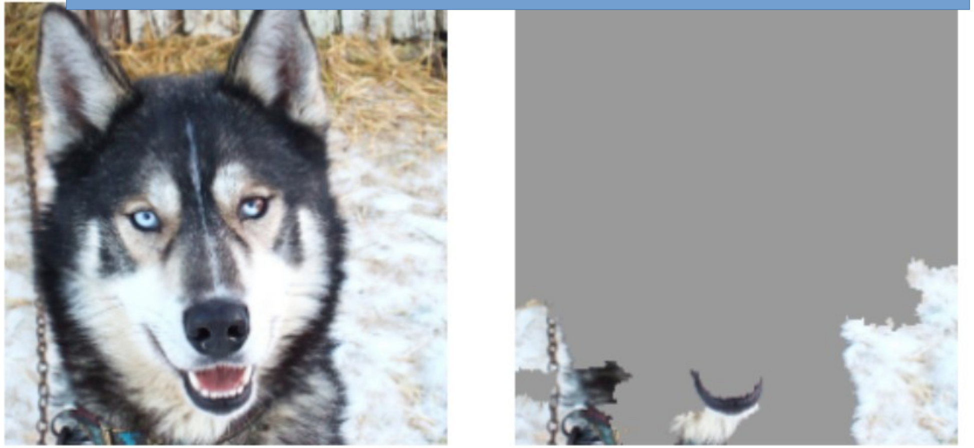
Figure 11: Raw data and explanation of a bad model's prediction in the "Husky vs Wolf" task.

XAI Cnt.: Example Diffs

1



LIME ([12])



(a) Husky classified as wolf (b) Explanation

Figure 11: Raw data and explanation of a bad model's prediction in the "Husky vs Wolf" task.

Kim et al. ([8])

Human: The car steadily driving **+** now that the cars are moving.
Ours (WAA): The car is driving forward **+** because traffic is moving freely.
Ours (SAA): The car heads down the road **+** because traffic is moving at a steady pace.
Rationalization: The car slows down **+** because it's getting ready to a stop sign.

Variety in XAI Cnt.: Example Diffs.

Medium / Style of Explanation

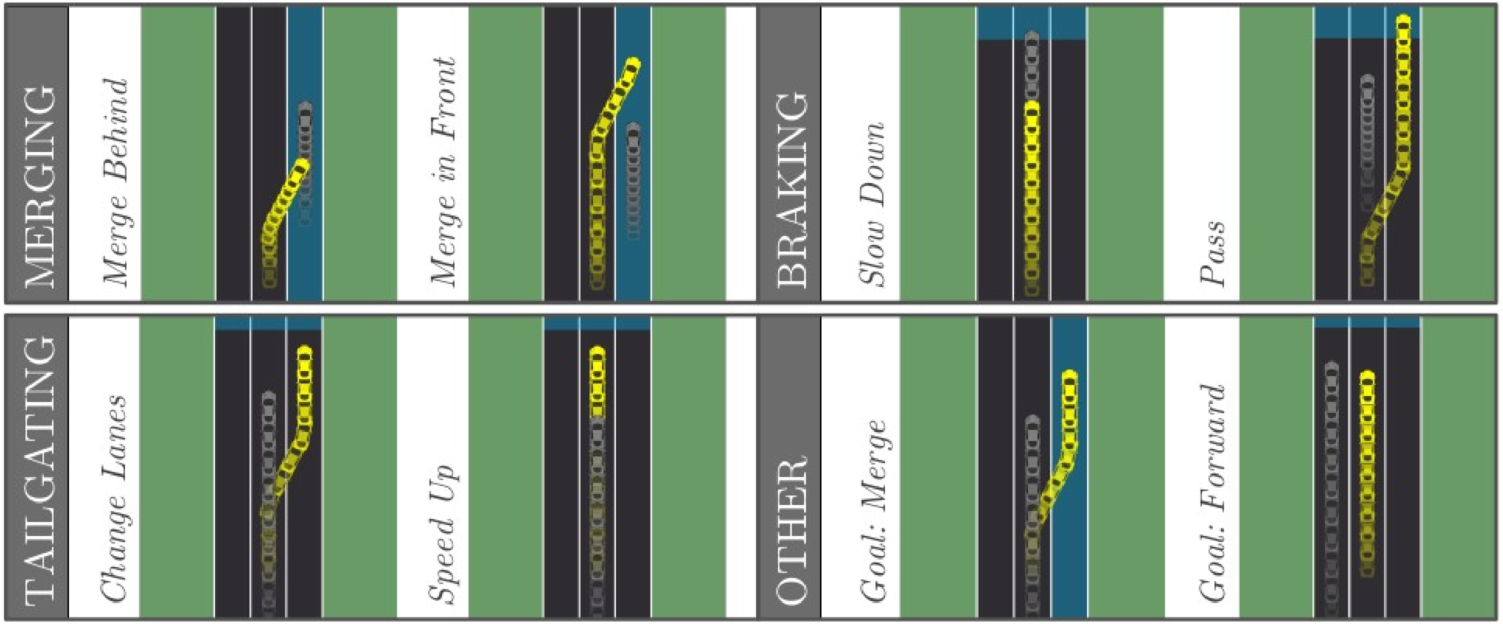
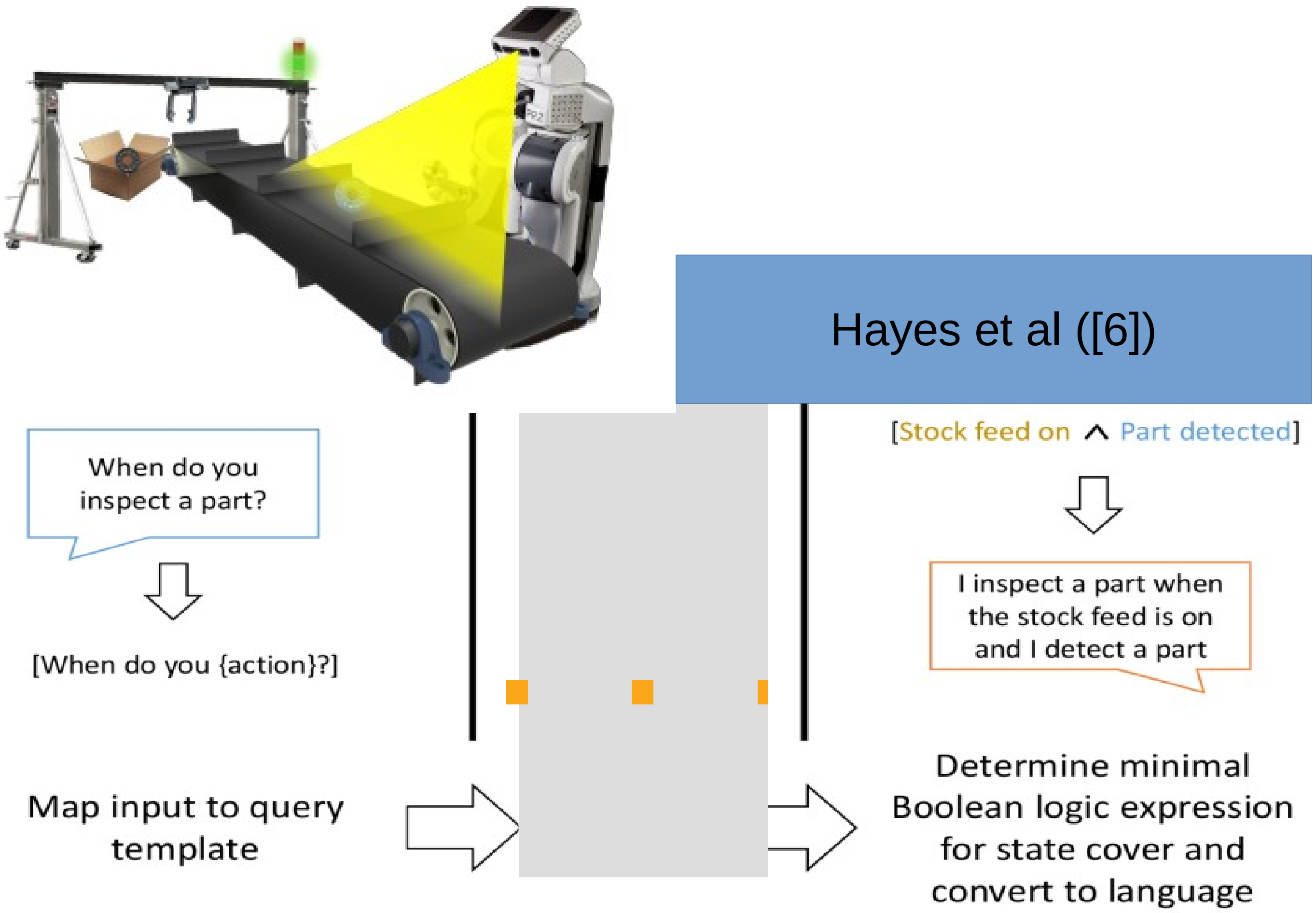


Fig. 2: The possible driving environments cluster naturally into four classes, with two trajectory strategies per class. Each image shows the trajectories of the autonomous car (yellow) and non-autonomous car (gray) in a particular environment. Positions later in the trajectory are more opaque. The goal of the autonomous car in each environment is highlighted in blue: merge into the right lane or drive forward.

Huang et al. ([7])

Variety in XAI Cnt.: Example Diffs.



Hayes et al ([6])

When do you inspect a part?

[When do you {action}?]

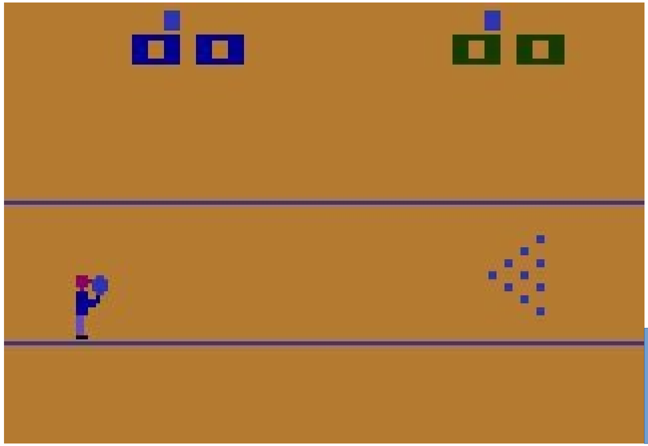
Map input to query template

[Stock feed on ∧ Part detected]

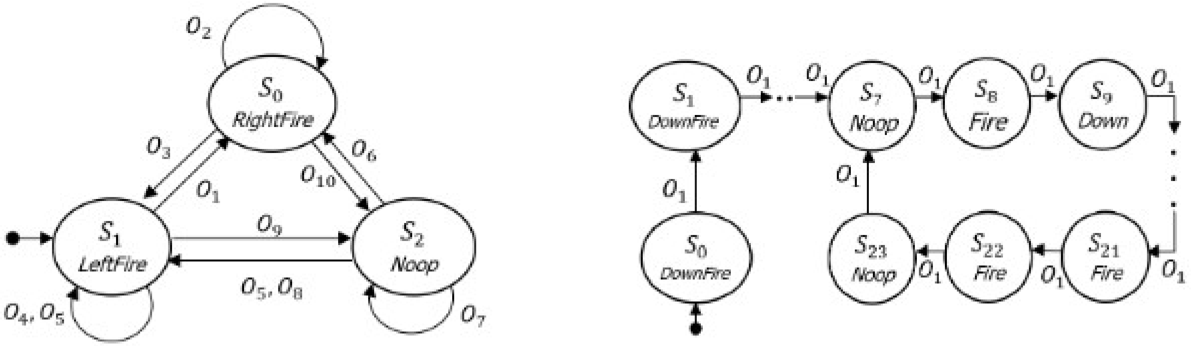
I inspect a part when the stock feed is on and I detect a part

Determine minimal Boolean logic expression for state cover and convert to language

Variety in XAI Cnt.: Example Diffs.



Koul et al ([9])



(a) Pong (B_h=64 and B_f=400)

(b) Bowling (B_h=128 and B_f=100)

Figure 2: Moore Machine representation for Atari policies

Current State and Trends in XAI for ML

- Fast growing, but currently has limitations:

Current State and Trends in XAI for ML

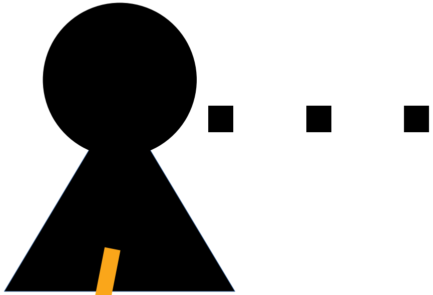
- Fast growing, but currently has limitations:
  - Lack of guarantees

Current State and Trends in XAI for ML

- Fast growing, but currently has limitations:
  - Lack of guarantees
  - Explanations of *single* granularity/abstraction level

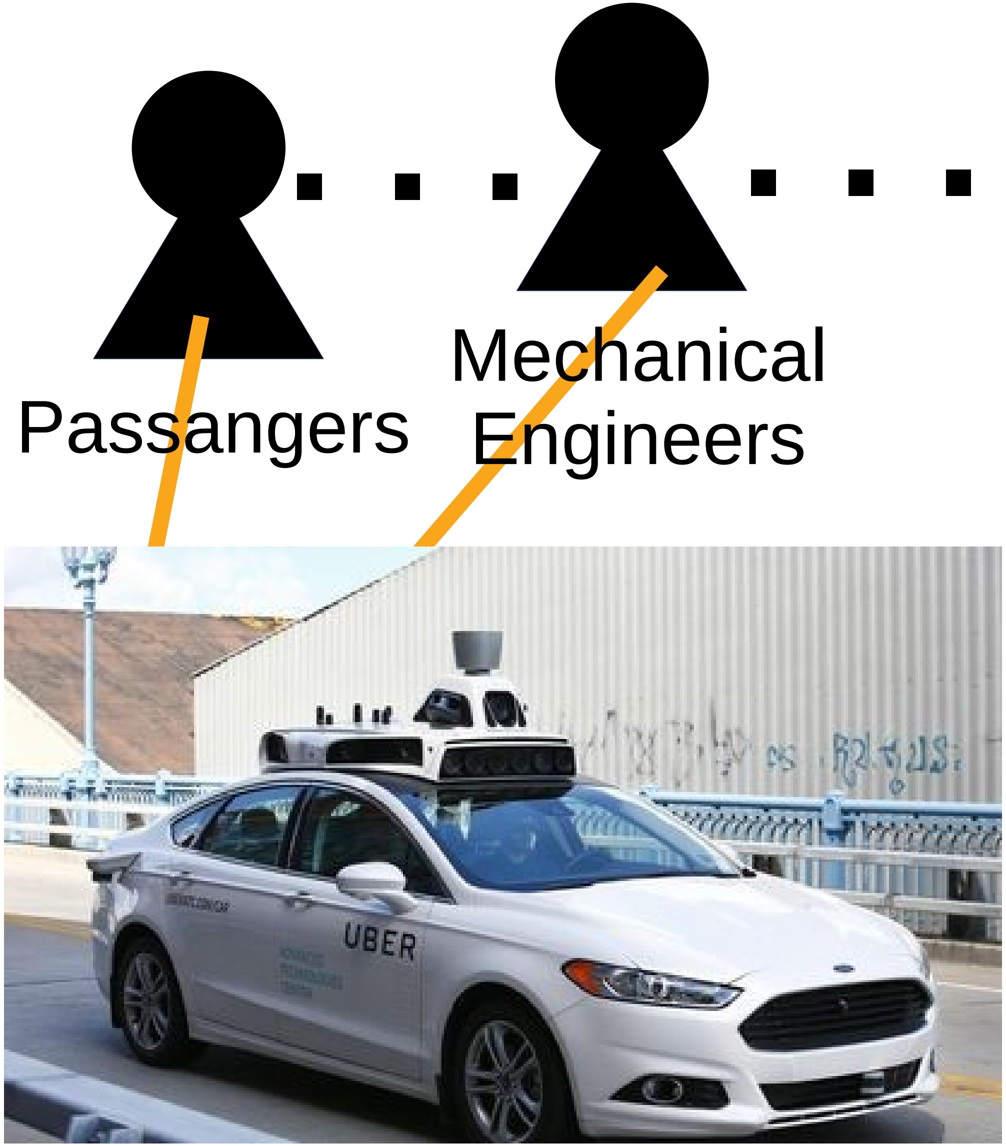
Individual ML systems are part of a larger whole in tackling problems

Image credit: https://www.chicagotribune.com/business/blue-sky/ct-uber-self-driving-cars-pittsburgh-20160906-story.html



Passangers

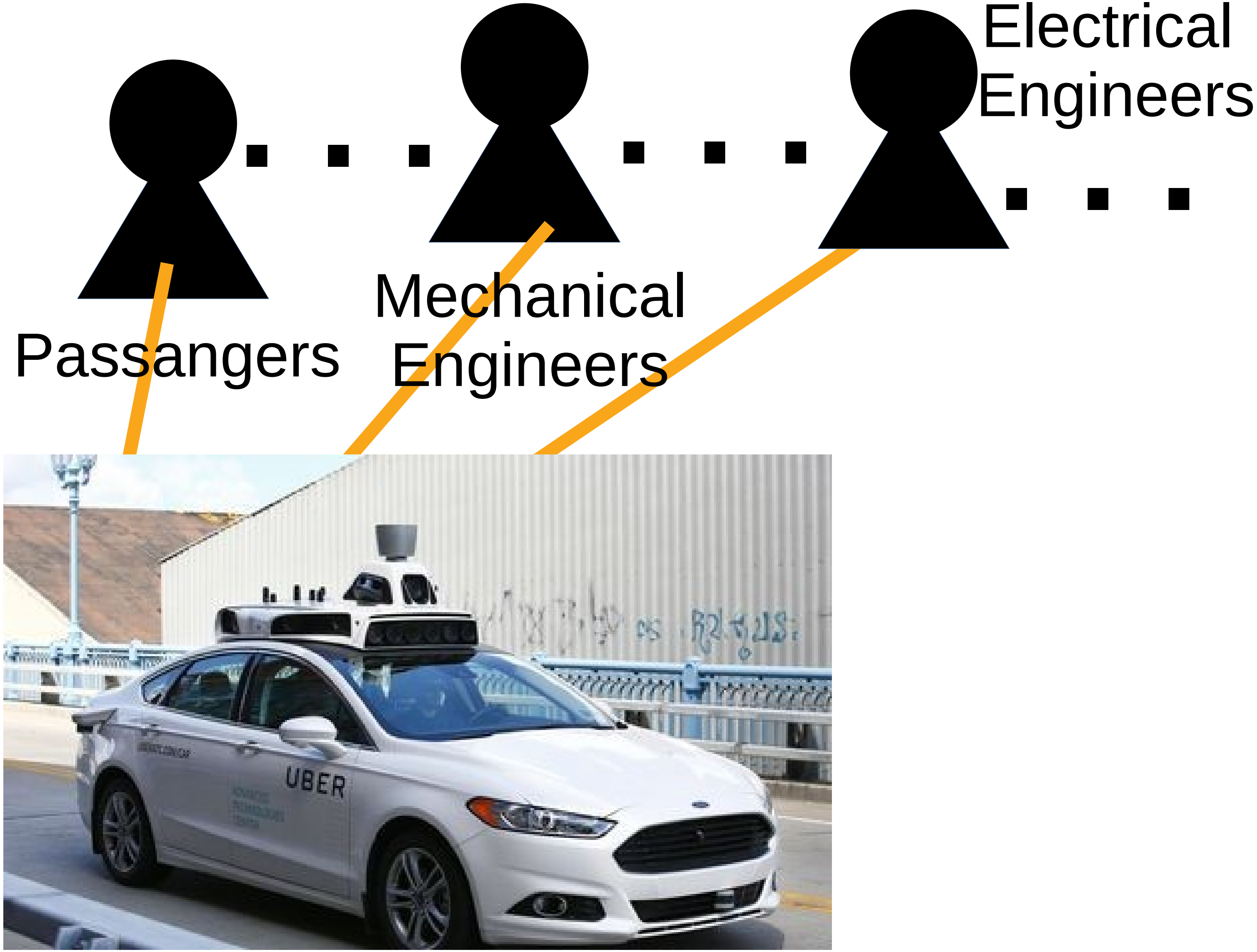
Image credit: https://www.chicagotribune.com/business/blue-sky/ct-uber-self-driving-cars-pittsburgh-20160906-story.html



Passangers

Mechanical Engineers

Image credit: https://www.chicagotribune.com/business/blue-sky/ct-uber-self-driving-cars-pittsburgh-20160906-story.html

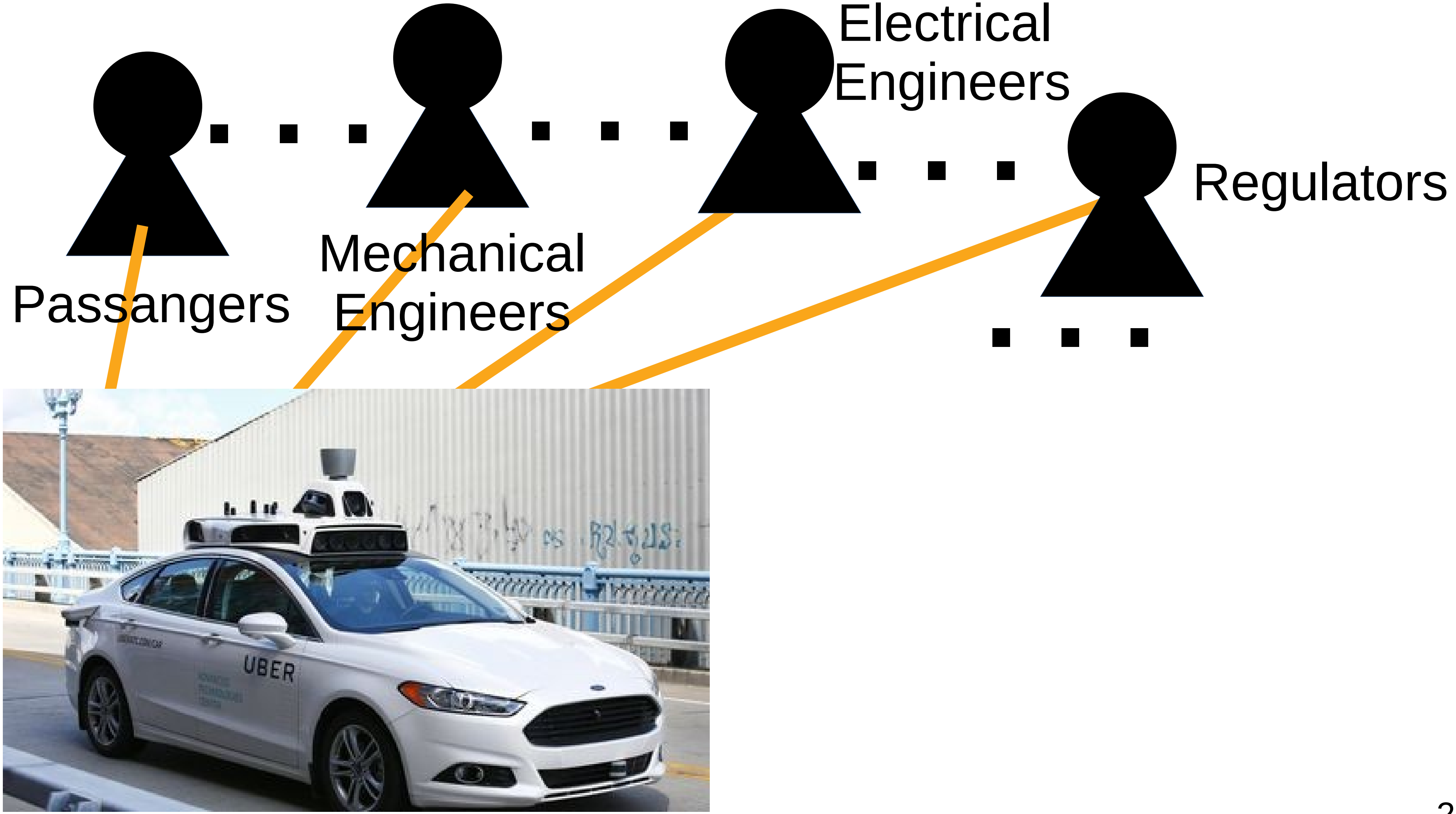


Electrical Engineers

Mechanical Engineers

Passangers

Image credit: https://www.chicagotribune.com/business/blue-sky/ct-uber-self-driving-cars-pittsburgh-20160906-story.html



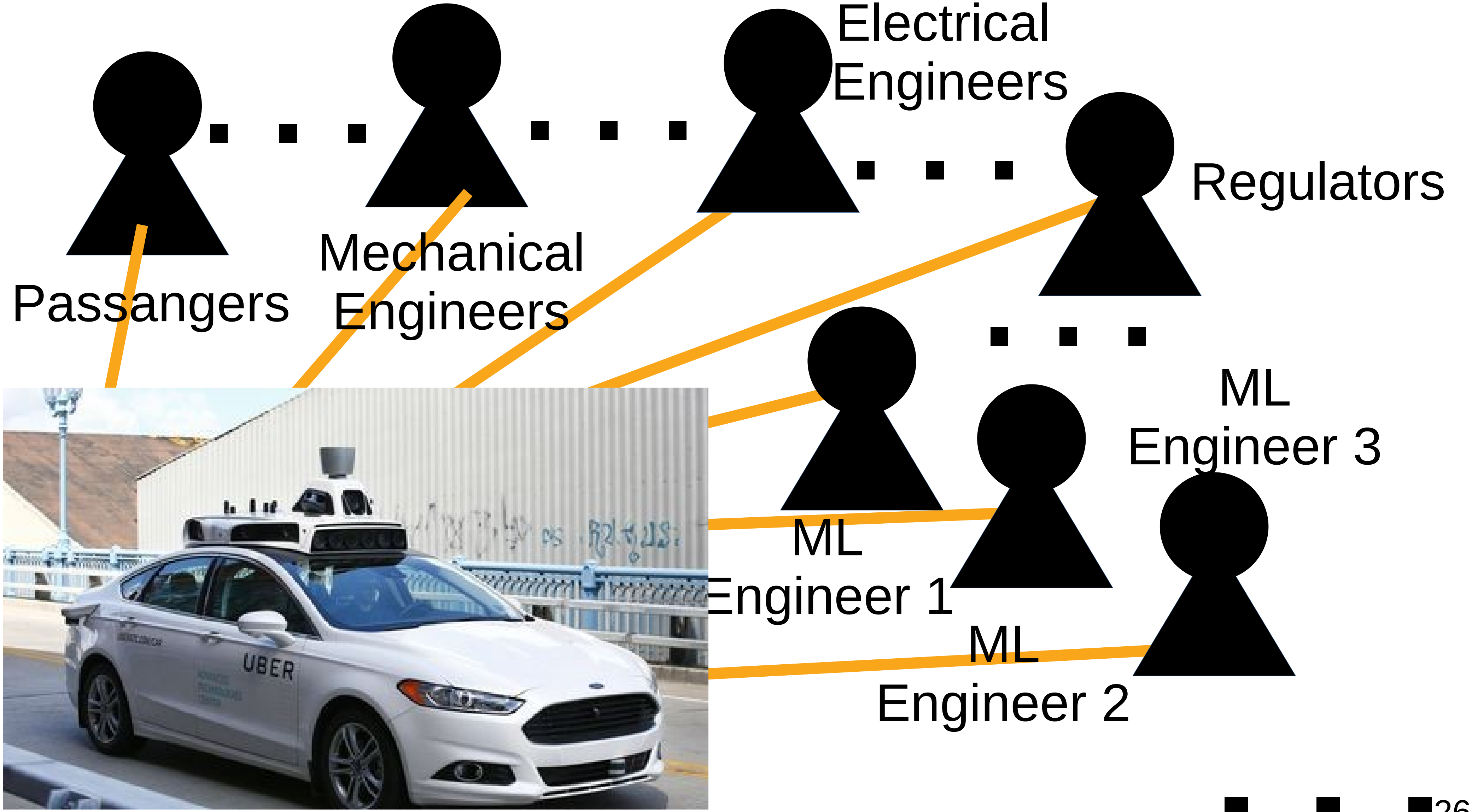
Electrical Engineers

Regulators

Passangers

Mechanical Engineers

Image credit: https://www.chicagotribune.com/business/blue-sky/ct-uber-self-driving-cars-pittsburgh-20160906-story.html



Passangers

Mechanical Engineers

Electrical Engineers

Regulators

ML Engineer 1

ML Engineer 2

ML Engineer 3

Image credit: https://www.chicagotribune.com/business/blue-sky/ct-uber-self-driving-cars-pittsburgh-20160906-story.html

Some Observations

Actors have many different:

- Interests
- Needs – depends on actor ***and task at hand***
  - Stakes vary. Safety : high stakes, efficiency tweaks: lower stakes
- Backgrounds / Expertise

Unfilled Desiderata for XAI

- *Interactive* (so can explore as needed)

Unfilled Desiderata for XAI

- *Interactive* (so can explore as needed)
- *Can provide multiple abstraction levels of information* (so can suite multiple audiences and needs)

Unfilled Desiderata for XAI

- *Interactive* (so can explore as needed)
- *Can provide multiple abstraction levels of information* (so can suite multiple audiences and needs)
- Can provide strong guarantees about info. provided (so *explanations necessarily reflect system behavior*)
  - Should be as pedantic about details as user needs (*sometimes want / don’t want corner-case info.*)

Our Solution: Fanoos

- Fanoos (فــانوس) - “Lantern” in Farsi

“Shining a Light on Black-Box AI”

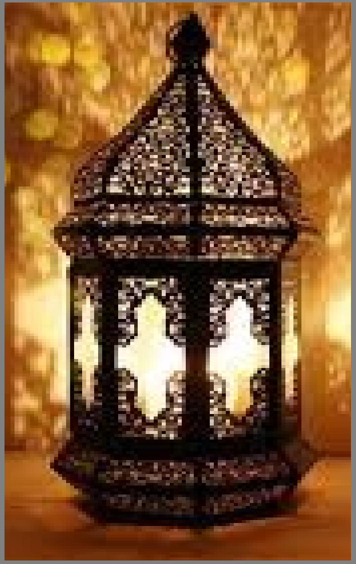
Masrawy.com

Plan For Next Few Slides

1) Overview of setup & what user sees
2) Description of the mechanics
3) Brief overview of experiments
4) Some high-level closing thoughts

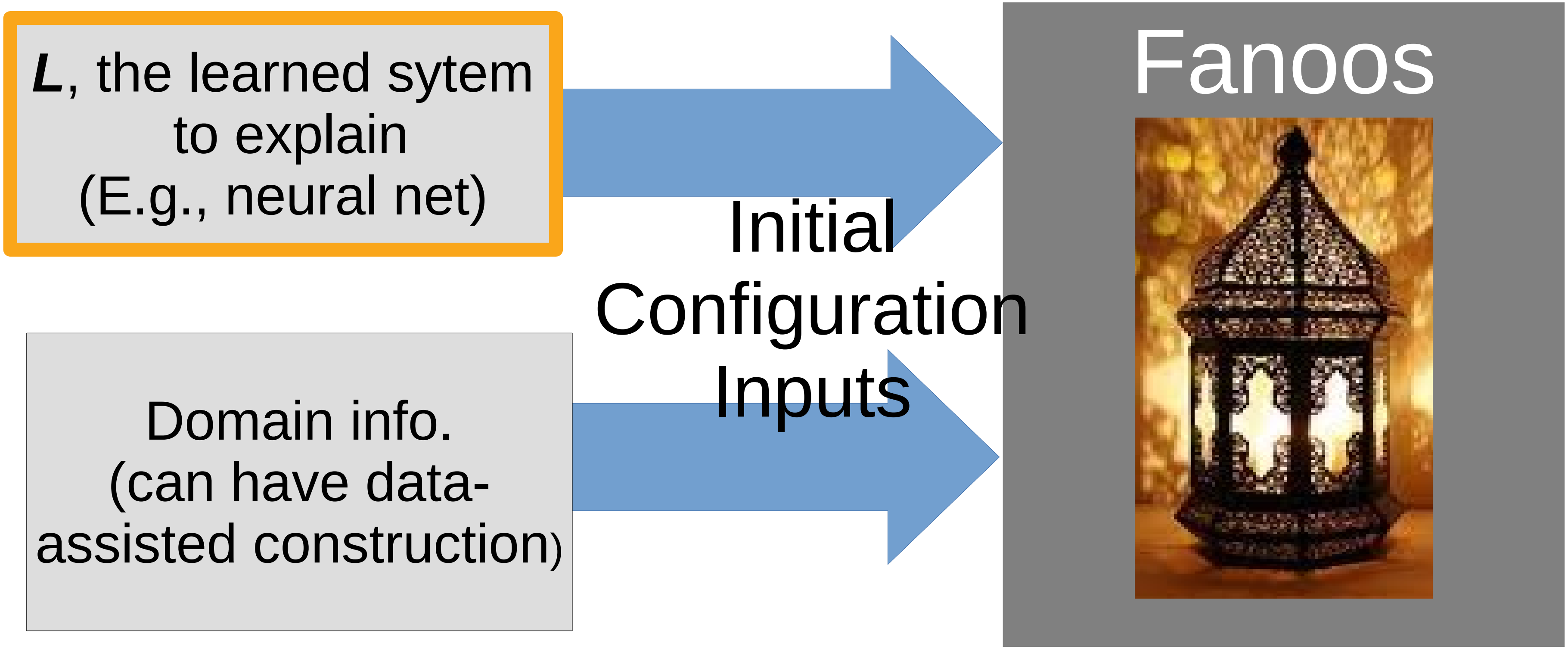
Setup & User View

Fanoos



Fanoos Overview:

Initial Setup



L, the learned sytem to explain (E.g., neural net)

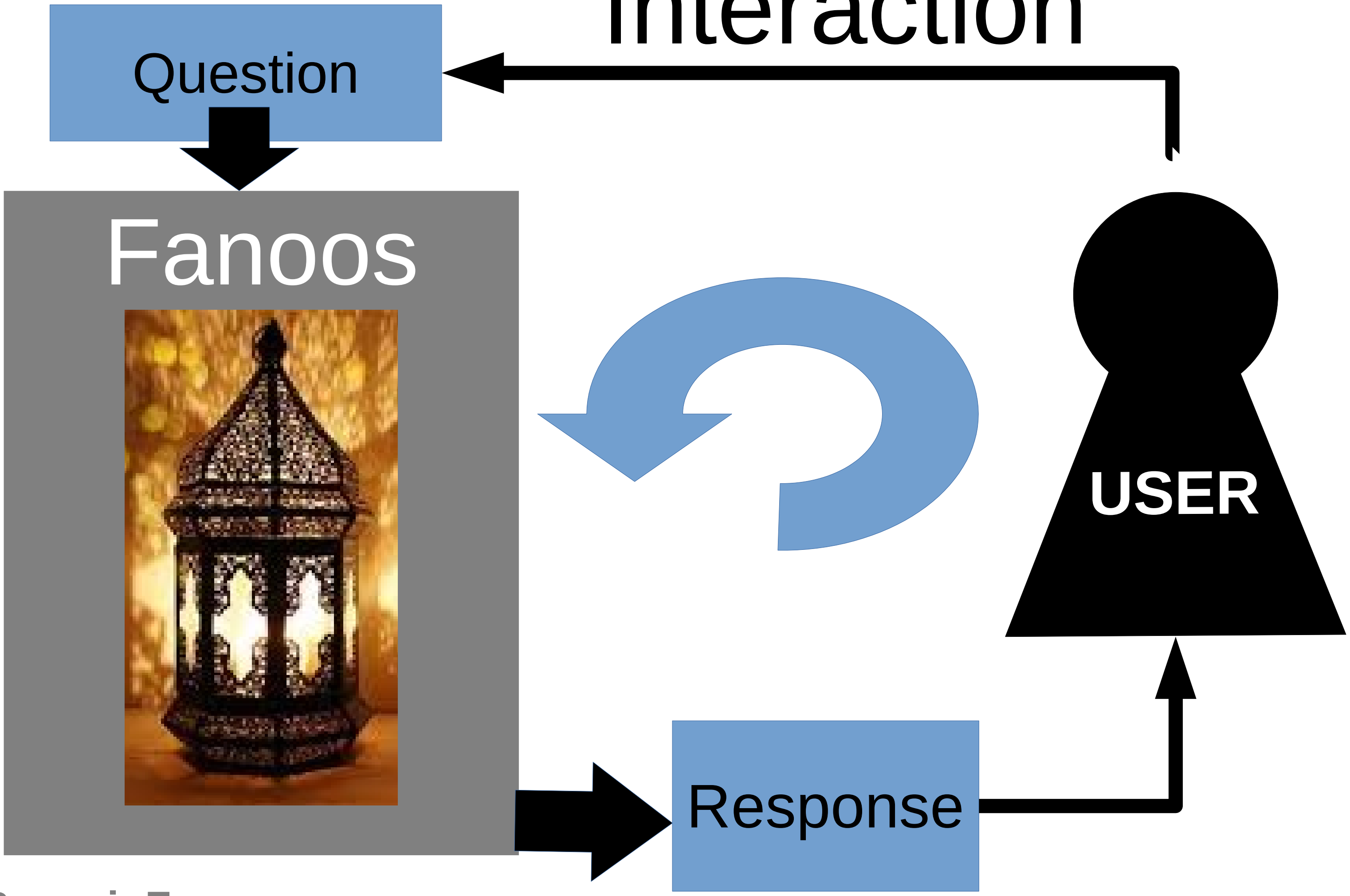
Domain info. (can have data-assisted construction)

Initial Configuration Inputs

Fanoos

Fanoos Overview:

Interaction



Question

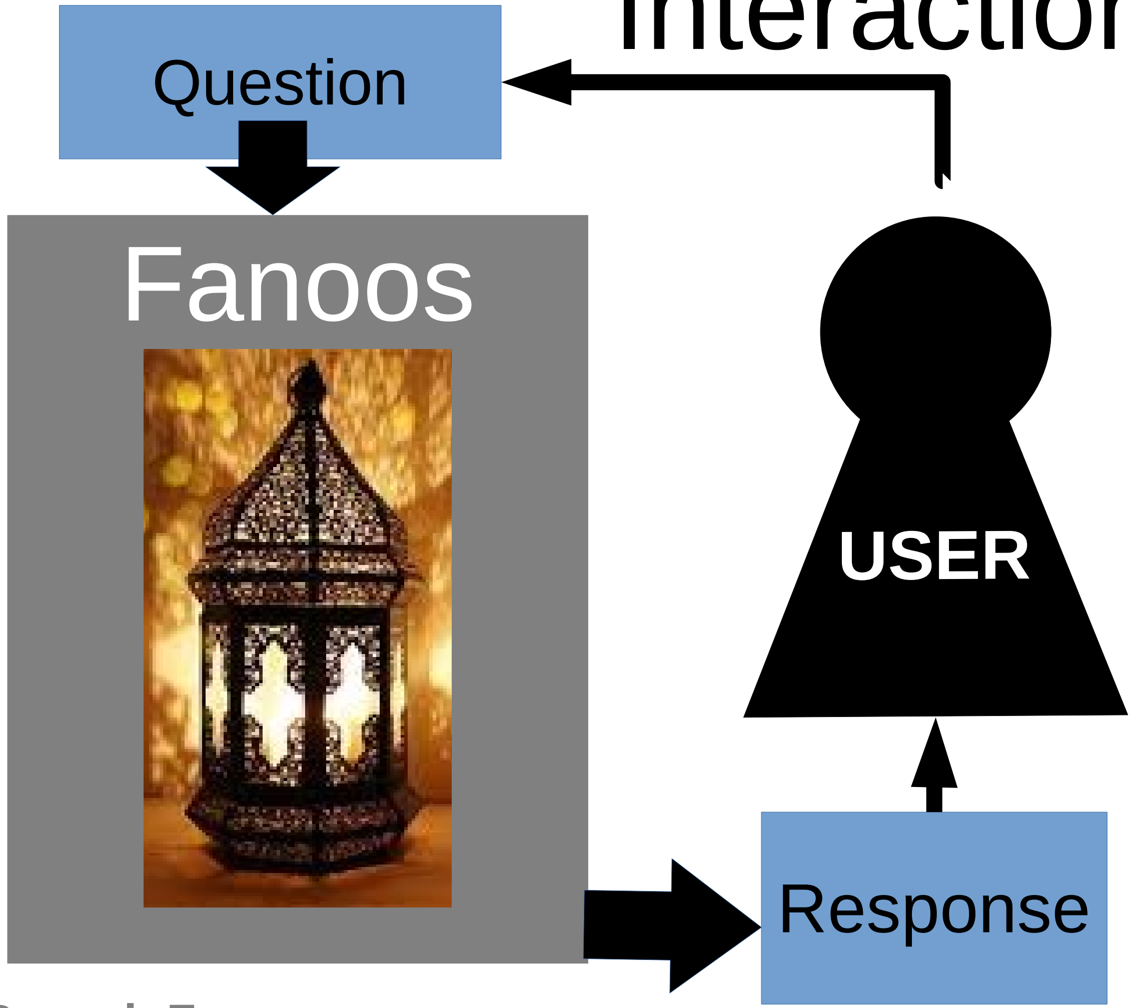
Fanoos

Response

USER

Fanoos Overview:

Interaction



Question

Fanoos

USER

Response

Question Types:

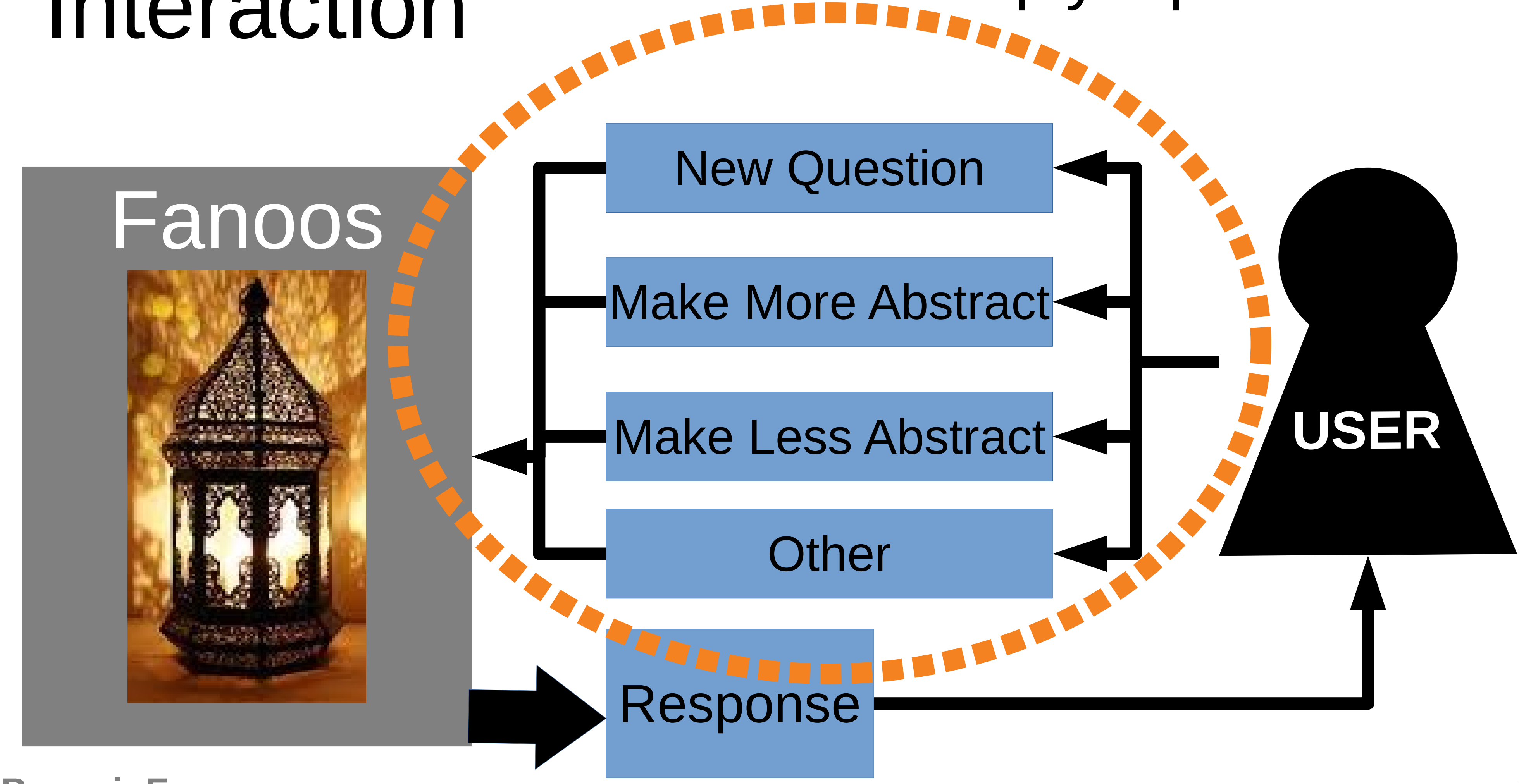
- When does ***L*** do X?
- What does ***L*** do when Y?
- In what circumstances is ***L*** doing X during Y?

Can be formally sound or probabilistically guaranteed

Fanoos Overview:

Interaction

User Reply Options



Fanoos

New Question

Make More Abstract

Make Less Abstract

Other

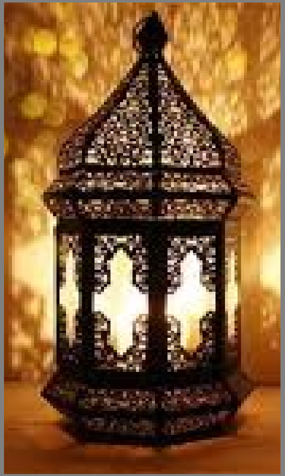
Response

USER

Fanoos Overview:

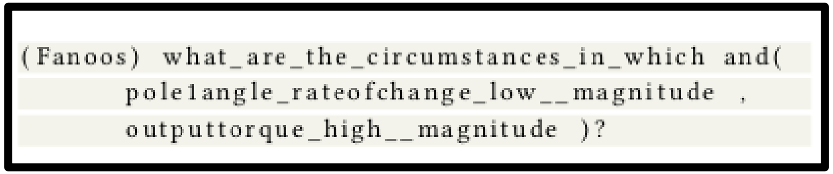
Interaction Example

Fanoos



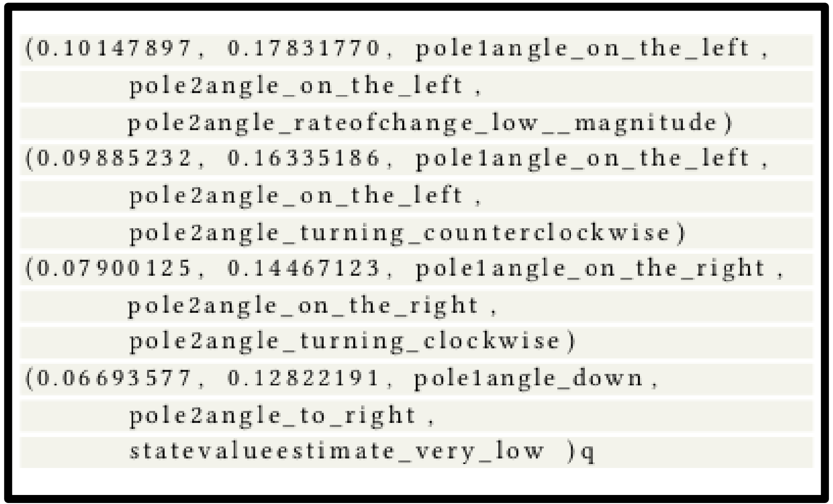
Example from robotics

Initial Question



```
(Fanoos) what_are_the_circumstances_in_which and(
        pole1angle_rateofchange_low__magnitude ,
        outputtorque_high__magnitude )?
```

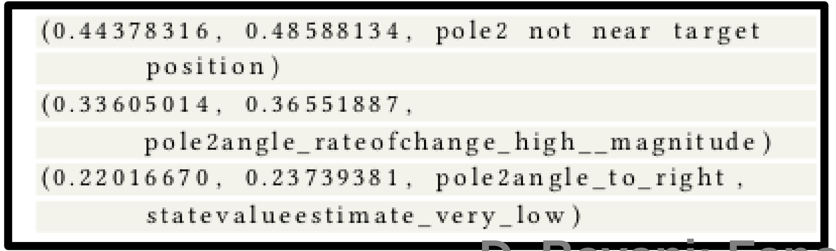
Initial Response



```
(0.10147897, 0.17831770, pole1angle_on_the_left ,
        pole2angle_on_the_left ,
        pole2angle_rateofchange_low__magnitude)
(0.09885232, 0.16335186, pole1angle_on_the_left ,
        pole2angle_on_the_left ,
        pole2angle_turning_counterclockwise)
(0.07900125, 0.14467123, pole1angle_on_the_right ,
        pole2angle_on_the_right ,
        pole2angle_turning_clockwise)
(0.06693577, 0.12822191, pole1angle_down ,
        pole2angle_to_right ,
        statevalueestimate_very_low )q
```

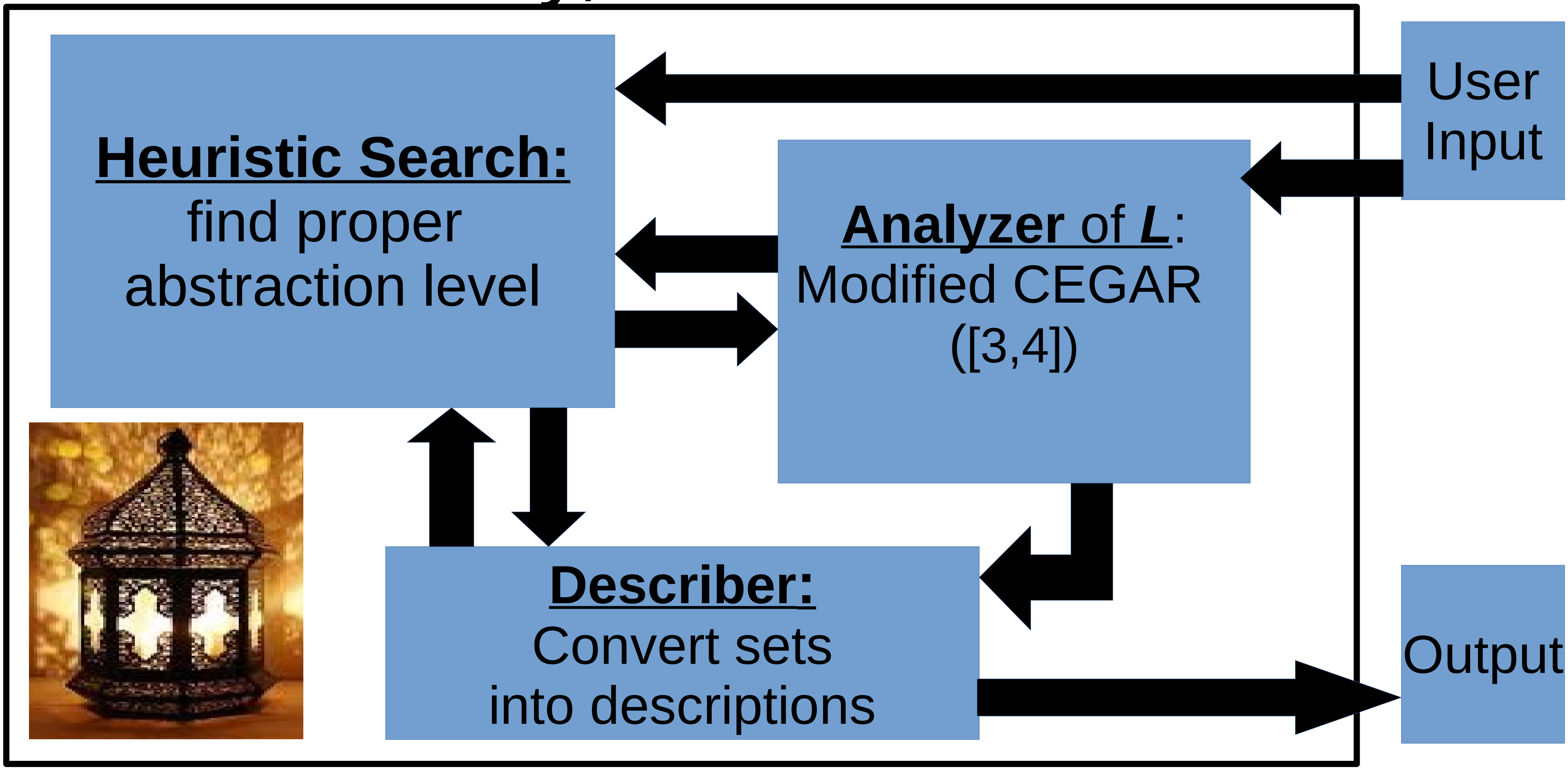
User Request ***More*** abstract

New Response



```
(0.44378316, 0.48588134, pole2 not near target
        position)
(0.33605014, 0.36551887,
        pole2angle_rateofchange_high__magnitude)
(0.22016670, 0.23739381, pole2angle_to_right ,
        statevalueestimate_very_low)
```

Briefly, Inside Fanoos



Heuristic Search:
find proper
abstraction level

Analyzer of *L*:
Modified CEGAR
([3,4])

Describer:
Convert sets
into descriptions

User
Input

Output

Mechanics

Fanoos

Domain Knowledge that User Provides

- The learned system, *L*
- Universe bounding-box for input space
  - Ex: for a constant velocity Dubin car:

$$(x, y, \theta) \in [-1, 1] \times [50.3, 100.0] \times [0, 2\pi]$$

- Predicates: connecting sets to something user grasps. Ex:
  - "left arm higher than right arm" : y_arm1 > y_arm2
  - “attempting spiral roll” :

$$\exists cx, cy \in B.|(x - cx)^2 + (y - cy)^2 - r^2| \leq \epsilon_1 \wedge |2(x - cx)dx - 2(y - cy)dy| \leq \epsilon_2 \wedge \ldots$$

Domain Knowledge that User Provides

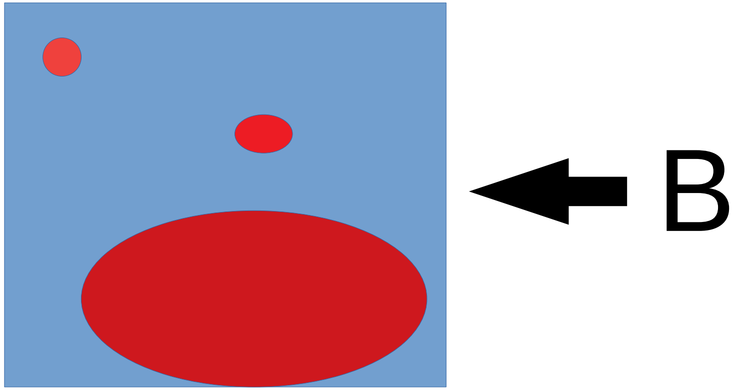
- Note: predicates are grounded

Domain Knowledge that User Provides

- Note: predicates are grounded
  - Facilitates semi-automatic or fully automatic generation (if desired)

Domain Knowledge that User Provides

- Note: predicates are grounded
  - Facilitates semi-automatic or fully automatic generation (if desired)
  - Using a SAT-solver, we can say whether predicate holds over sets



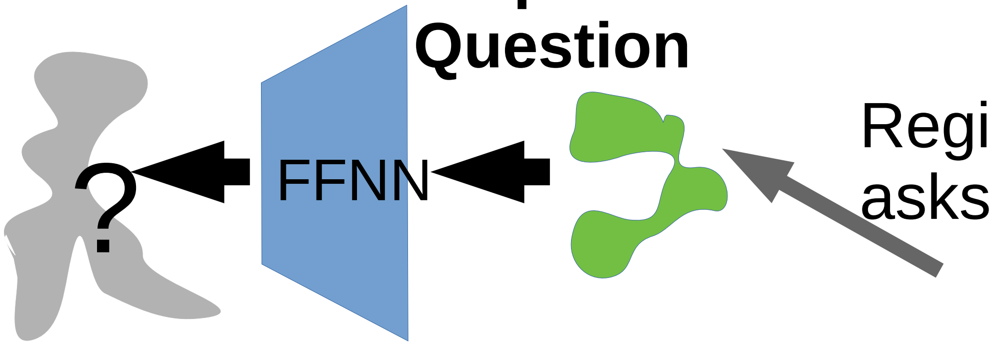
Red : P fails to hold
Blue: P holds

$$\text{“}\forall v \in B.P(v)\text{” is false}$$
$$\text{“}\exists v \in B.P(v)\text{” is true}$$

Overview: Responding to Questions

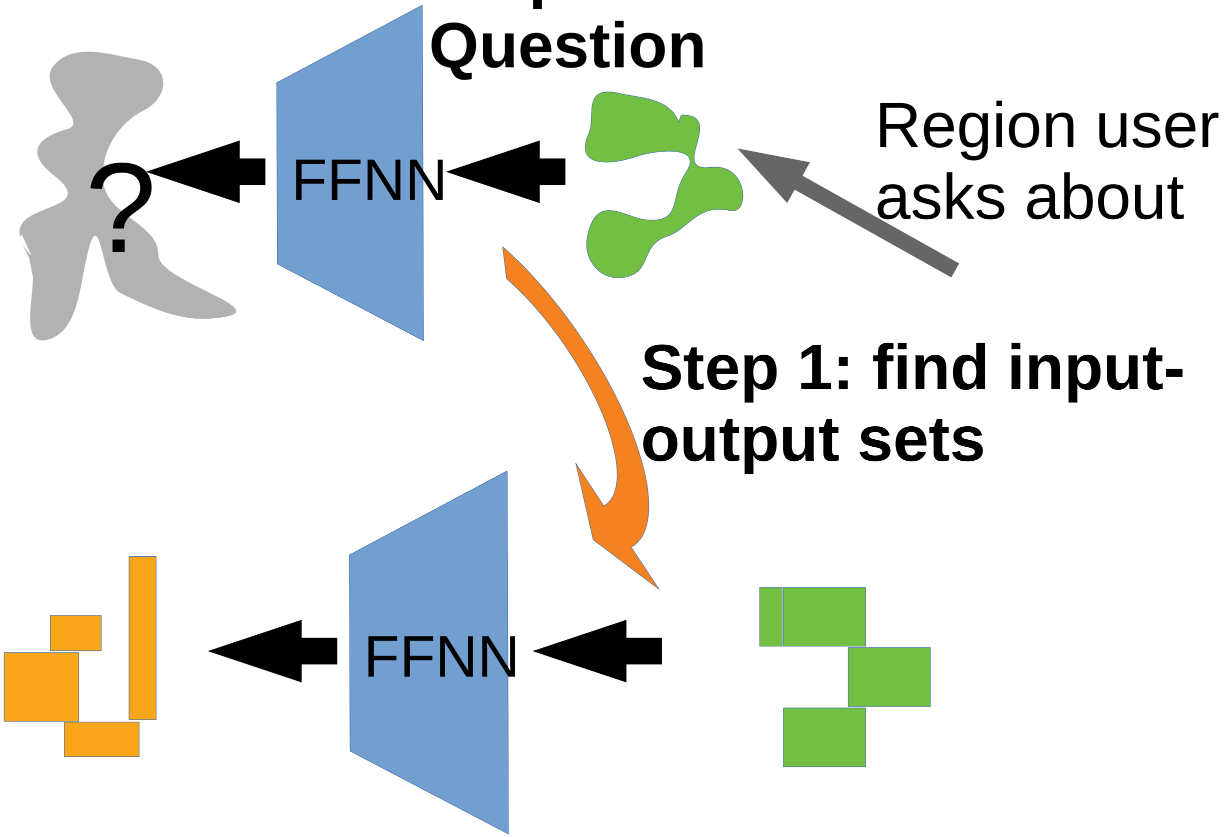
Overview: Responding to Questions

Step 0: User asks Question



Region user asks about

Overview: Responding to Questions

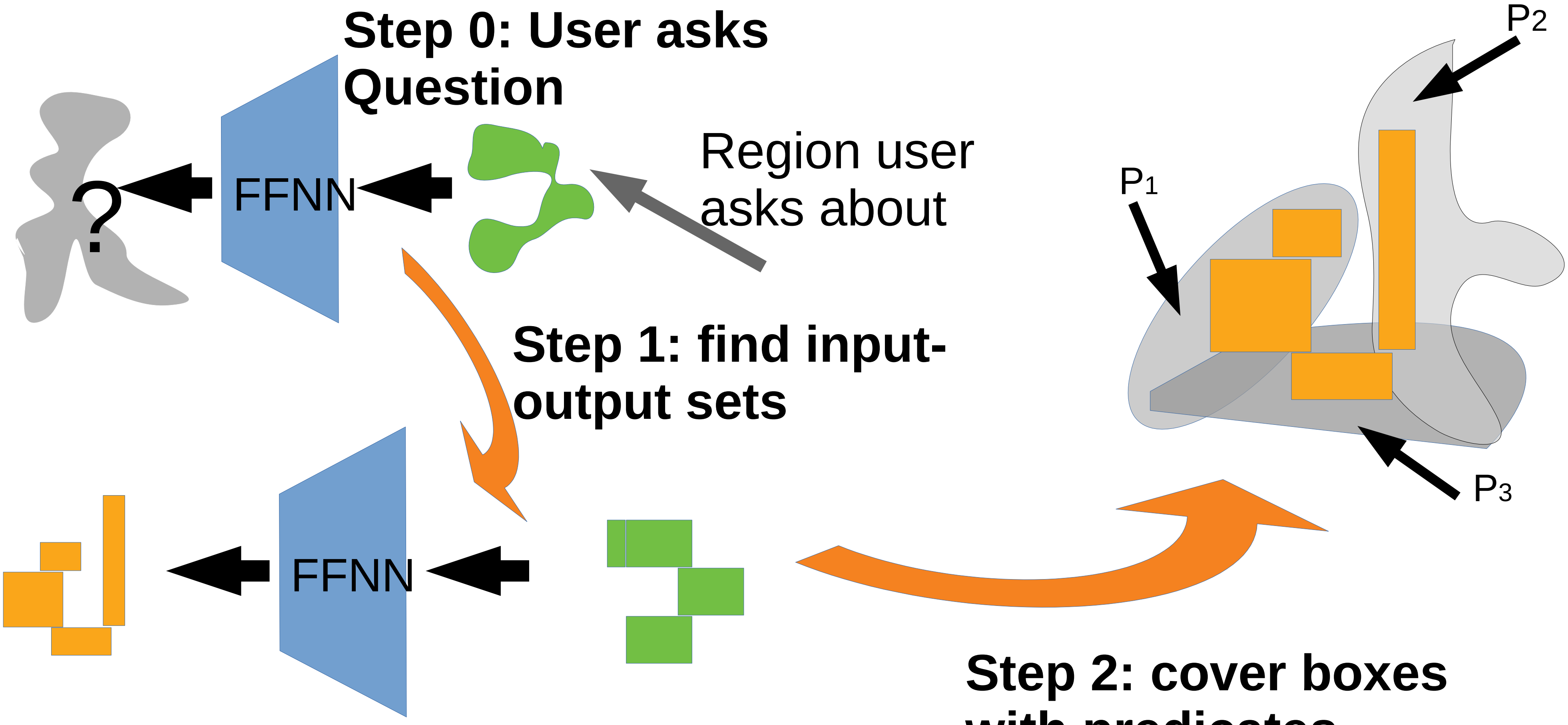


Step 0: User asks Question

Region user asks about

Step 1: find input-output sets

Overview: Responding to Questions



Step 0: User asks Question

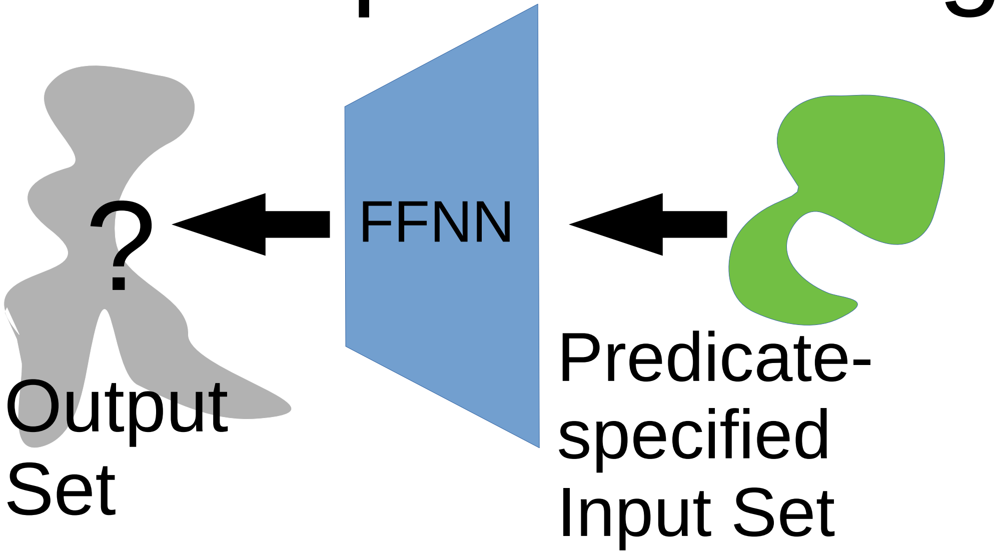
Region user asks about

Step 1: find input-output sets

Step 2: cover boxes with predicates

Step 1: “Finding the Other Set”

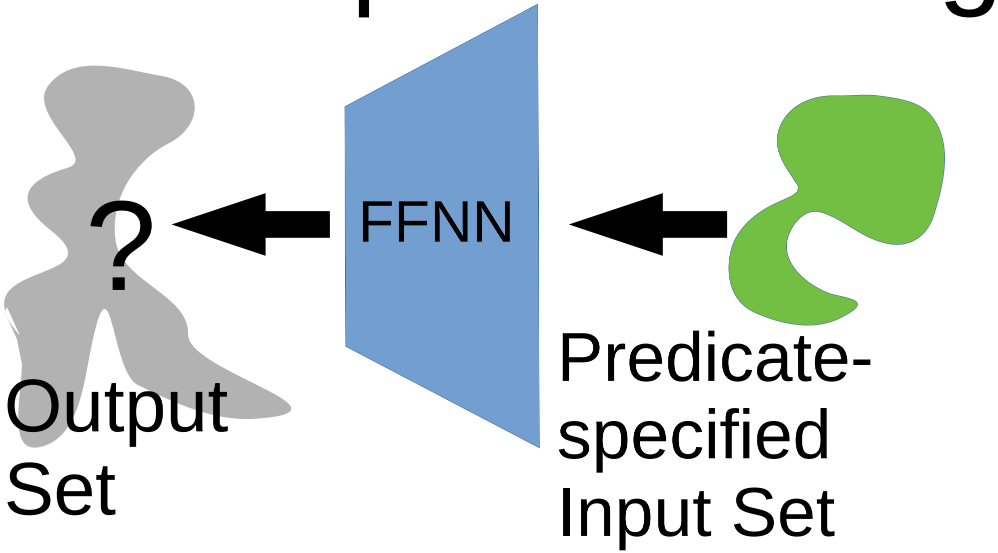
Step 1: “Finding the Other Set”



Output Set

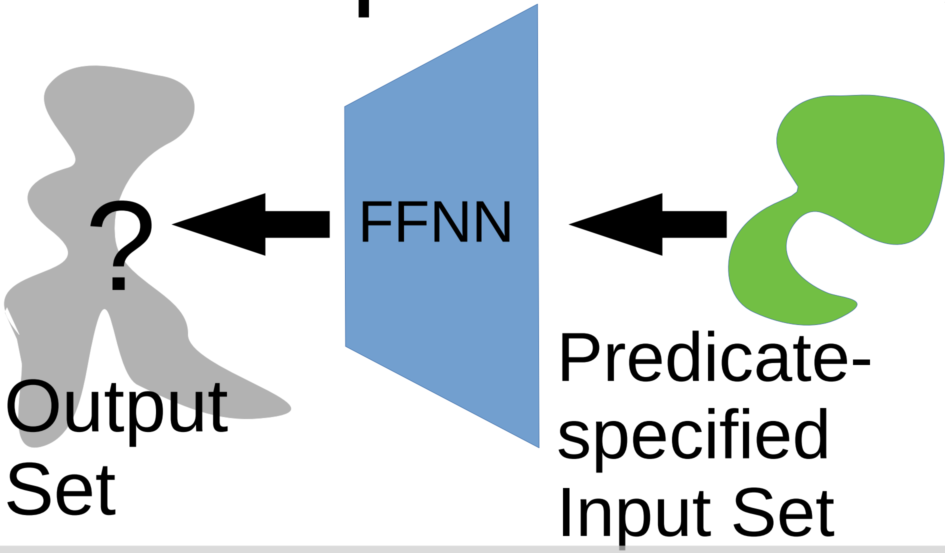
Predicate-specified Input Set

Step 1: “Finding the Other Set”



Question: “What do you do when …?”

Step 1: “Finding the Other Set”

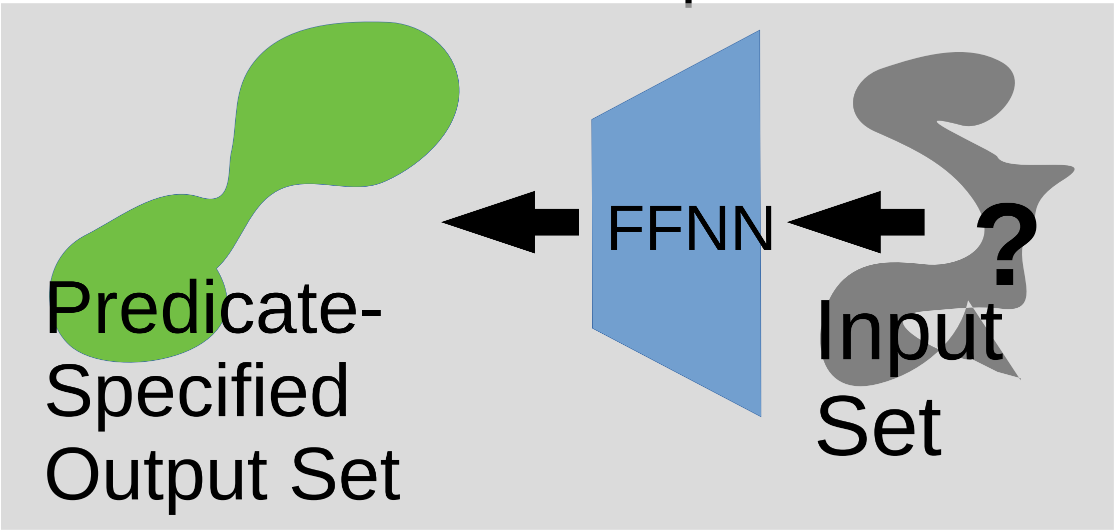


?

FFNN

Output Set

Predicate-specified Input Set



FFNN

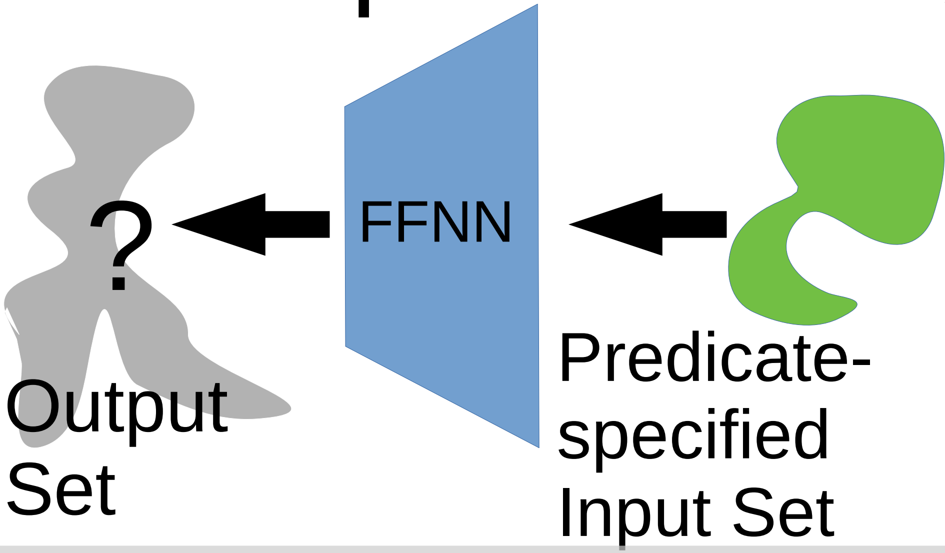
?

Predicate-Specified Output Set

Input Set

Question: “What do you do when …?”

Step 1: “Finding the Other Set”



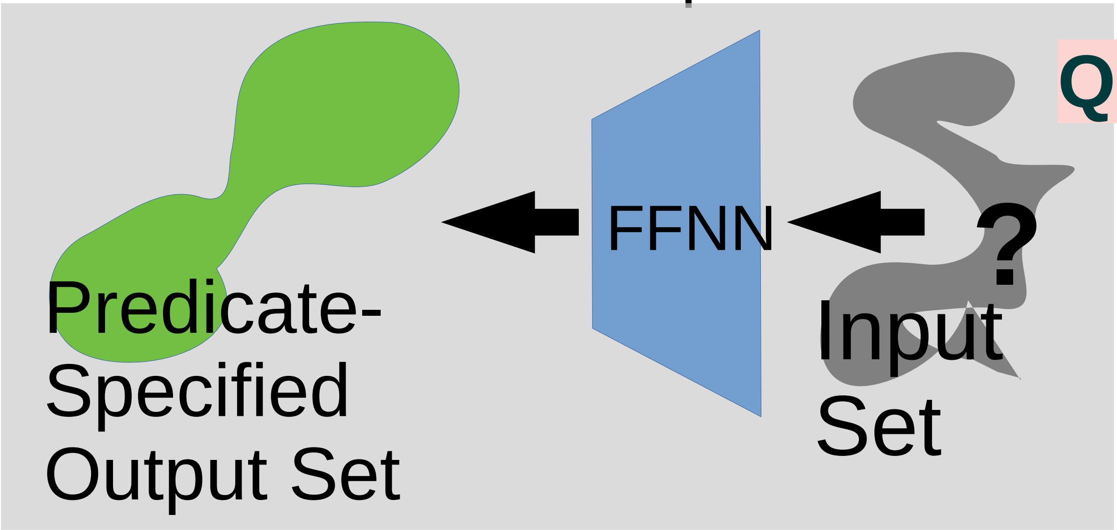
?

FFNN

Output Set

Predicate-specified Input Set

Question: “What do you do when …?”



FFNN

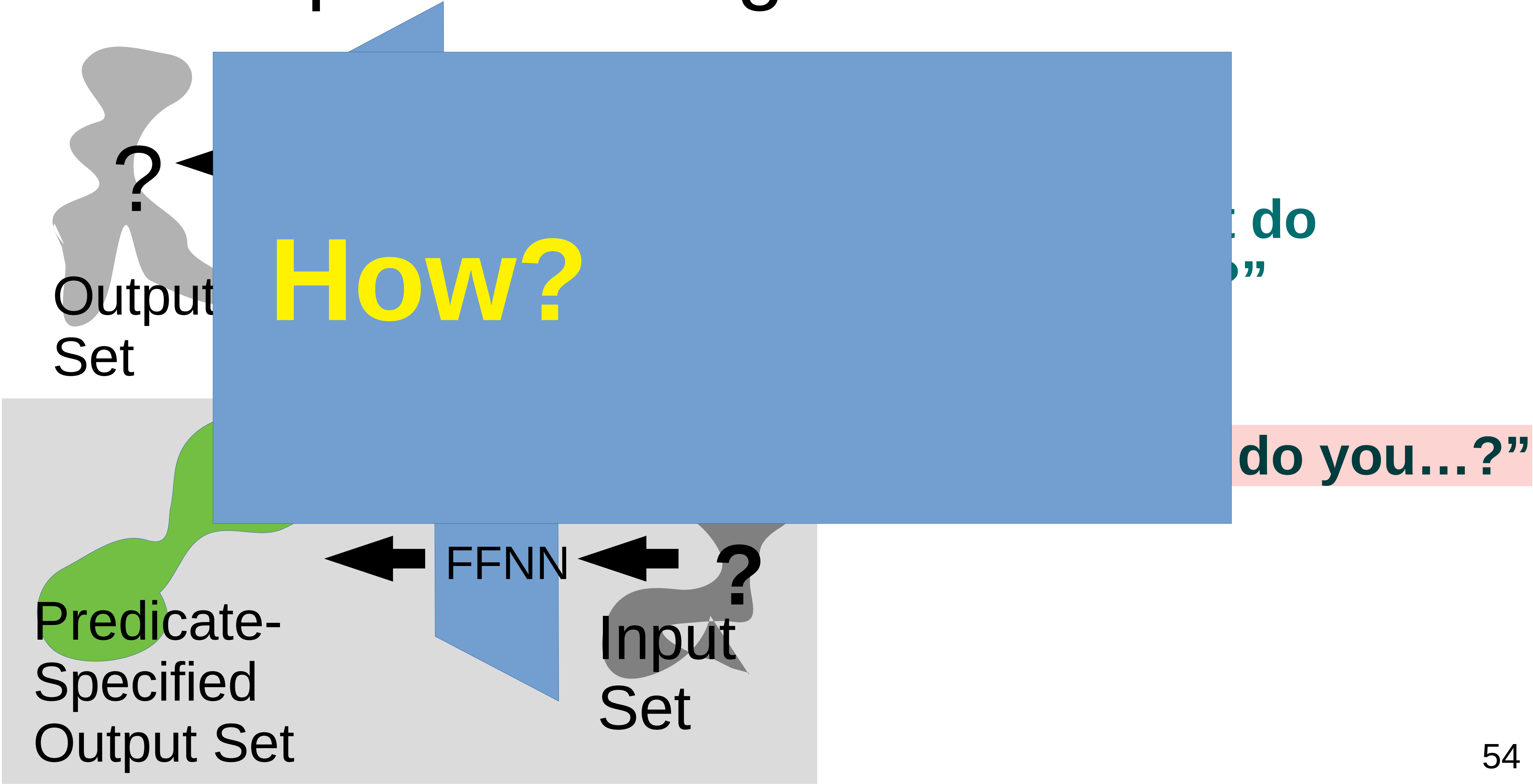
?

Predicate-Specified Output Set

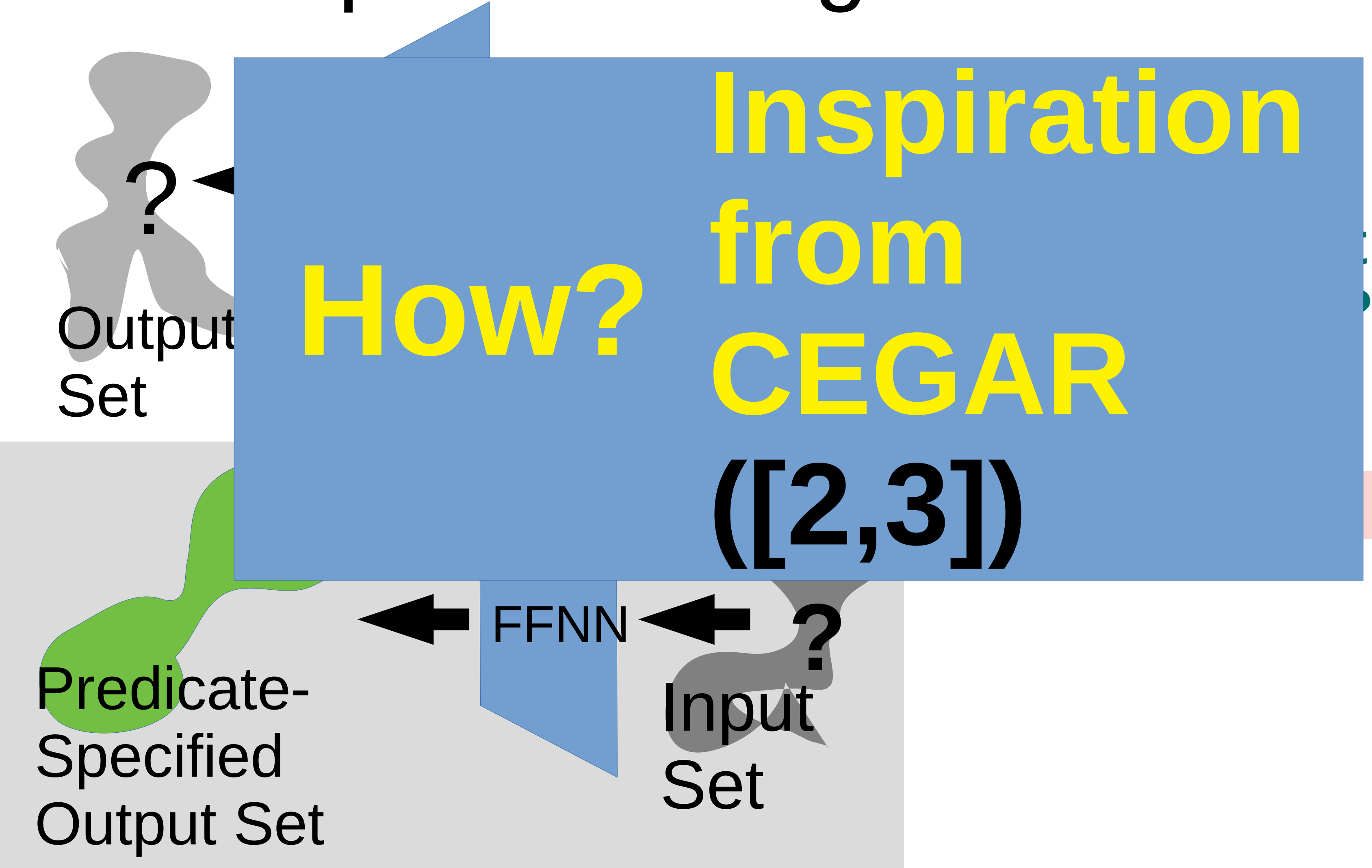
Input Set

Question: “When do you…?”

Step 1: “Finding the Other Set”



Step 1: “Finding the Other Set”



How?

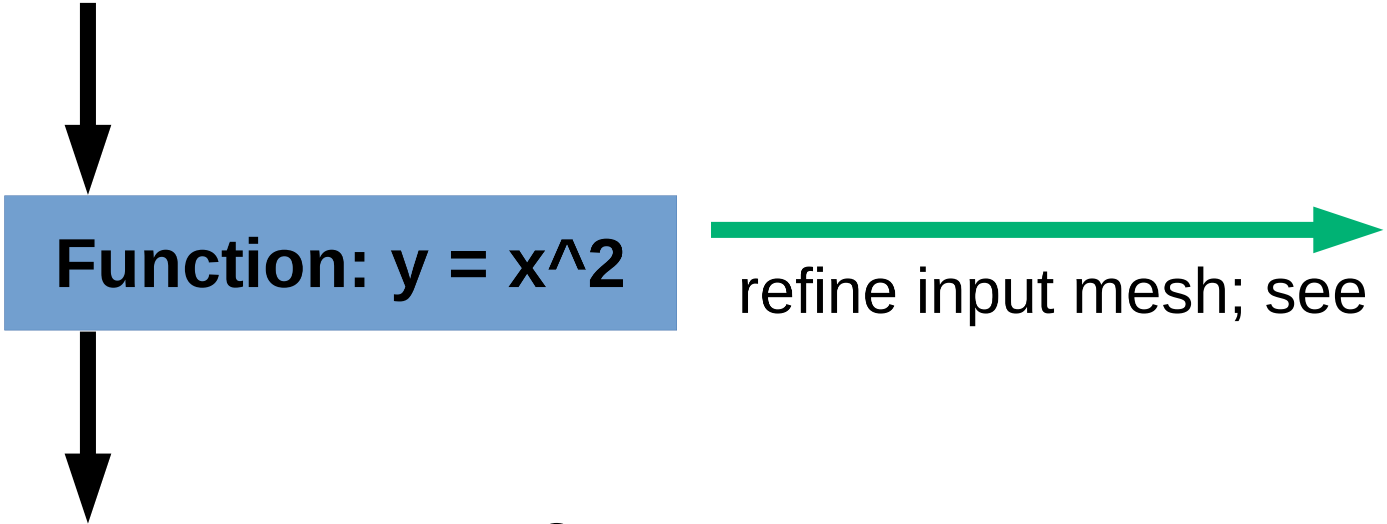
Inspiration from CEGAR ([2,3])

t do
?”

do you…?”

CEGAR-esque Method

- CEGAR solution: dynamically refine based on property you want to check for
- For us, used hyper-cubes as the abstraction



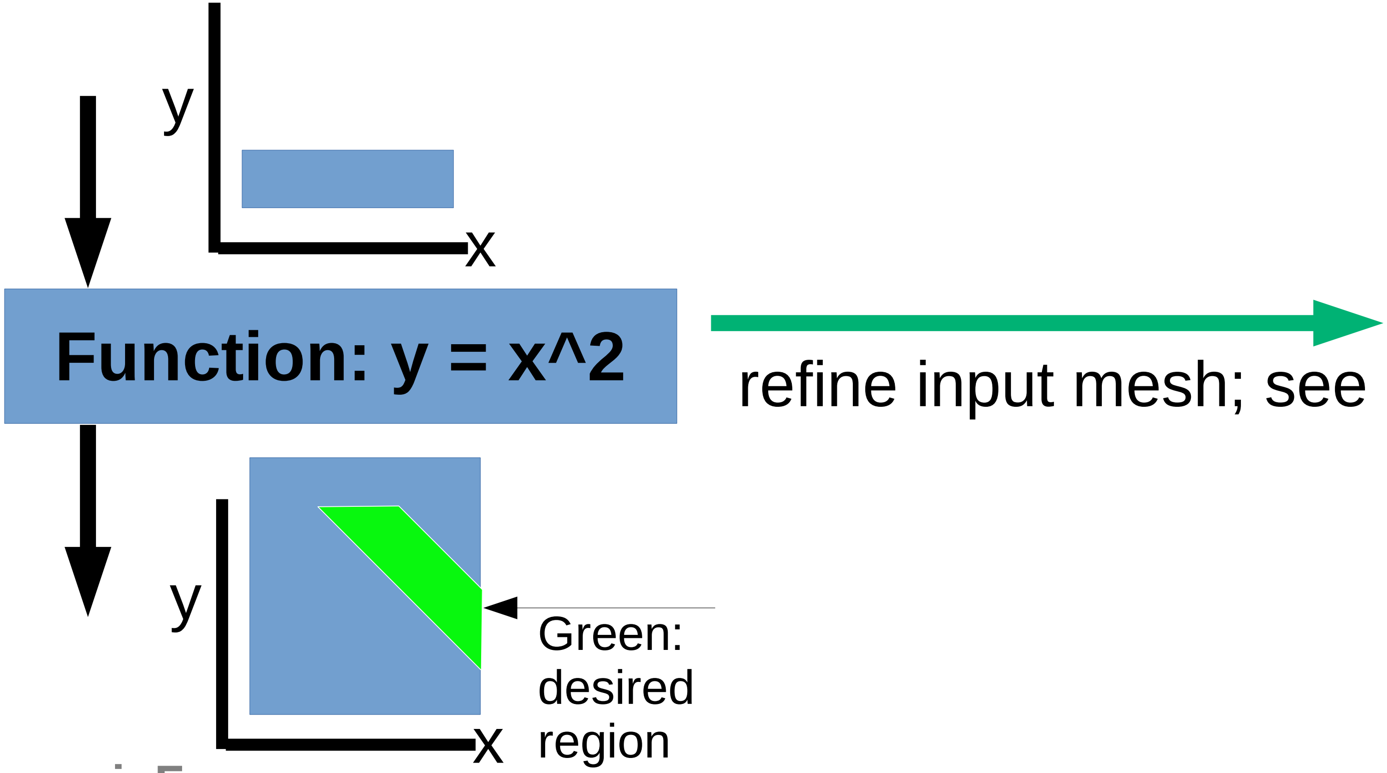
Function: y = x^2

refine input mesh; see paper for details

Green:
desired
region

CEGAR-esque Method

- CEGAR solution: dynamically refine based on property you want to check for
- For us, used hyper-cubes as the abstraction



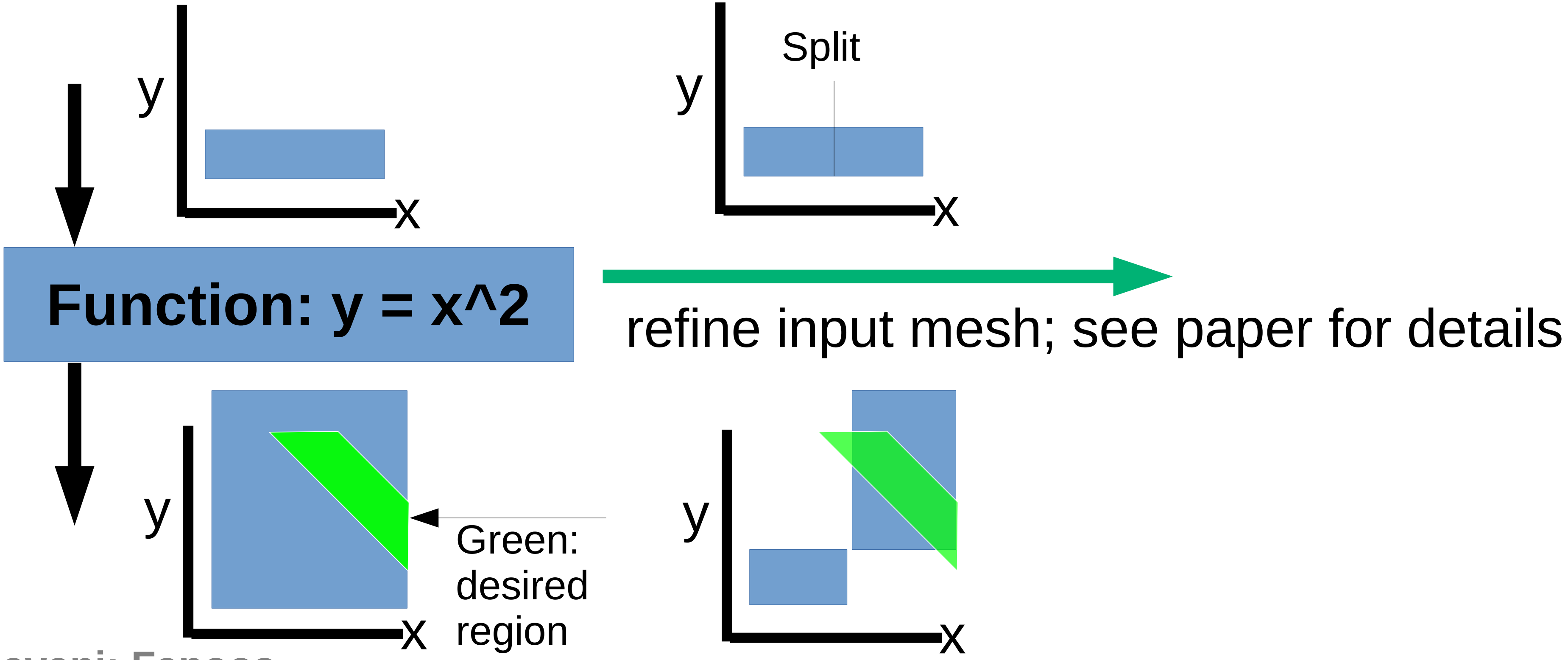
Function: y = x^2

refine input mesh; see paper for details

Green:
desired
region

CEGAR-esque Method

- CEGAR solution: dynamically refine based on property you want to check for
- For us, used hyper-cubes as the abstraction



Split

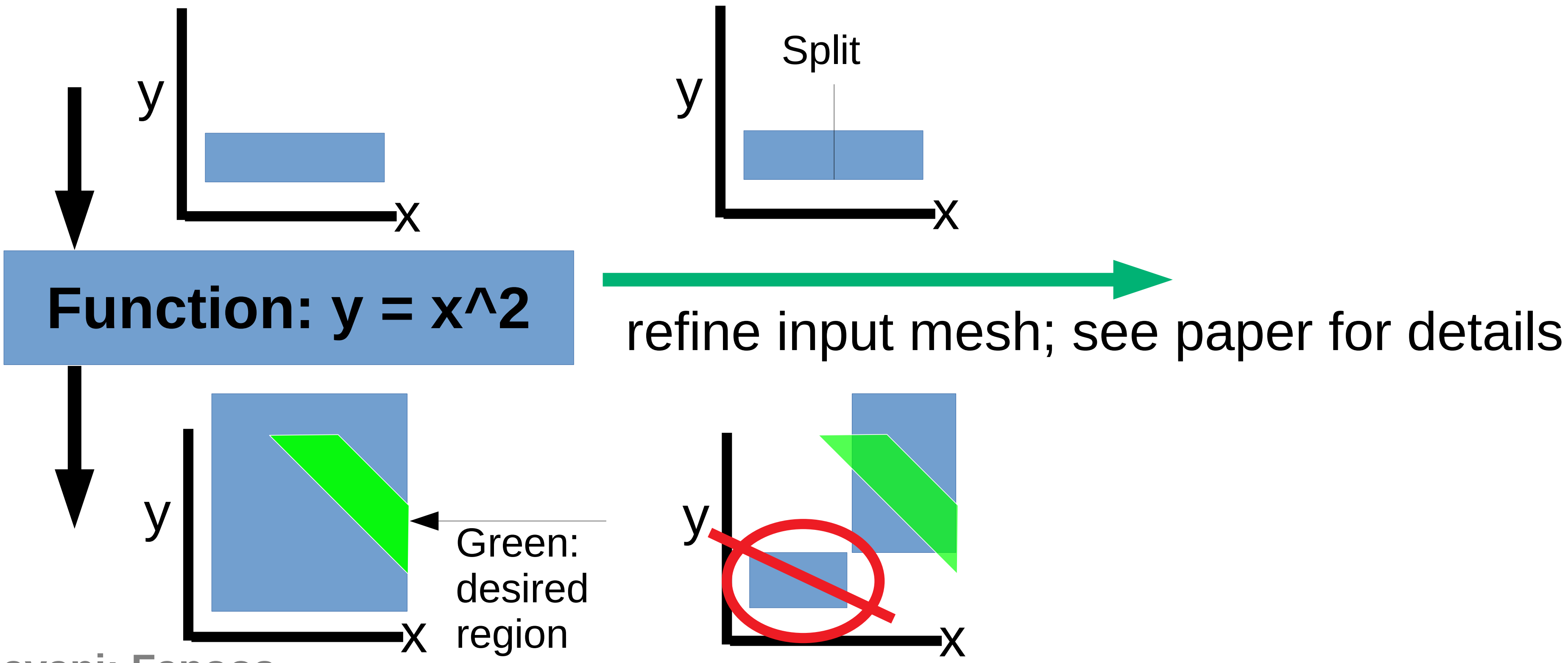
Function: y = x^2

refine input mesh; see paper for details

Green: desired region

CEGAR-esque Method

- CEGAR solution: dynamically refine based on property you want to check for
- For us, used hyper-cubes as the abstraction



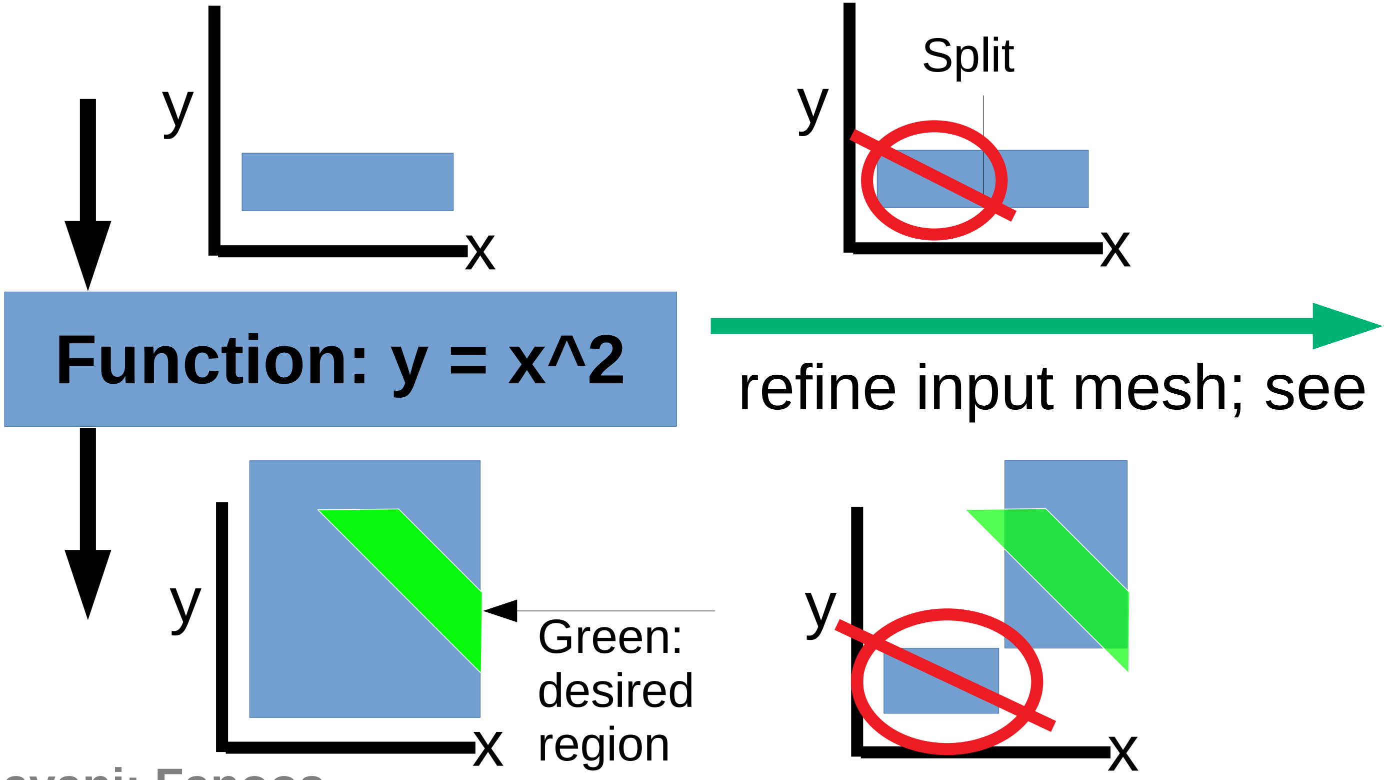
Function: y = x^2

refine input mesh; see paper for details

Green: desired region

CEGAR-esque Method

- CEGAR solution: dynamically refine based on property you want to check for
- For us, used hyper-cubes as the abstraction

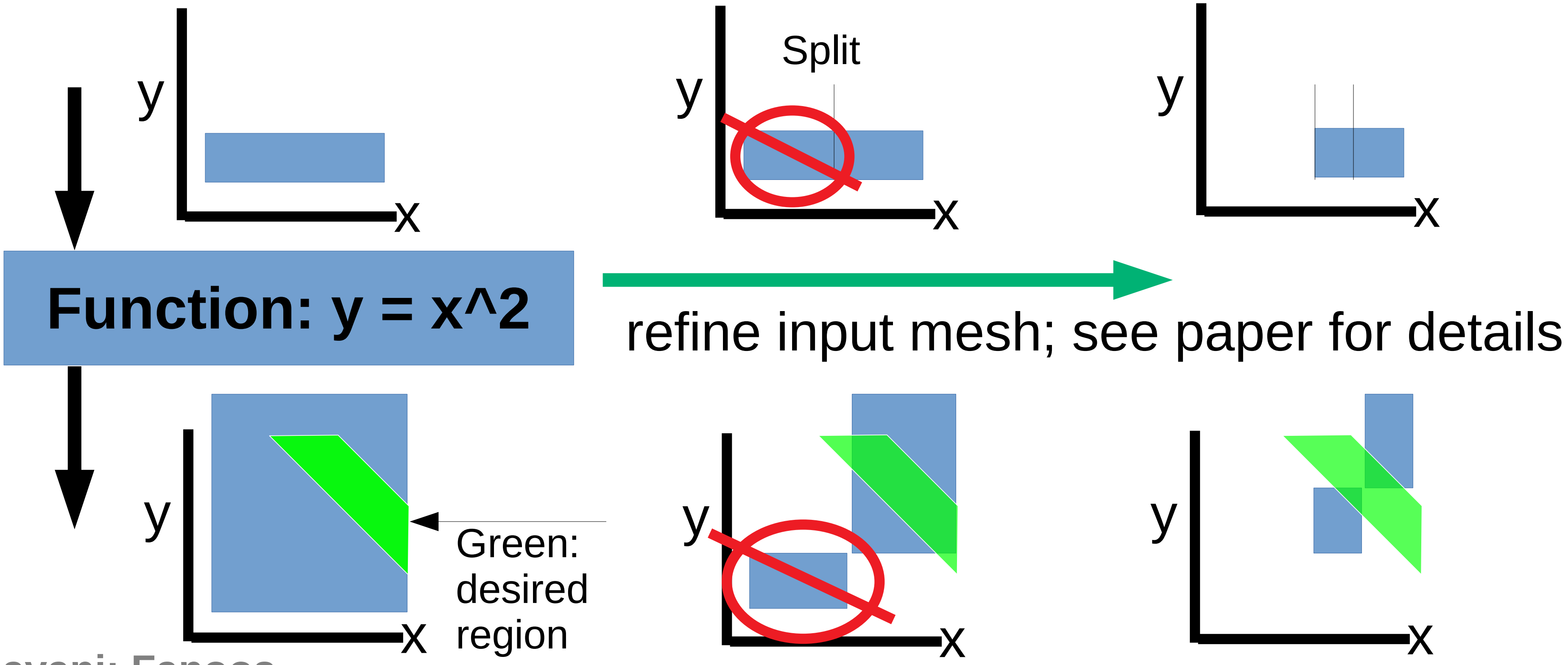


Function: y = x^2

refine input mesh; see paper for details

CEGAR-esque Method

- CEGAR solution: dynamically refine based on property you want to check for
- For us, used hyper-cubes as the abstraction



Function: y = x^2

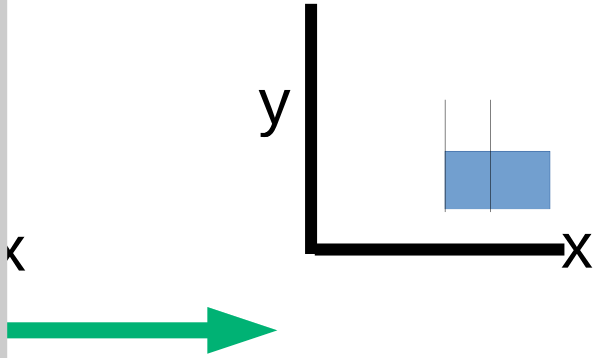
refine input mesh; see paper for details

Green: desired region

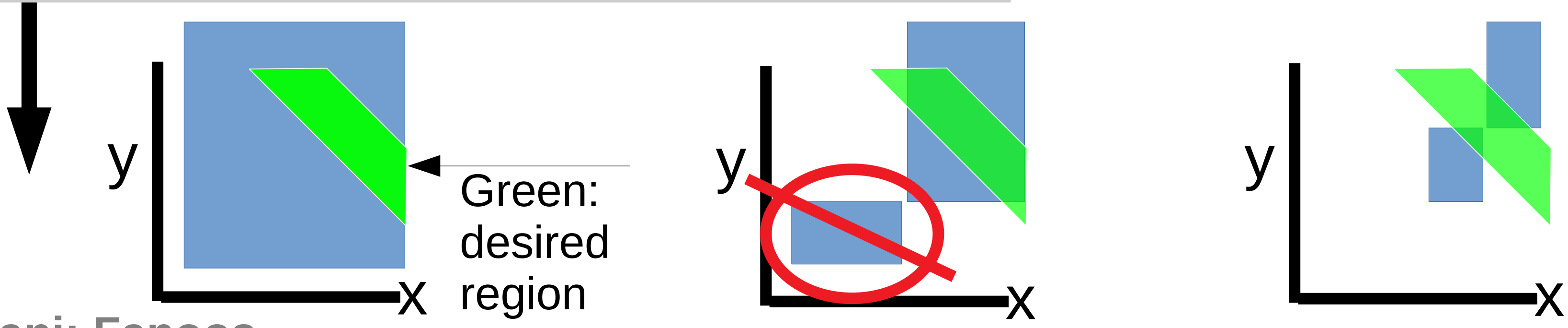
CEGAR-esque Method

n property you want to check for

Fanoos:
(1) bi-/tri-sect longest (normalized) axis,
(2) continue to refine as desired *while overlapping with* user's predicate-described region



esh; see paper for details



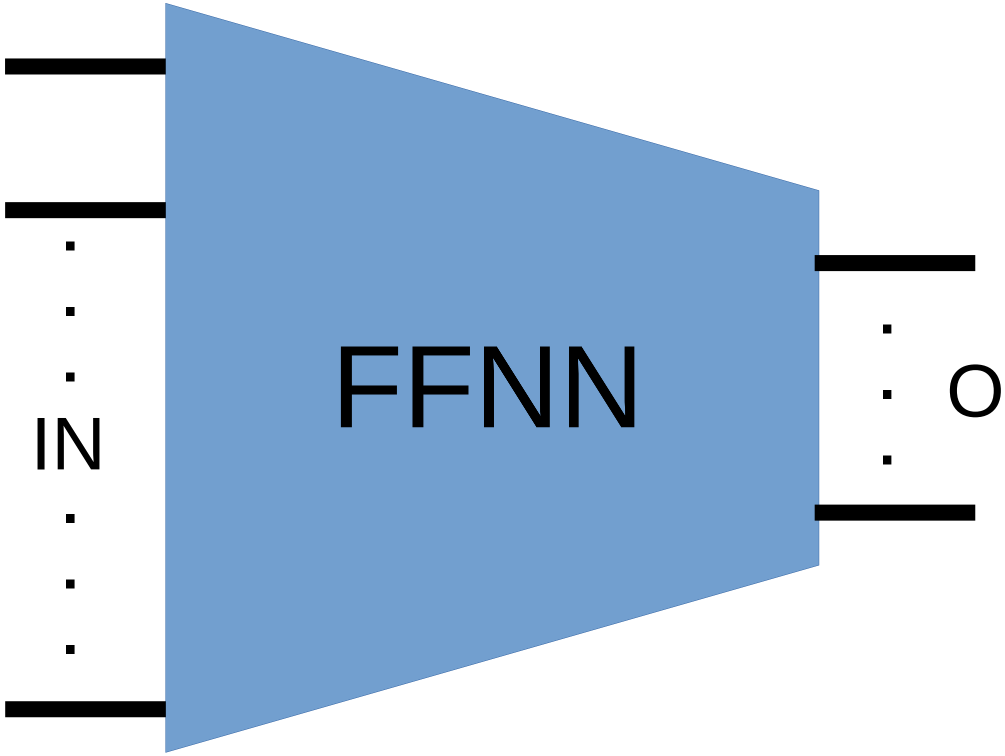
Green: desired region

Getting One Box In / Out of System

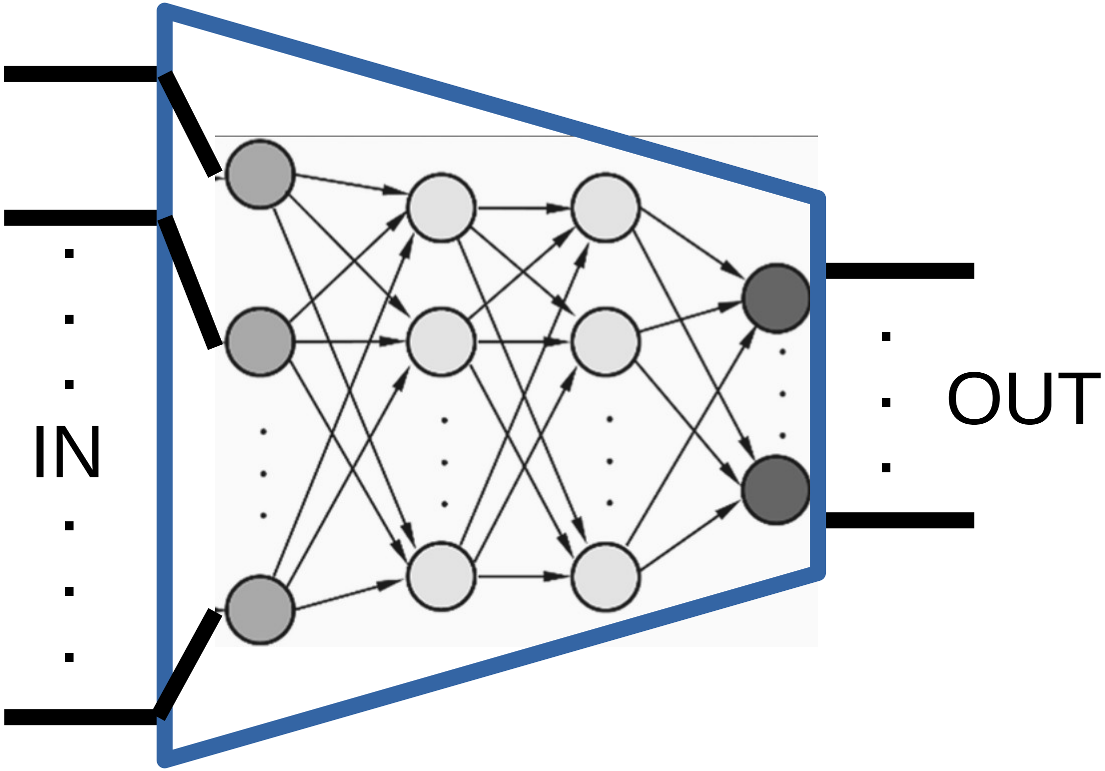
- Will discuss process for neural net (NN): process similar for other systems

Getting One Box In / Out of System

- Will discuss process for neural net (NN): process similar for other systems

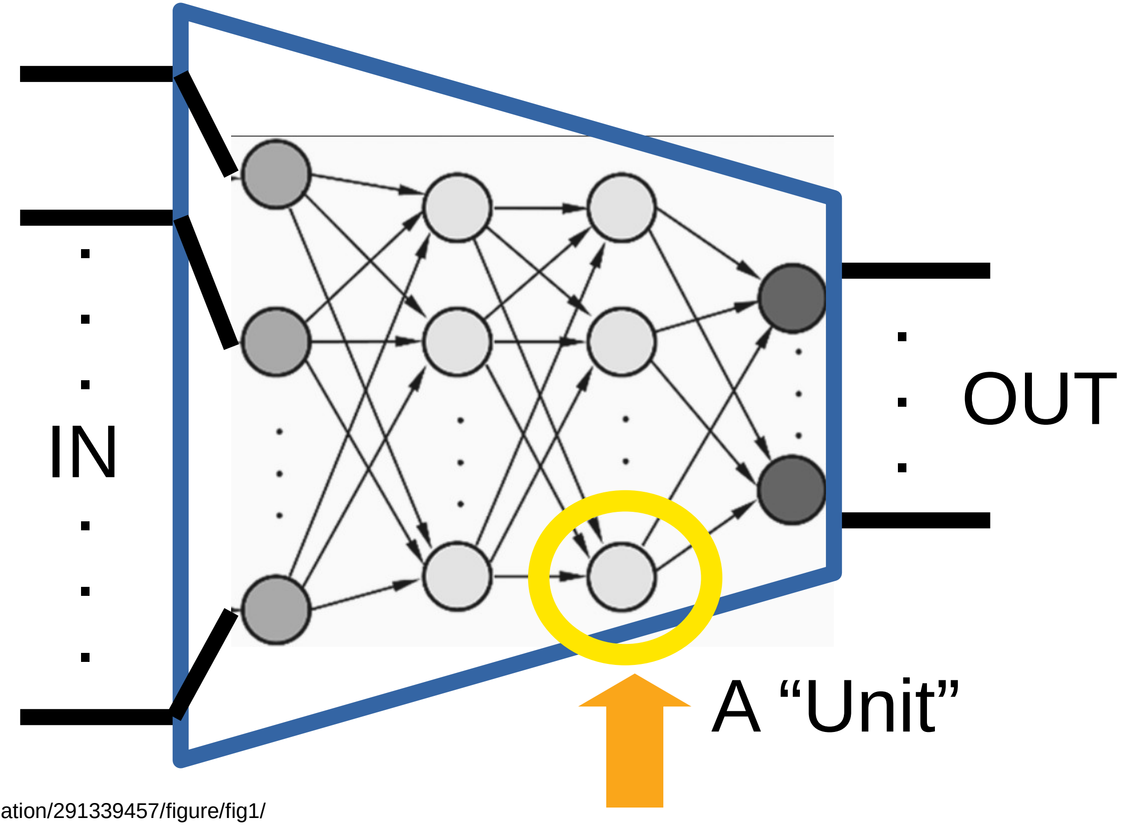


Getting One Box In / Out of System



https://www.researchgate.net/profile/Maria-Haritou/publication/291339457/figure/fig1/AS:334187497312256@1456687916125/Typical-structure-of-a-feed-forward-multilayer-neural-network.png

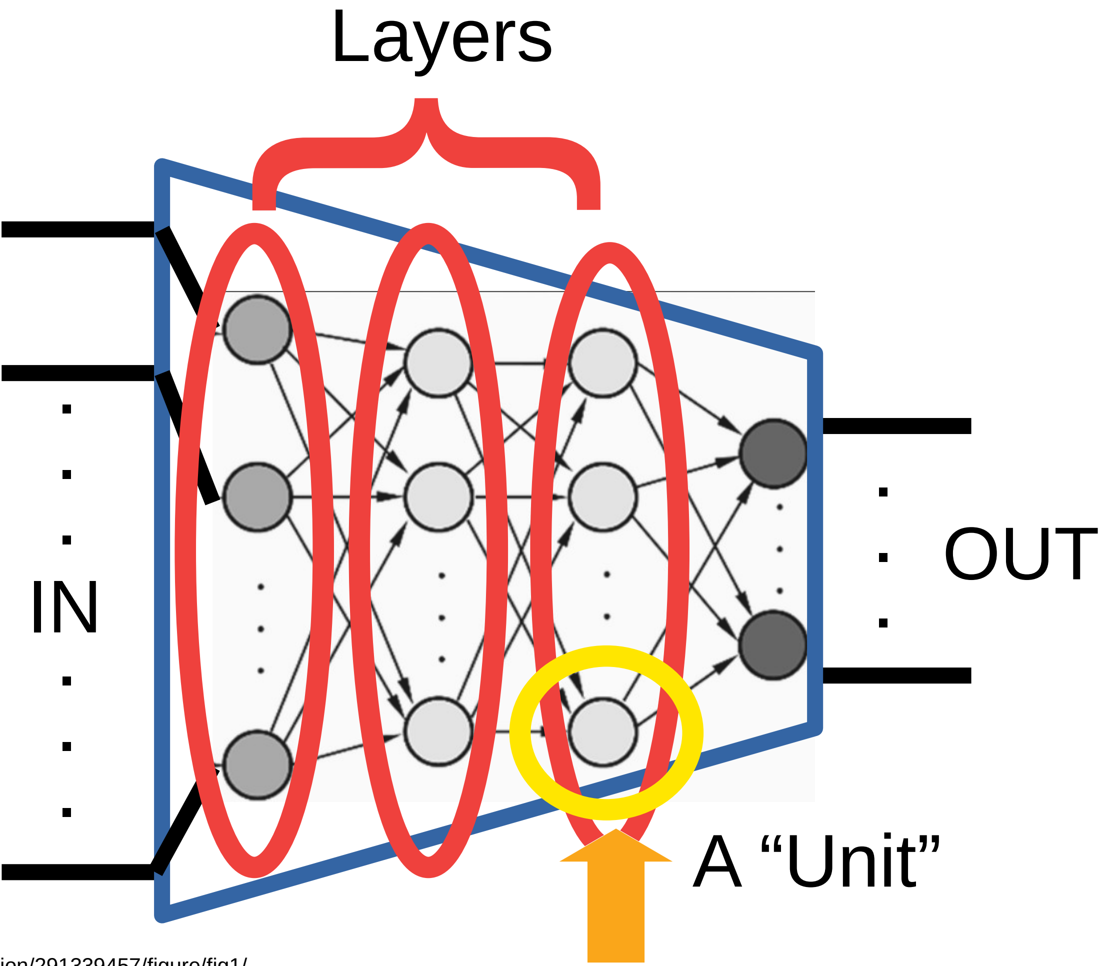
Getting One Box In / Out of System



D. Bayani: Fanoos

https://www.researchgate.net/profile/Maria-Haritou/publication/291339457/figure/fig1/AS:334187497312256@1456687916125/Typical-structure-of-a-feed-forward-multilayer-neural-network.png

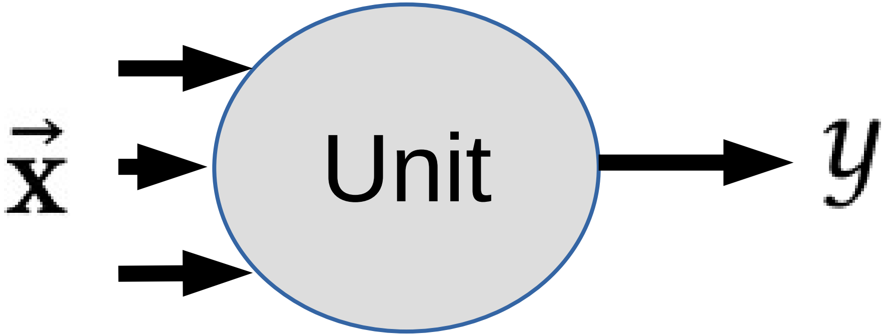
Getting One Box In / Out of System



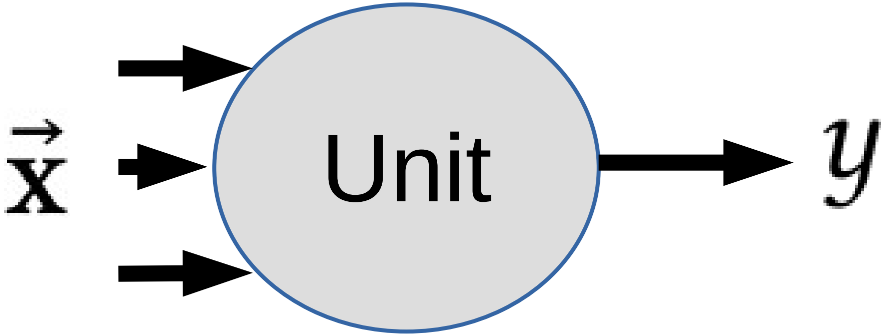
D. Bayani: Fanoos

https://www.researchgate.net/profile/Maria-Haritou/publication/291339457/figure/fig1/
AS:334187497312256@1456687916125/Typical-structure-of-a-feed-forward-multilayer-neural-network.png

Getting One Box In / Out of System, Cnt.

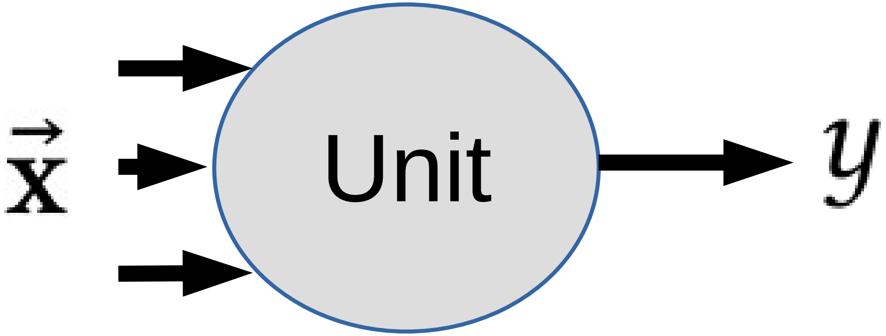


Getting One Box In / Out of System, Cnt.



$$y = \rho(T(\vec{\mathbf{x}}))$$

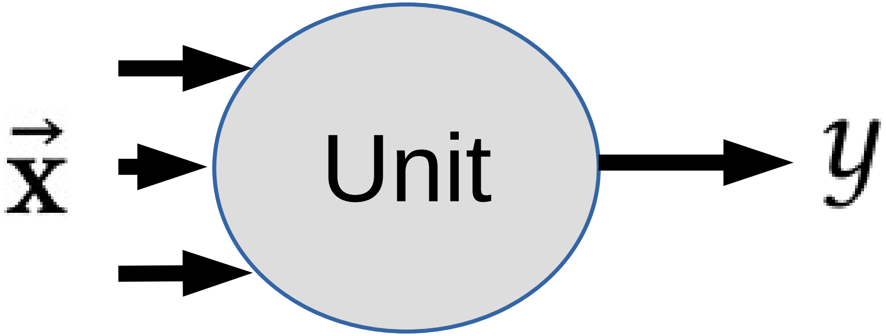
Getting One Box In / Out of System, Cnt.



$$y = \rho(T(\vec{\mathbf{x}}))$$

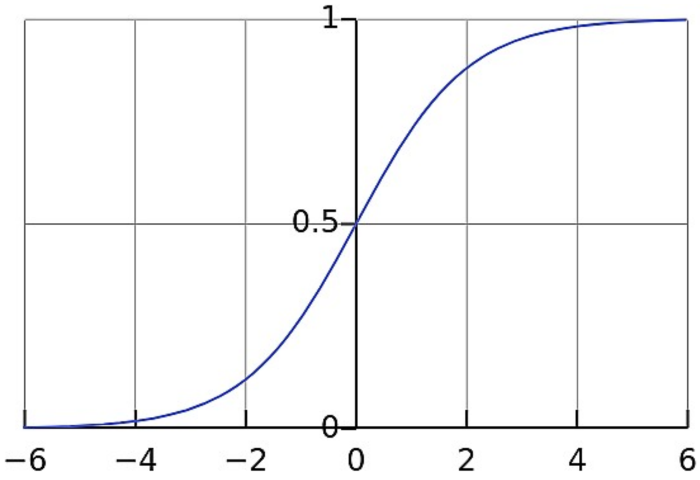
Affine transform $T(\vec{\mathbf{x}}) = \langle \vec{\mathbf{w}}, \vec{\mathbf{x}} \rangle + \mathbf{b}$

Getting One Box In / Out of System, Cnt.



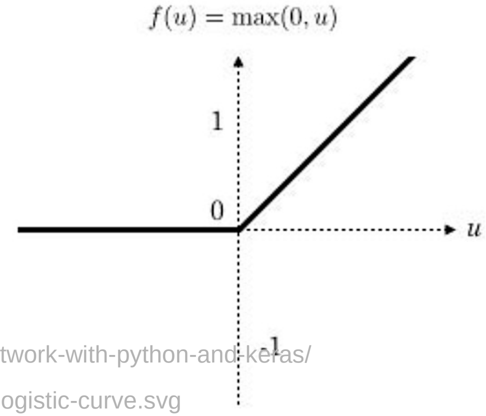
$$y = \rho(T(\vec{\mathbf{x}}))$$

Affine transform $T(\vec{\mathbf{x}}) = \langle \vec{\mathbf{w}}, \vec{\mathbf{x}} \rangle + \mathbf{b}$



An “**activation function**”

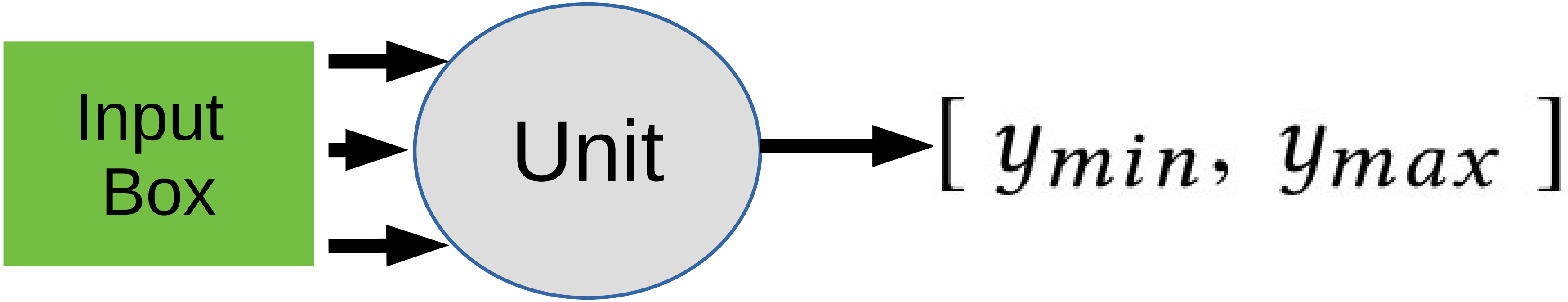
Here, a non-decreasing function from R to R.



https://ehackz.com/2018/03/17/build-your-first-neural-network-with-python-and-keras/

https://upload.wikimedia.org/wikipedia/commons/8/88/Logistic-curve.svg

Getting One Box In / Out of System, Cnt.



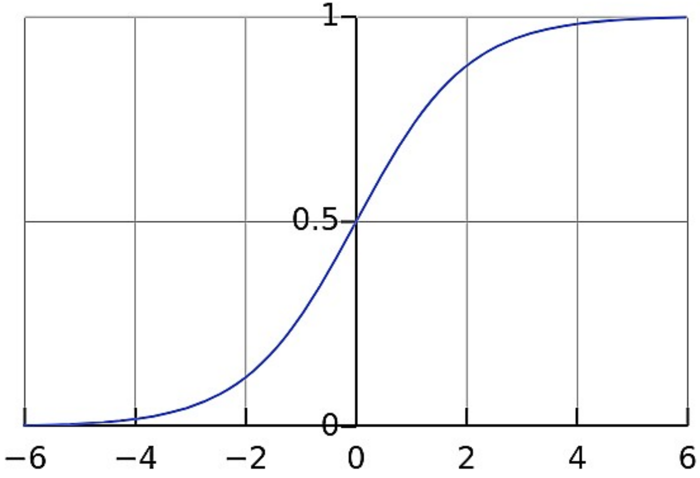
Input Box

Unit

$[\, y_{min},\ y_{max} \,]$

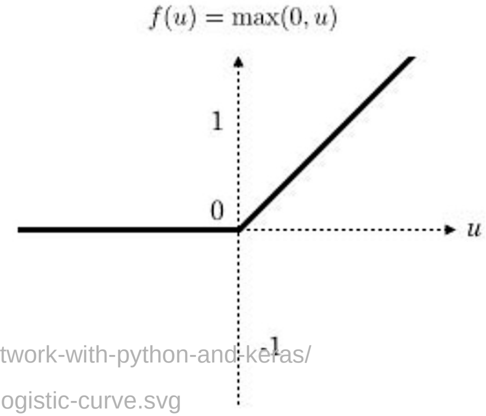
$$y = \rho(T(\vec{\mathbf{x}}))$$

Affine transform $T(\vec{\mathbf{x}}) = \langle \vec{\mathbf{w}}, \vec{\mathbf{x}} \rangle + \mathbf{b}$



An “**activation function**”

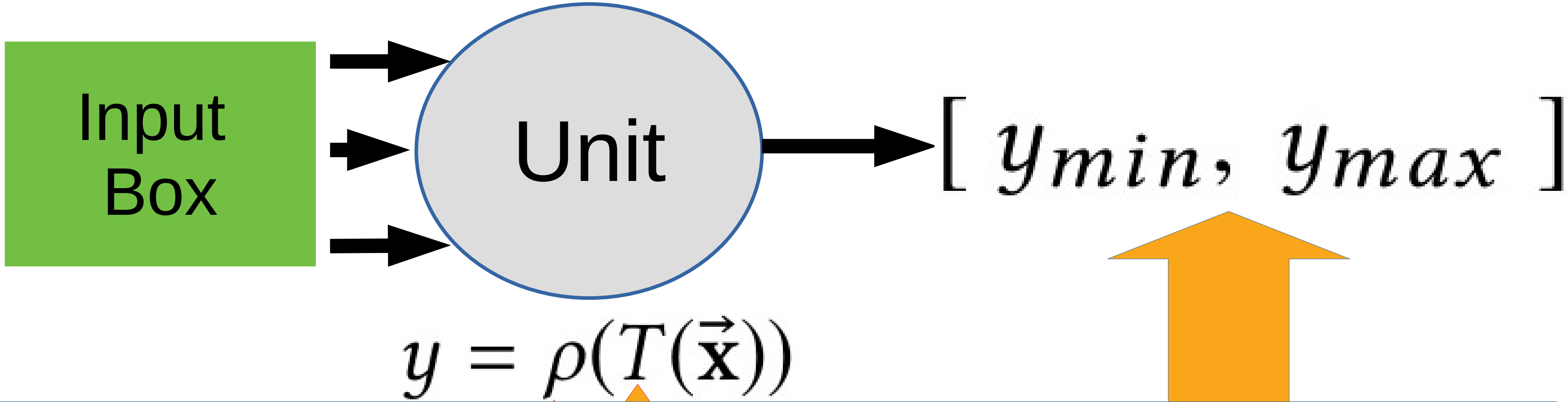
Here, a non-decreasing function from R to R.



https://ehackz.com/2018/03/17/build-your-first-neural-network-with-python-and-keras/

https://upload.wikimedia.org/wikipedia/commons/8/88/Logistic-curve.svg

Getting One Box In / Out of System, Cnt.



$$y = \rho(T(\vec{\mathbf{x}}))$$

Can compute in an ***exact*** closed form

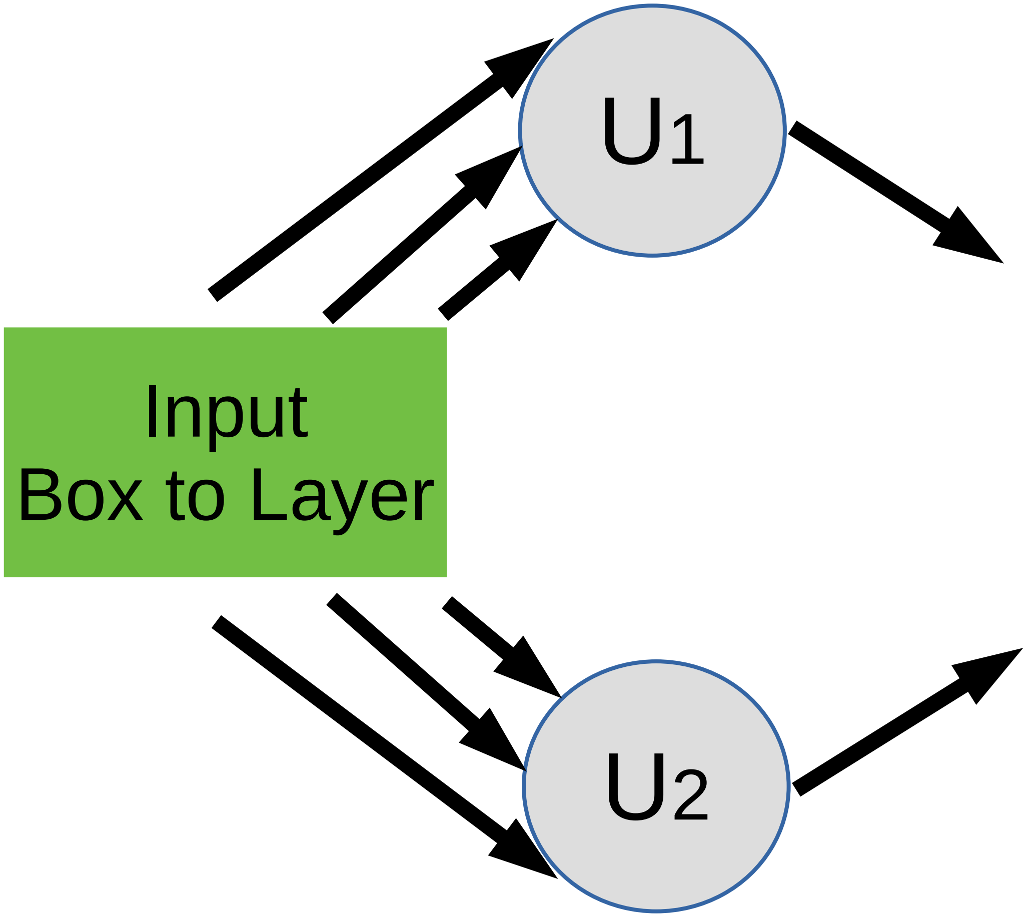
https://upload.wikimedia.org/wikipedia/commons/8/88/Logistic-curve.svg

Getting One Box In / Out of System, Cnt.

Propagating Through Layers

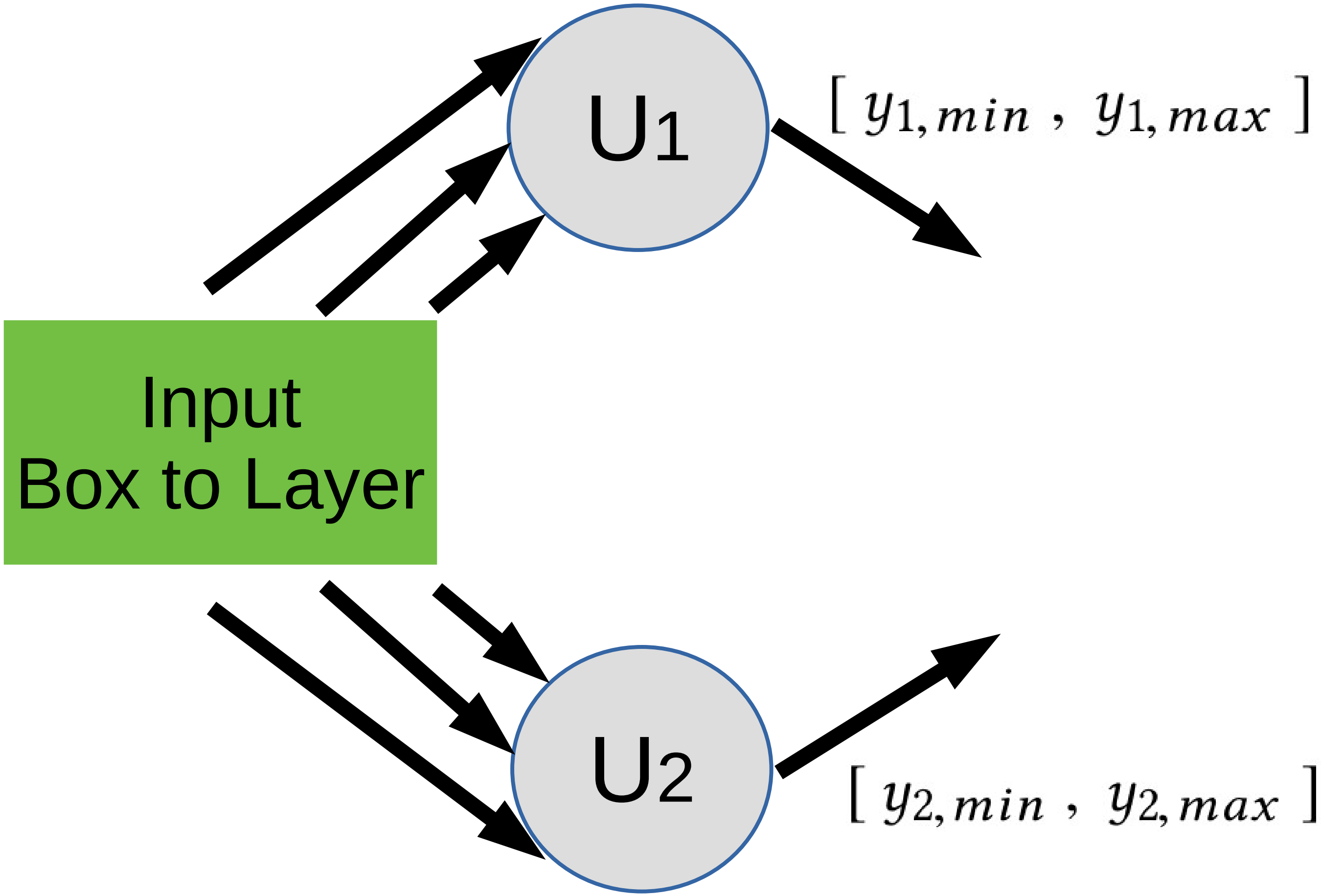
Getting One Box In / Out of System, Cnt.

Propagating Through Layers



Getting One Box In / Out of System, Cnt.

Propagating Through Layers



Input Box to Layer

U1

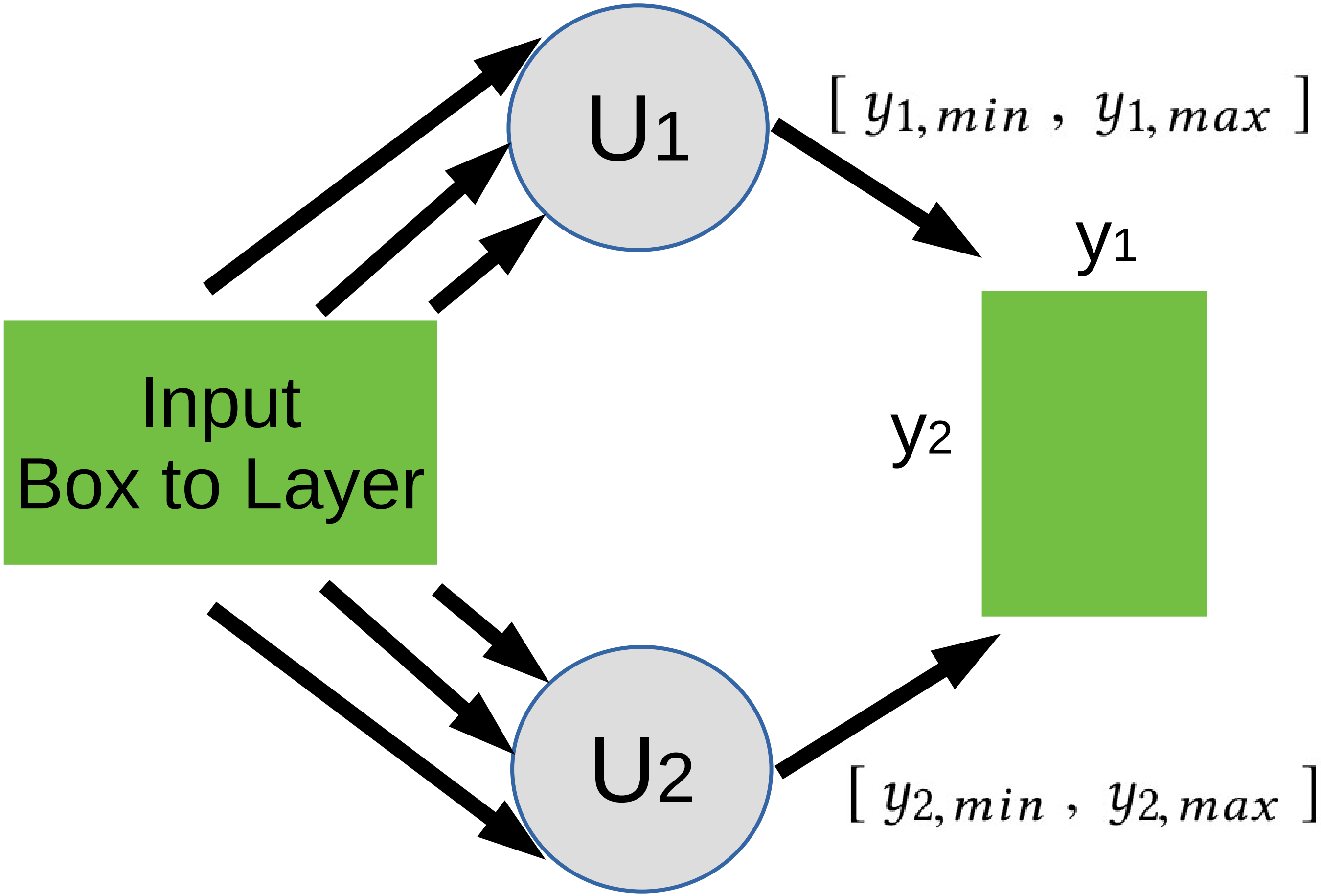
$[\, y_{1,min} \;,\; y_{1,max} \,]$

U2

$[\, y_{2,min} \;,\; y_{2,max} \,]$

Getting One Box In / Out of System, Cnt.

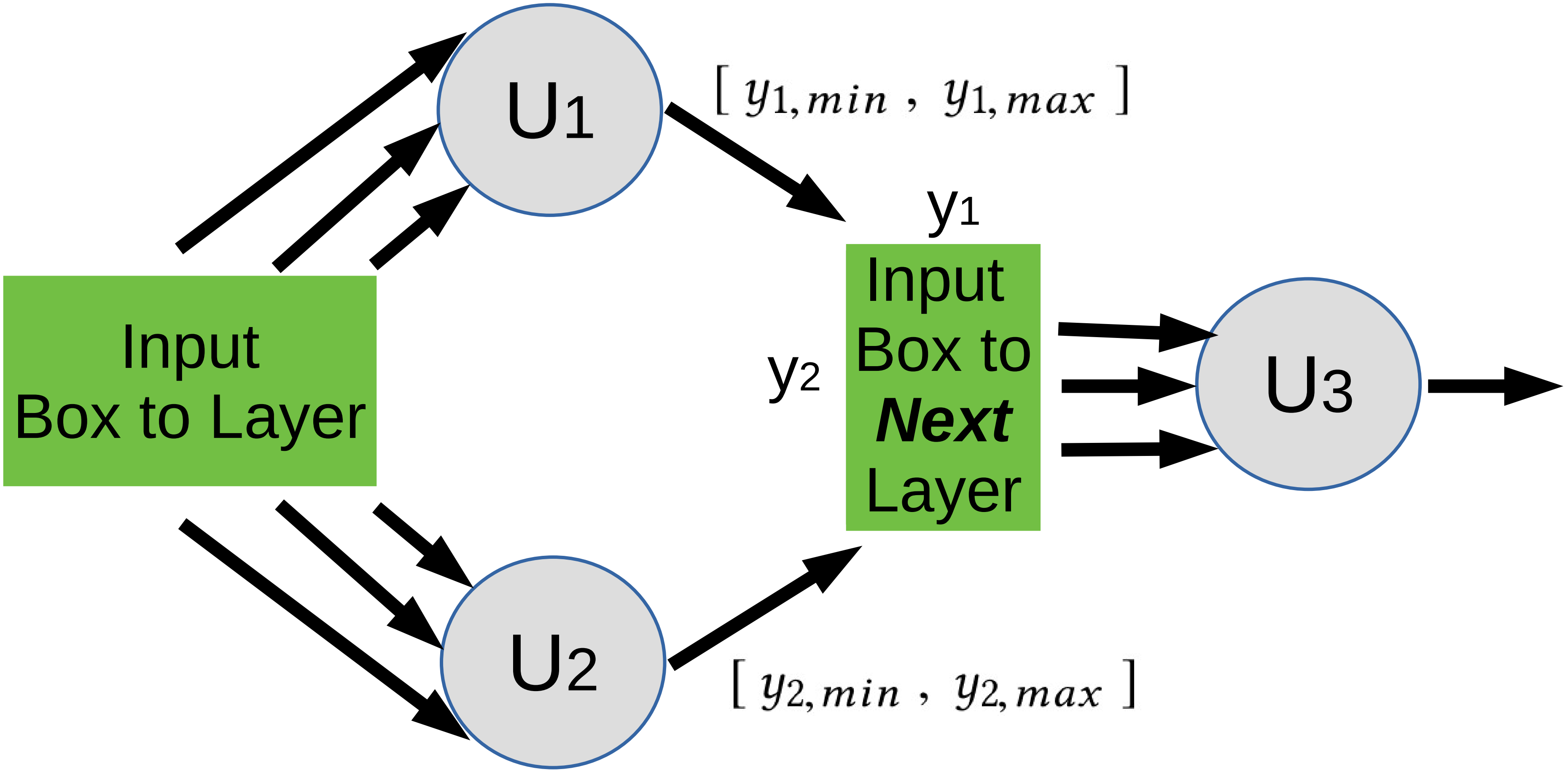
Propagating Through Layers



Input
Box to Layer
U1
U2
$[\, y_{1,min} \,,\, y_{1,max} \,]$
$[\, y_{2,min} \,,\, y_{2,max} \,]$
y1
y2

Getting One Box In / Out of System, Cnt.

Propagating Through Layers



U1
$[\, y_{1,min} \, , \, y_{1,max} \,]$
y1
Input Box to Layer
y2
Input Box to ***Next*** Layer
U3
U2
$[\, y_{2,min} \, , \, y_{2,max} \,]$

Getting One Box In / Out of System, Cnt.

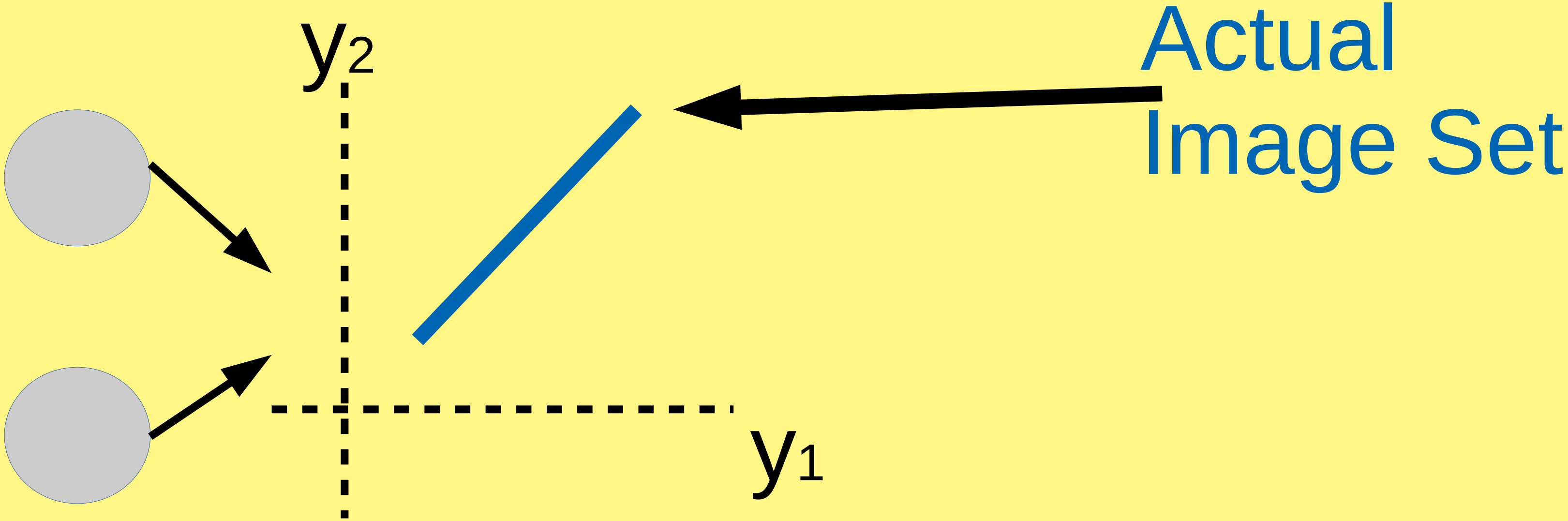
Propagating Through Layers

This is where approximation enters. Ex: consider if $y_1 = y_2$

Getting One Box In / Out of System, Cnt.

Propagating Through Layers

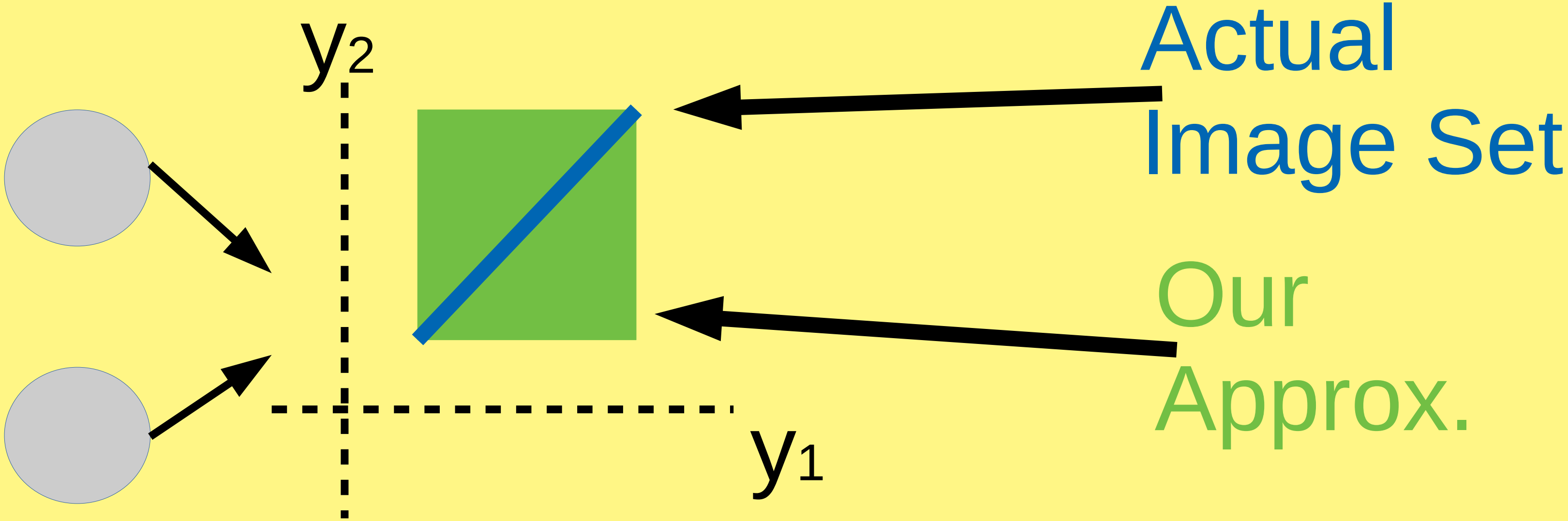
This is where approximation enters. Ex: consider if $y_1 = y_2$



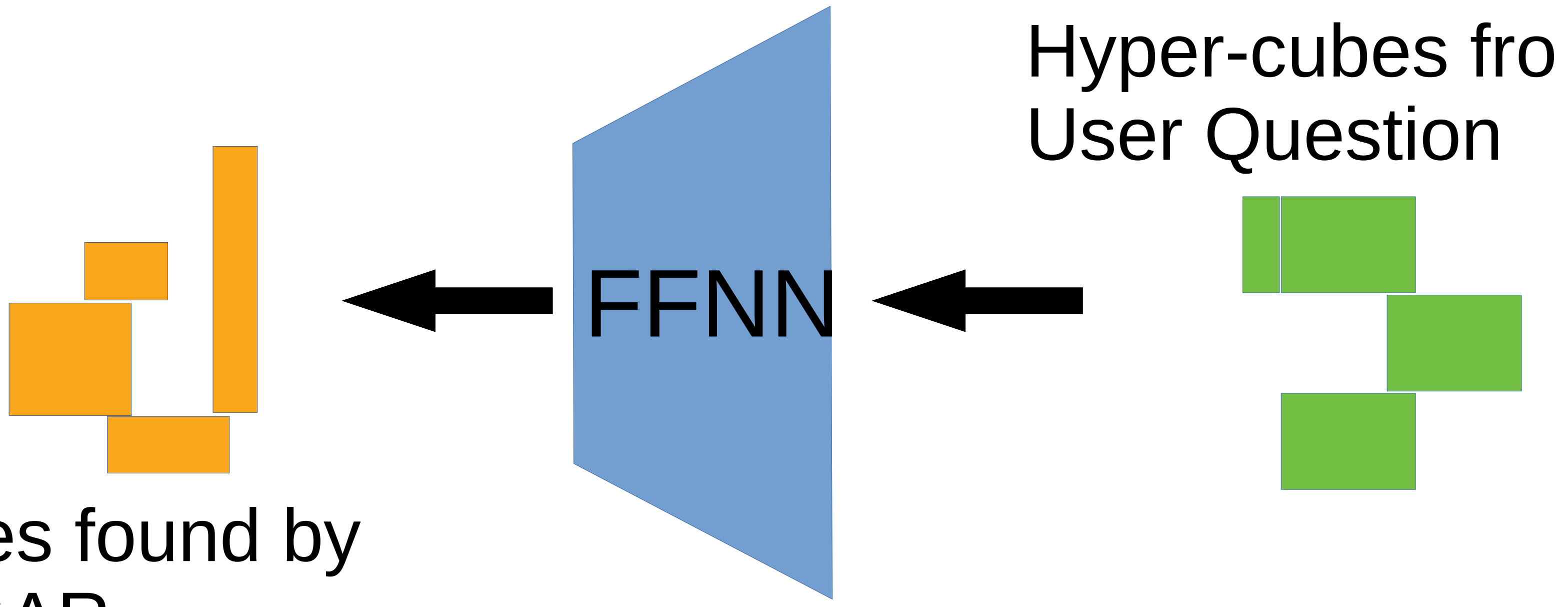
Getting One Box In / Out of System, Cnt.

Propagating Through Layers

This is where approximation enters. Ex: consider if $y_1 = y_2$



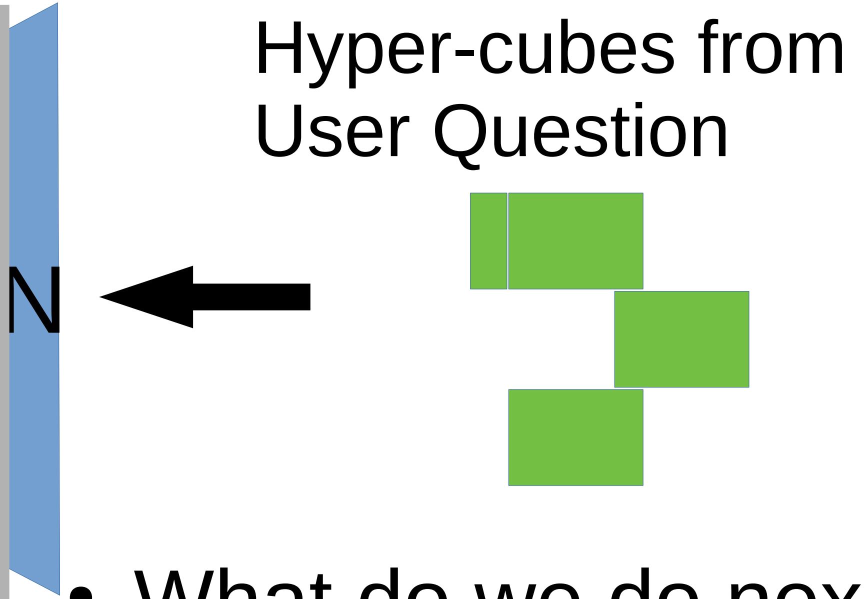
What we have so far:



- **Next: Step 2:** Describe the found boxes
  - How? Form a covering with the predicates.

What we have so far:

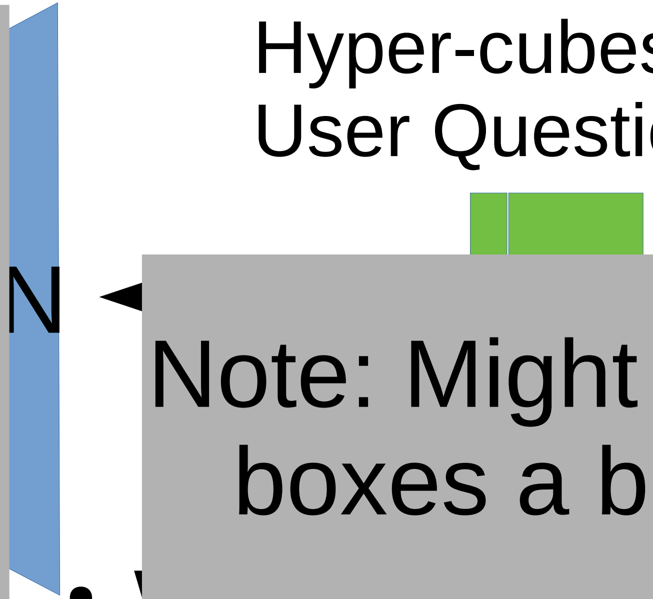
Sub-steps:
2.1) Get candidate predicates for each box
2.2) Form global covering from the candidates



- What do we do next?
  - Describe the found boxes
- How? Form a covering with the predicates.

What we have so far:

Sub-steps:
2.1) Get candidate predicates for each box
2.2) Form global covering from the candidates



Hyper-cubes from User Question

Note: Might merge boxes a bit first

- What do we do next?
  - Describe the found boxes

- How? Form a covering with the predicates.

Step 2.1: Getting Candidate Preds. For Each Box

For each box, B:

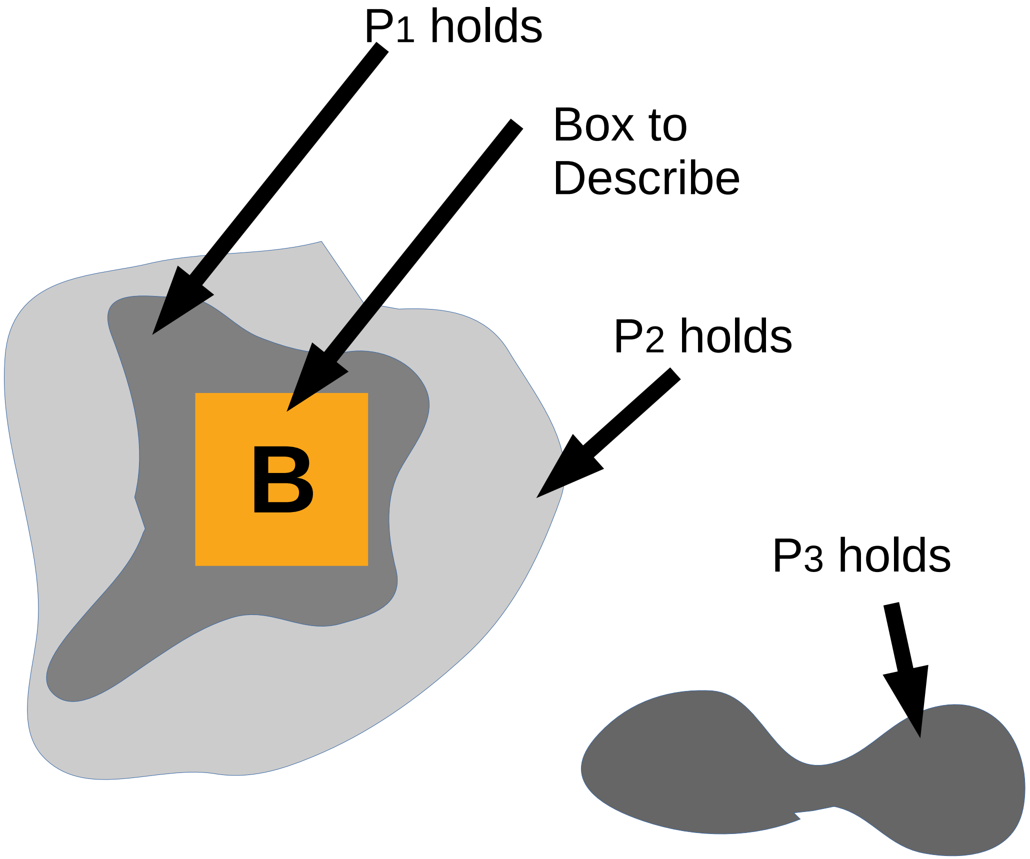
1) Get predicates that hold over B
    1. “feasibility check”: try on random sample from B first
    2. Check with SAT-Solver
2) Get most specific predicates

Step 2.1: Getting Candidate Preds. For Each Box

For each box, B:

1) Get predicates that hold over B
   1. “feasibility check”: try on random sample from B first
   2. Check with SAT-Solver

2) Get most specific predicates

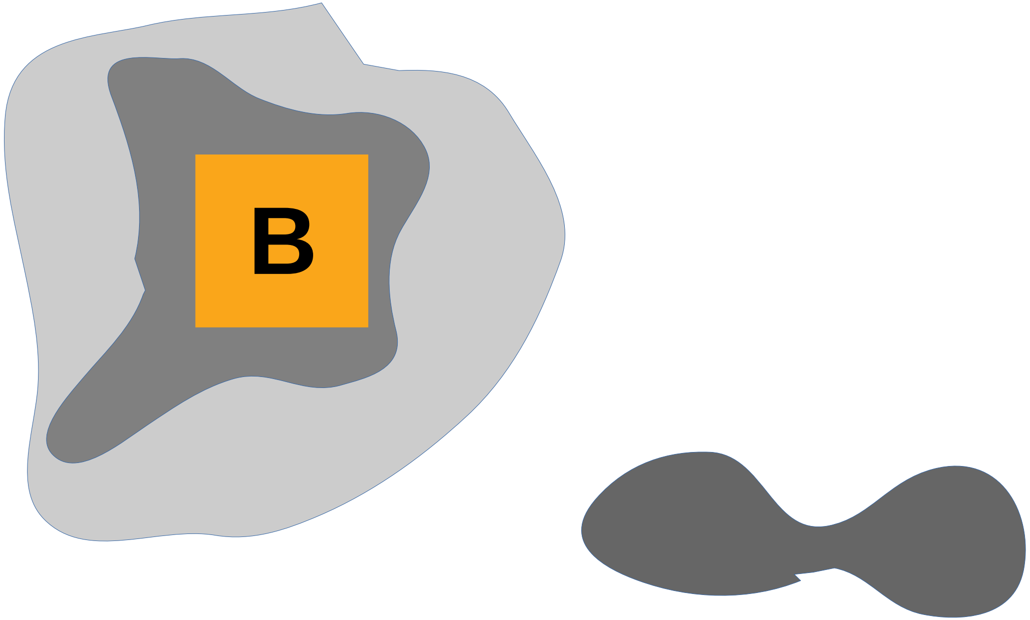


Step 2.1: Getting Candidate Preds. For Each Box

For each box, B:

1) Get predicates that hold over B
   1. “feasibility check”: try on random sample from B first
   2. Check with SAT-Solver

2) Get most specific predicates

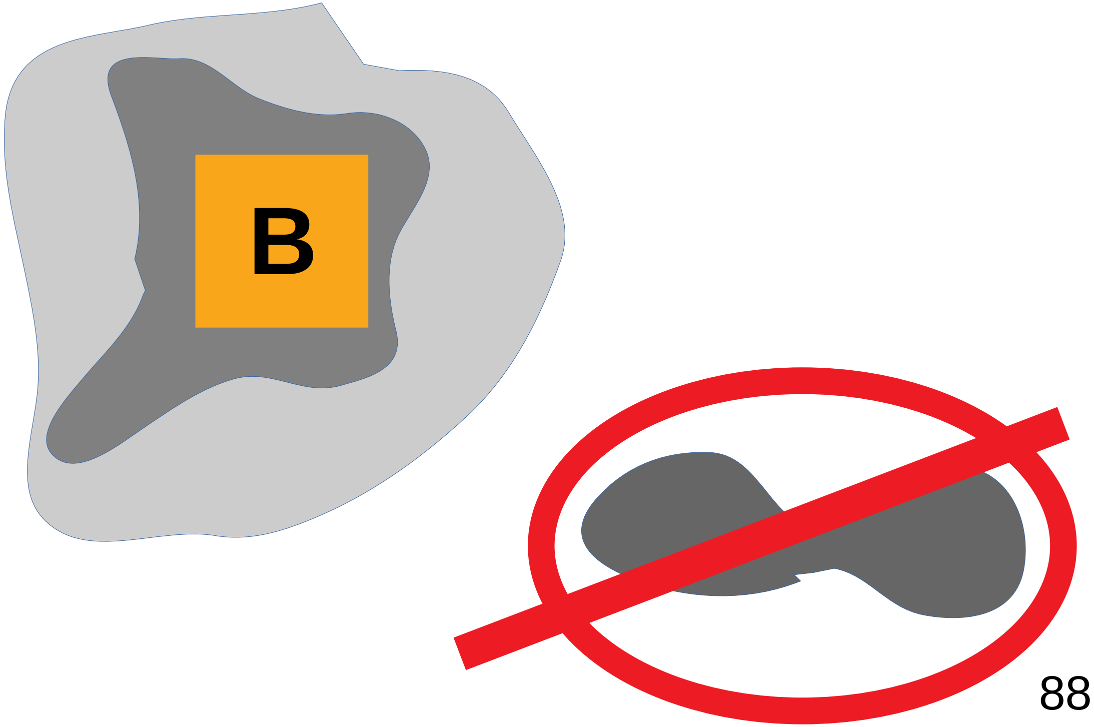


Step 2.1: Getting Candidate Preds. For Each Box

For each box, B:

1) Get predicates that hold over B
   1. “feasibility check”: try on random sample from B first
   2. Check with SAT-Solver

2) Get most specific predicates

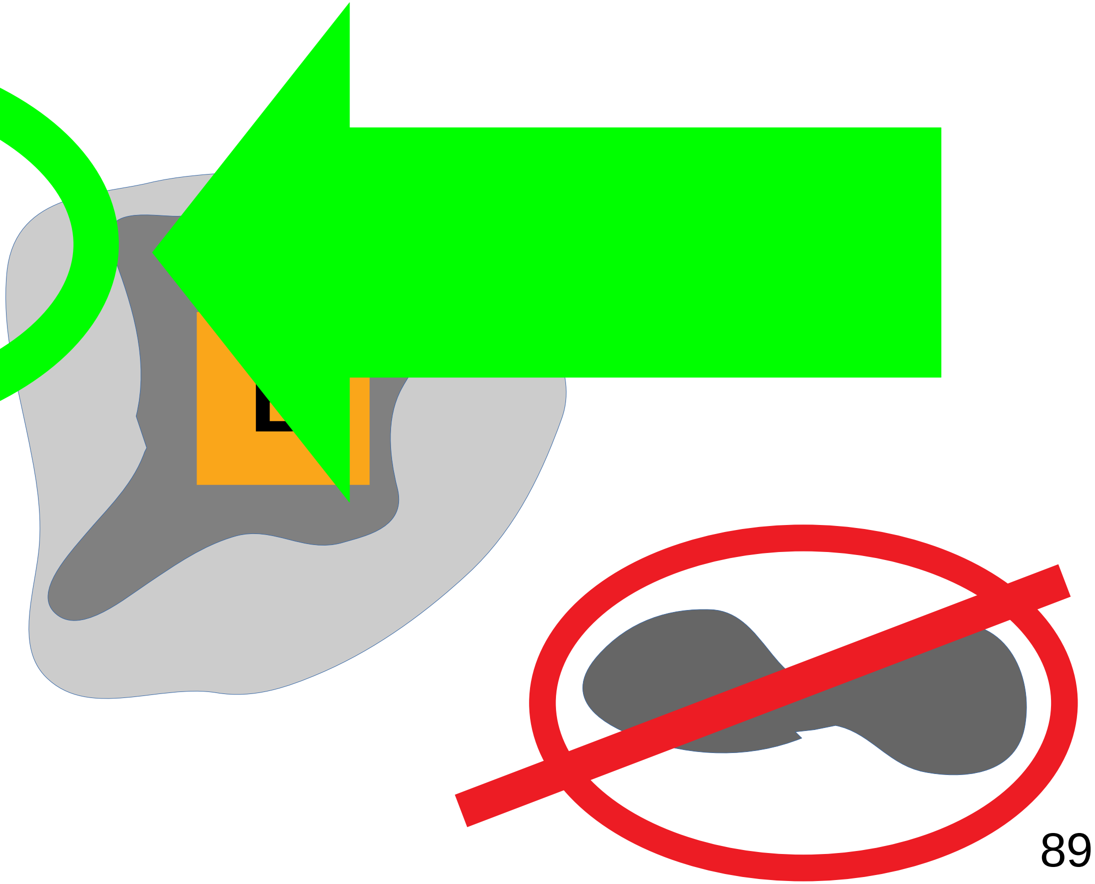


Step 2.1: Getting Candidate Preds. For Each Box

For each box, B:

1) Get predicates that hold over B, **“H(B)”**
   1. “feasibility check”: try on random samples from B first
   2. Check with SAT-Solver

2) Get most specific predicates

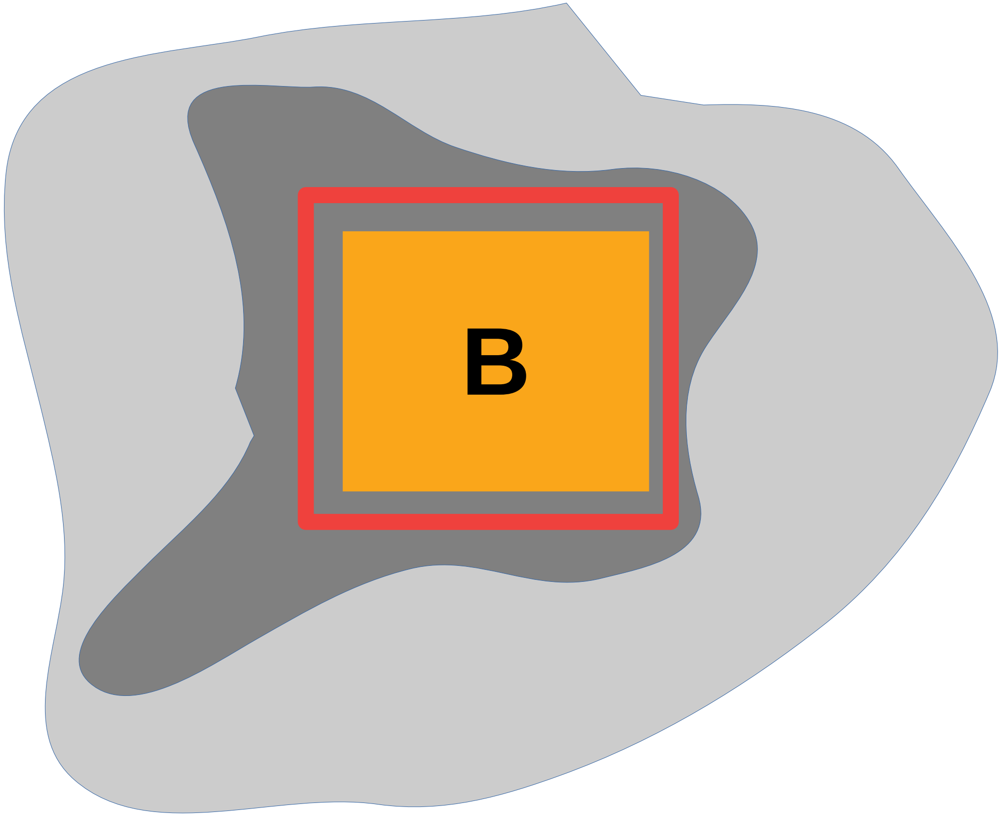


Step 2.1: Getting Candidate Preds. For Each Box

For each box, B:

1) Get predicates that hold over B, **“H(B)”**
    1. “feasibility check”: try on random sample from B first
    2. Check with SAT-Solver

2) Get most specific predicates

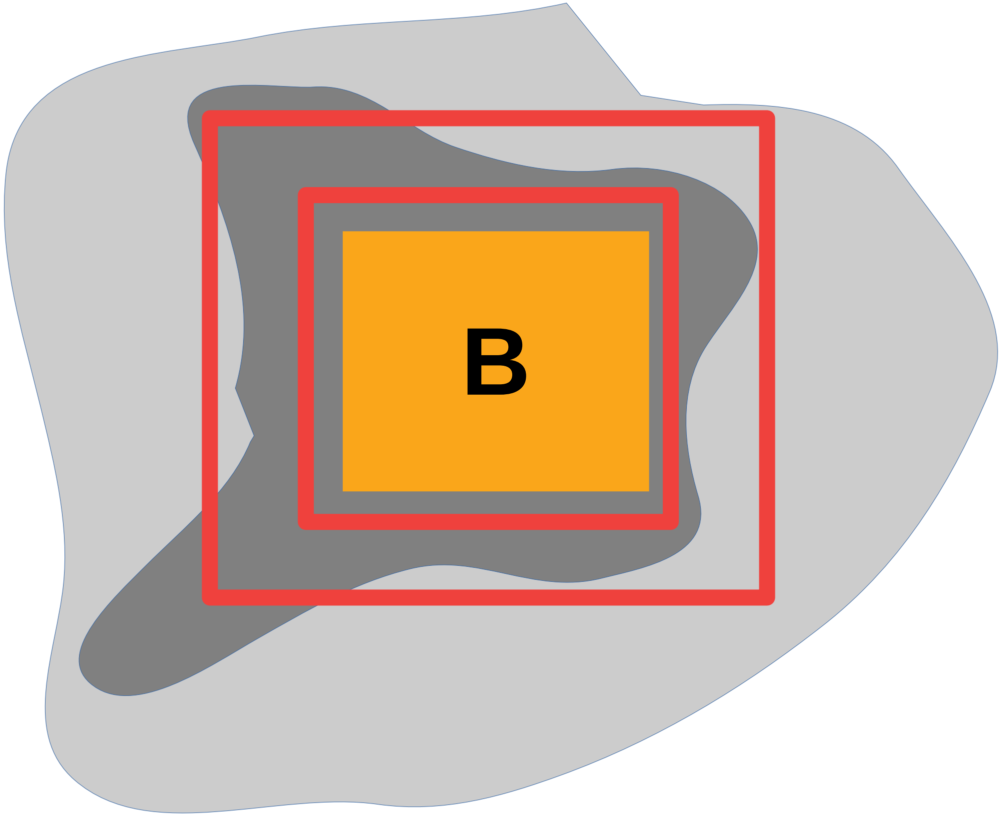


Step 2.1: Getting Candidate Preds. For Each Box

For each box, B:

1) Get predicates that hold over B, **“H(B)”**
   1. “feasibility check”: try on random sample from B first
   2. Check with SAT-Solver

2) Get most specific predicates

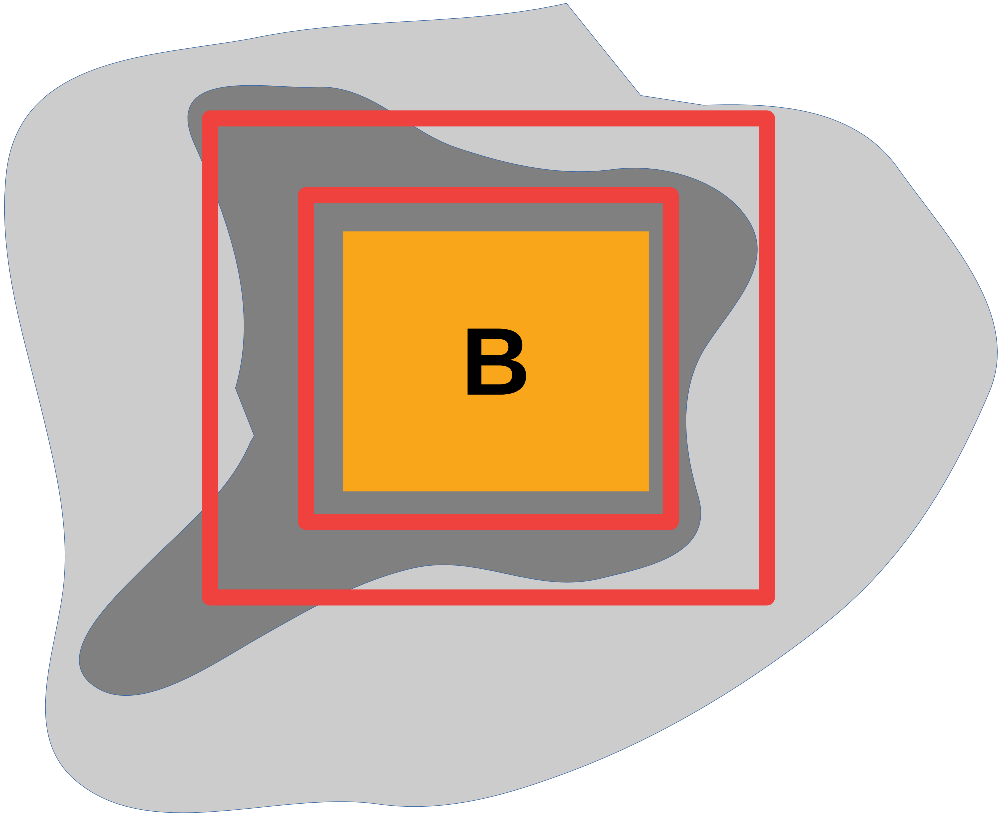


Step 2.1: Getting Candidate Preds. For Each Box

For each box, B:

1) Get predicates that hold over B, **“H(B)”**
   1. “feasibility check”: try on random sample from B first
   2. Check with SAT-Solver

2) Get most specific predicates



Note: Can utilize a taxonomy for filtering,
If provided

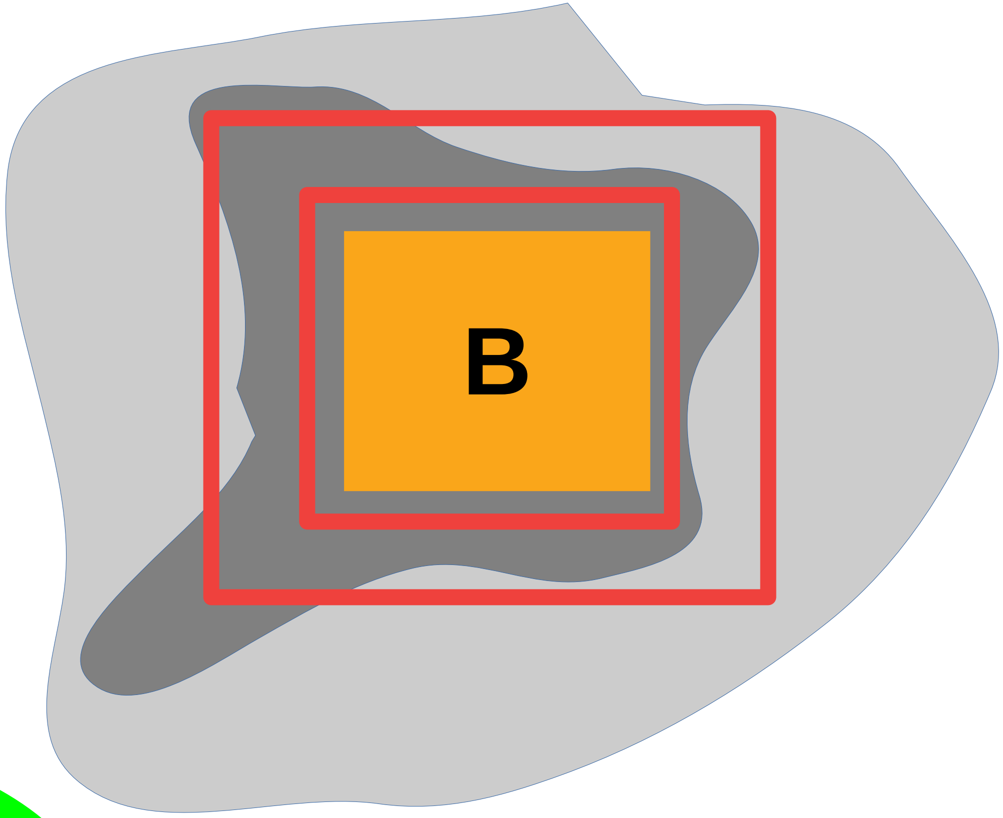
Step 2.1: Getting Candidate Preds. For Each Box

For ox, B:

1)Ge cates that hol B, **“H(B)”**

1. “fe check”: try on ample fr
2. Check with SAT-Solver

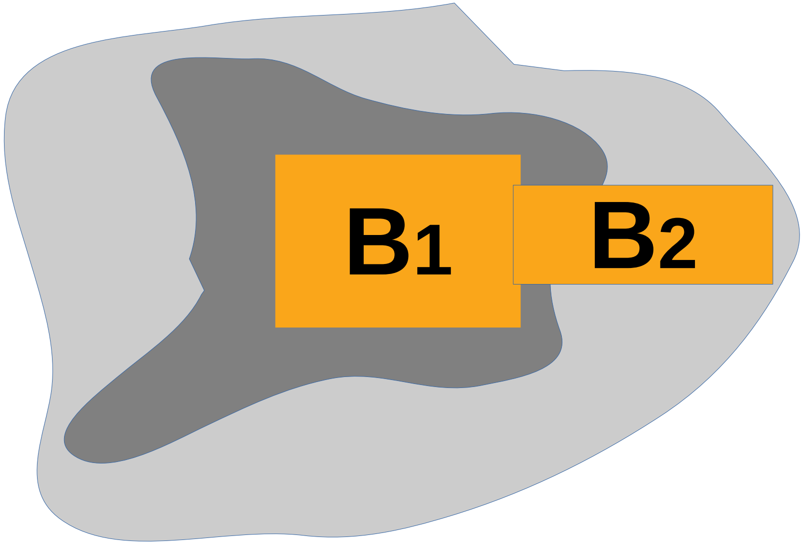
2) Get most specific predicates , **“S(B)”**

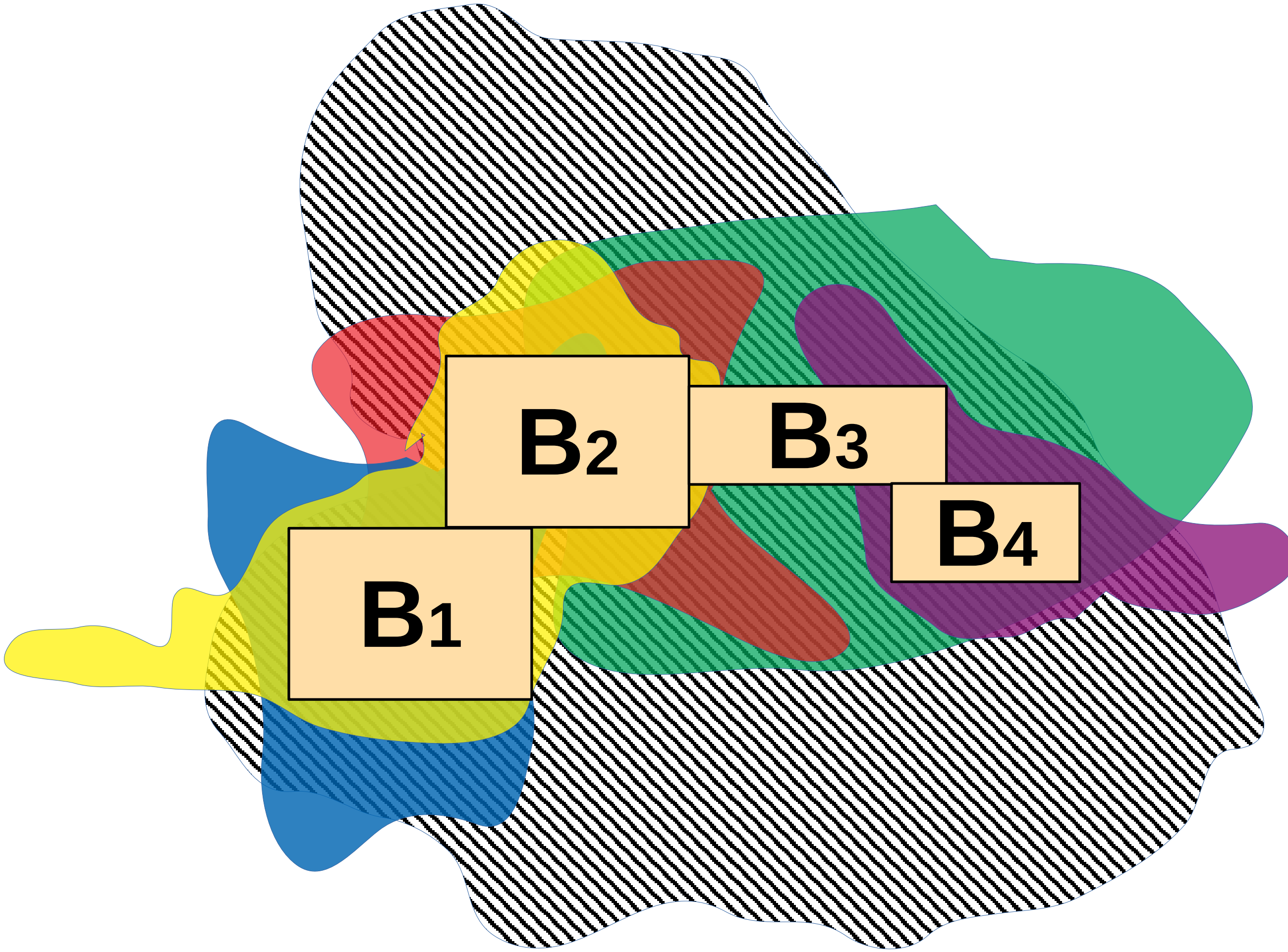


Note: Can utilize a taxonomy for filtering, if provided

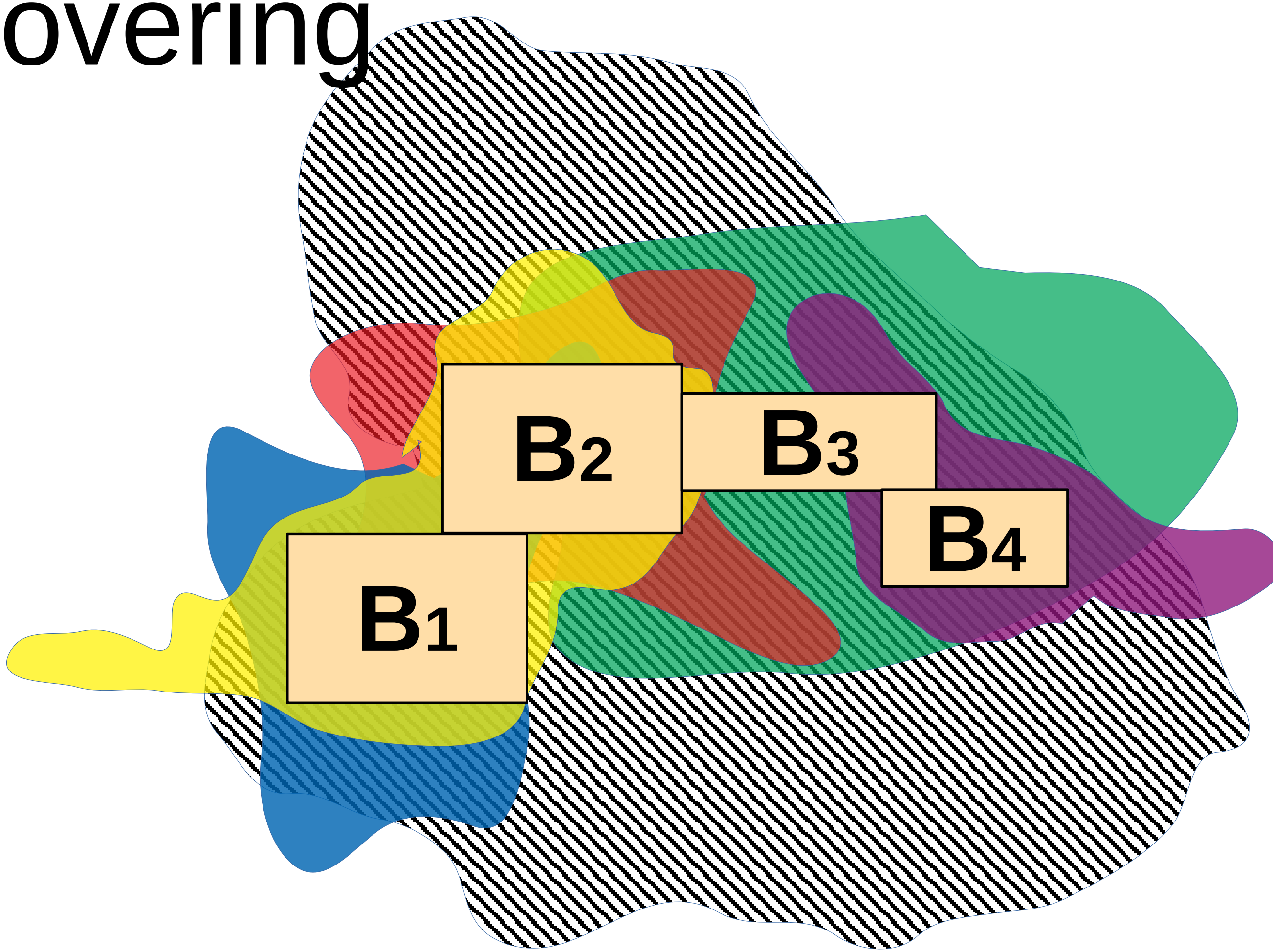
Step 2.2: Forming Global Covering

- Actually do ***two*** coverings: details in next slides
- Some subtleties for multi-dimensional setting
  - Ex: may need multiple preds to cover box; one pred variable x, another might cover y



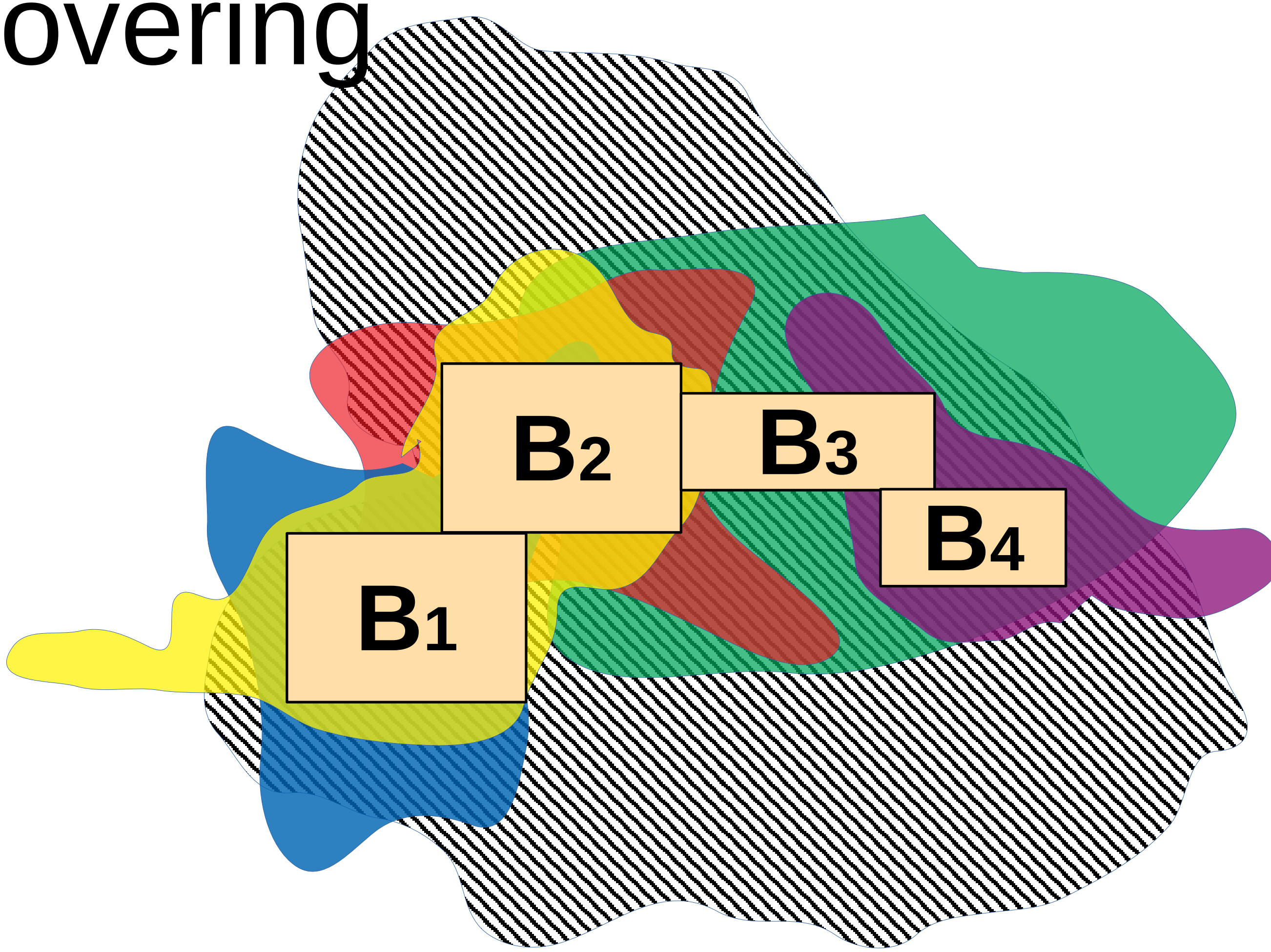

B_2
B_3
B_4
B_1

Start of *First* Covering



Start of *First* Covering

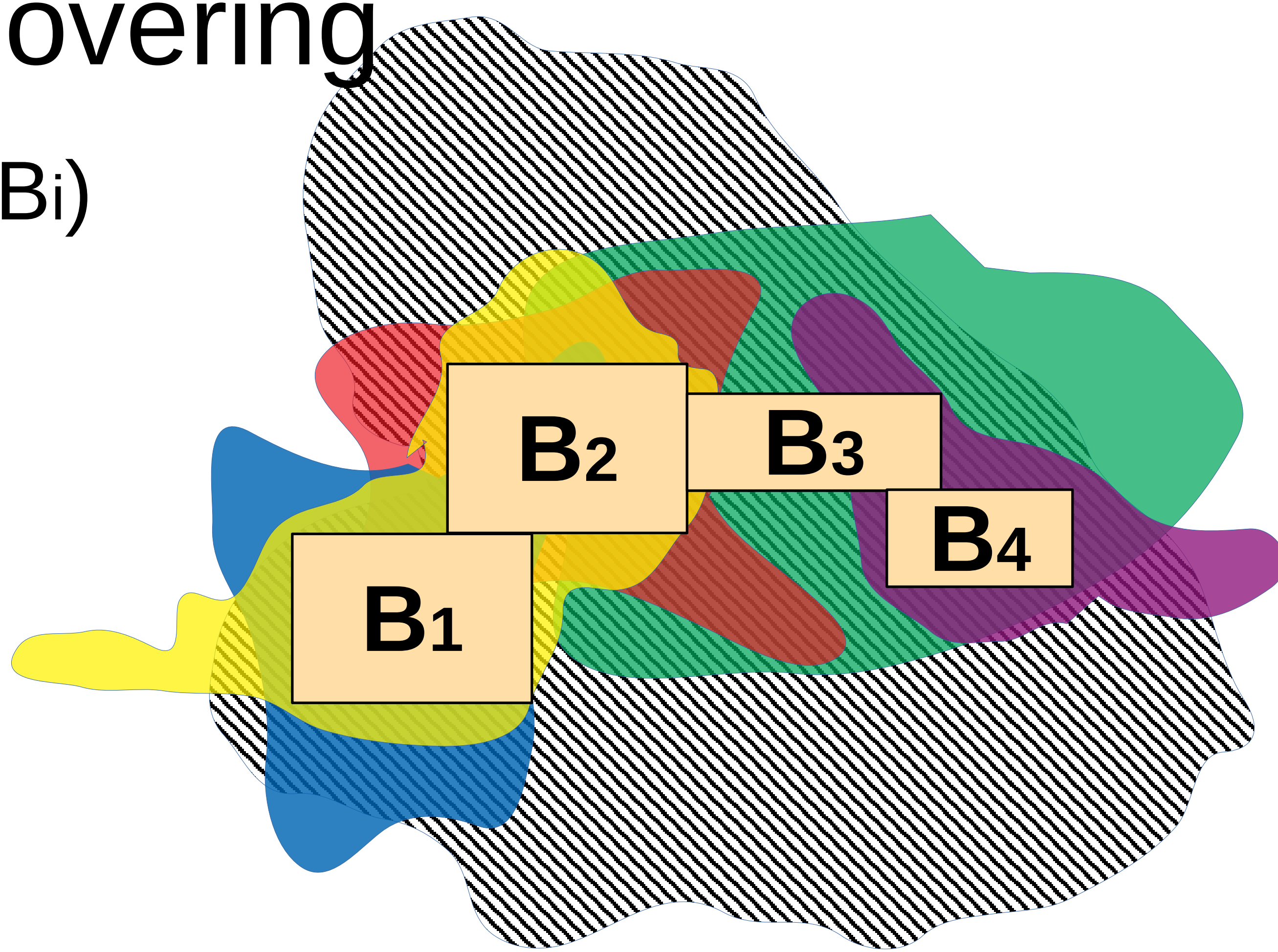
Boxes	Options
B_1	
B_2	
B_3	
B_4	



Start of *First* Covering

- Options(B_i) = S(B_i)

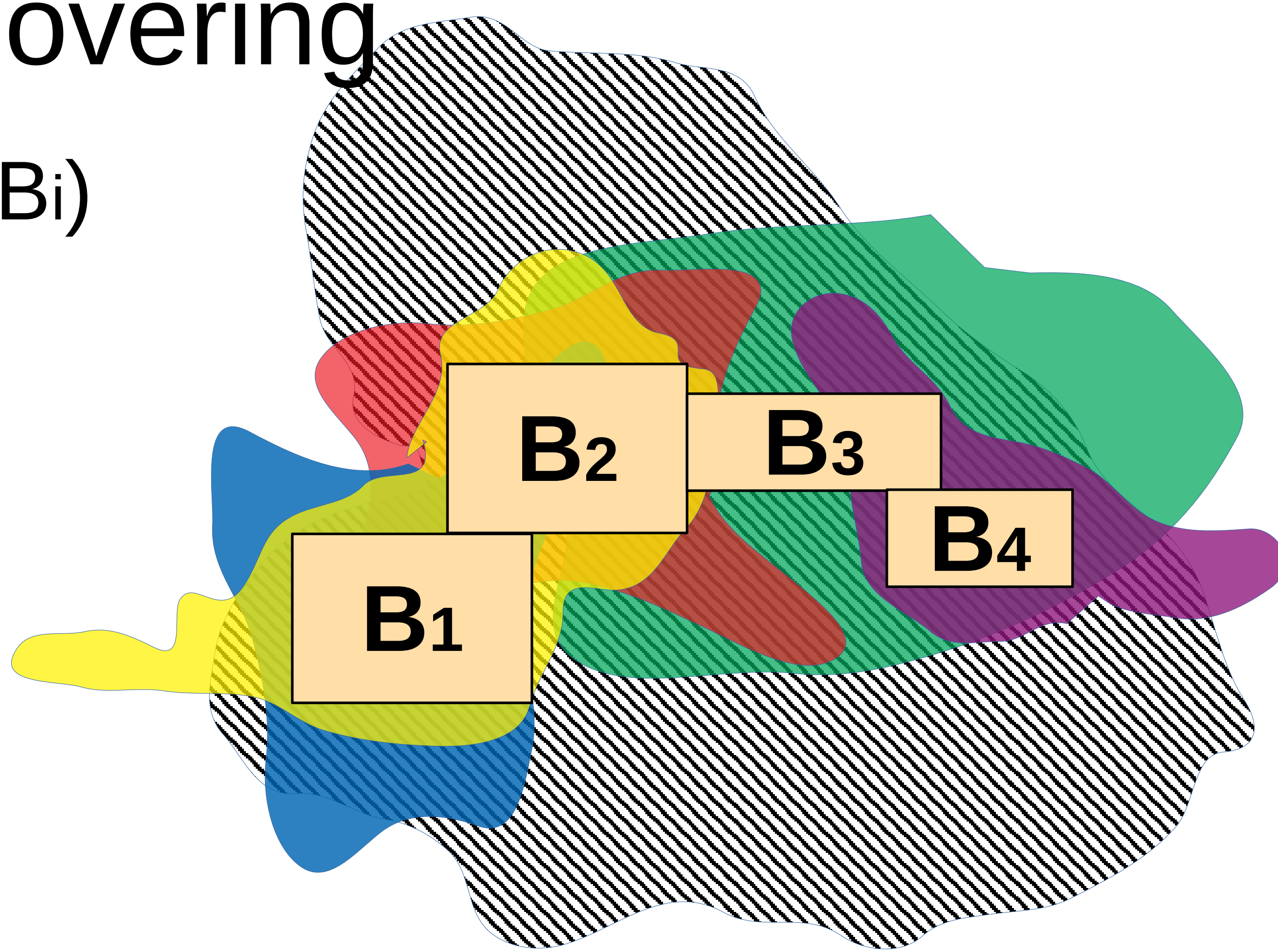
Boxes	Options
B_1	🟨 🟦
B_2	🟨 🟥
B_3	🟩
B_4	🟪



Start of *First* Covering

- Options(B_i) = S(B_i)

Boxes	Options
B_1	🟨 🟦
B_2	🟨 🟥
B_3	🟩
B_4	🟪

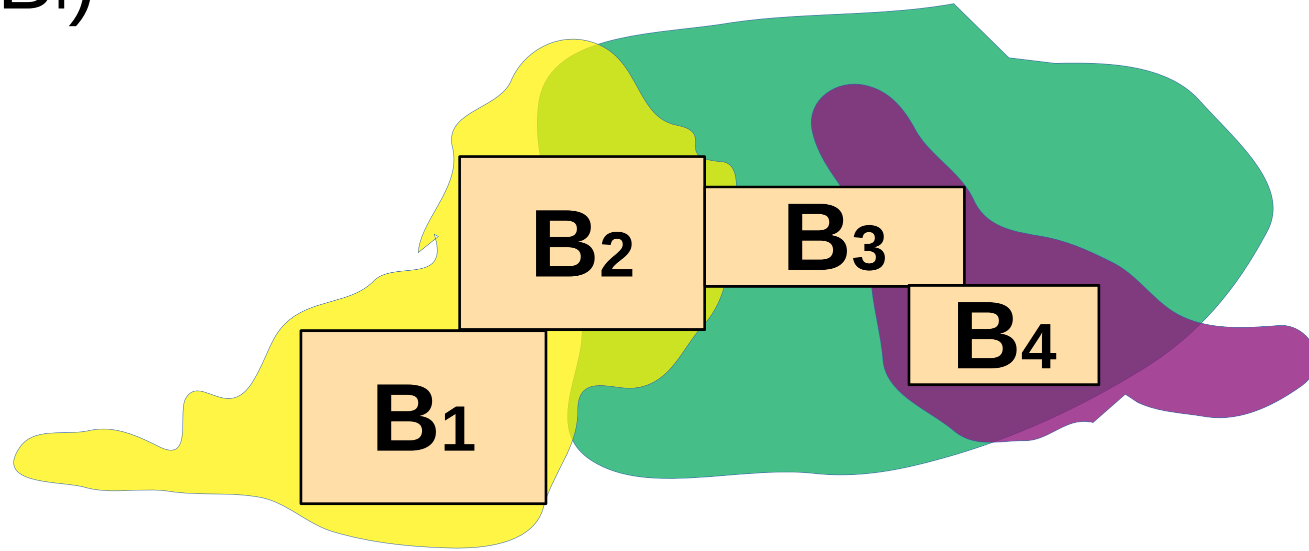


Green not listed because it was less specific,
Despite green covering B4.

Start of *First* Covering

- Options(B_i) = S(B_i)

Boxes	Options
B_1	🟨 🟦
B_2	🟨 🟥
B_3	🟩
B_4	🟪

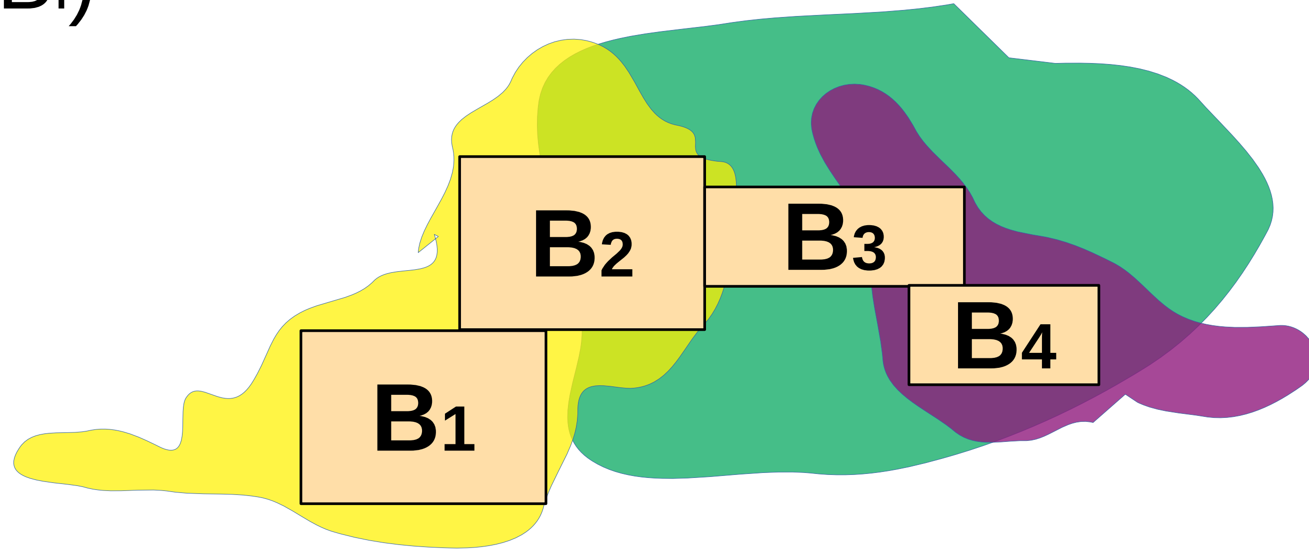


Green not listed because it was less specific,
Despite green covering B_4.

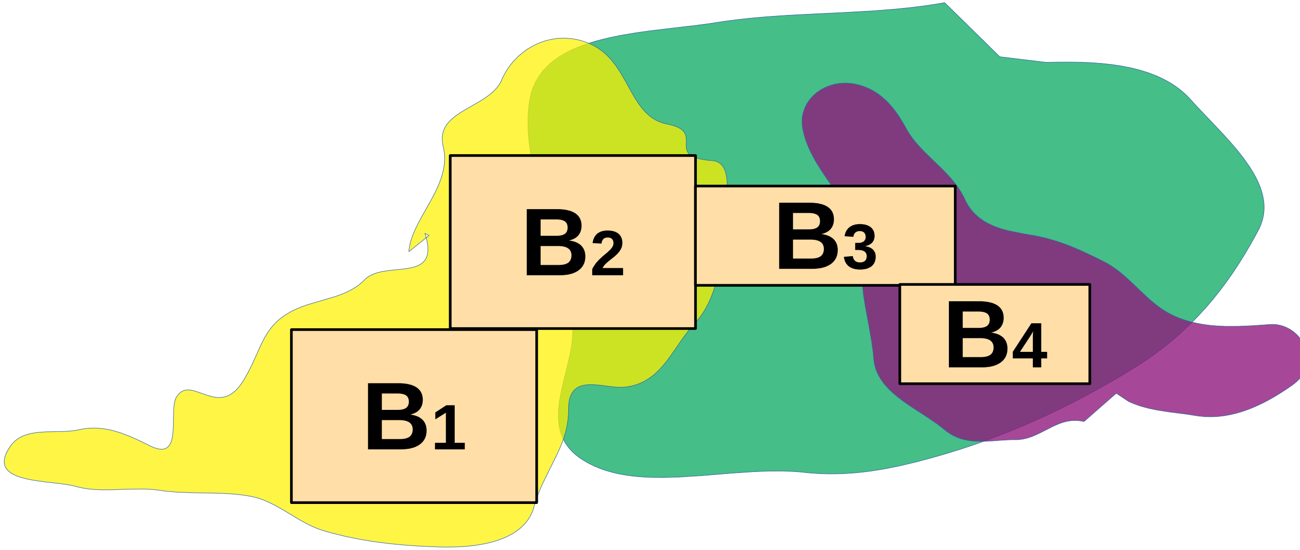
Start of *First* Covering

- Options(B_i) = S(B_i)

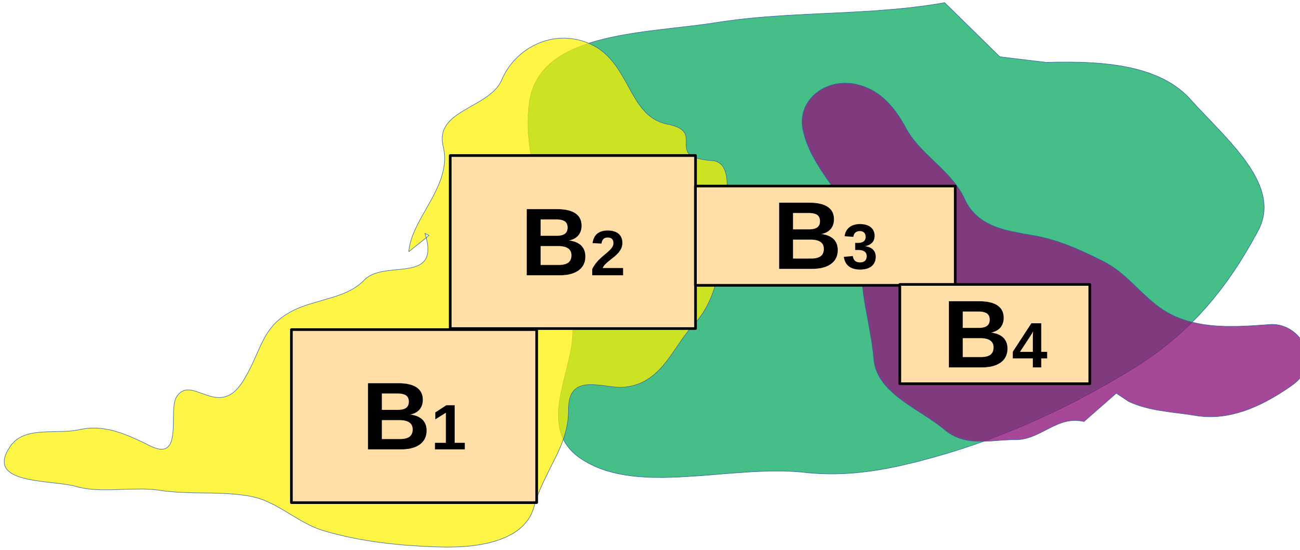
Boxes	Options
B_1	🟨 🟦
B_2	🟨 🟥
B_3	🟩
B_4	🟪



C_1 = { 🟨 , 🟩 , 🟪 }

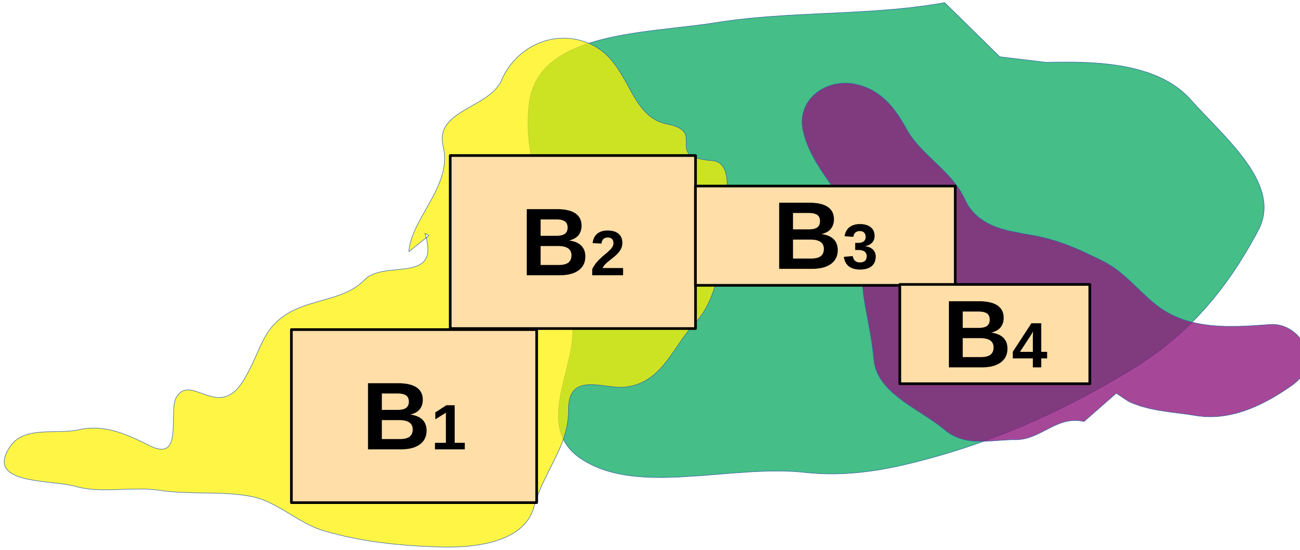

B_2
B_3
B_4
B_1

Start of *Second* Covering



Start of *Second* Covering

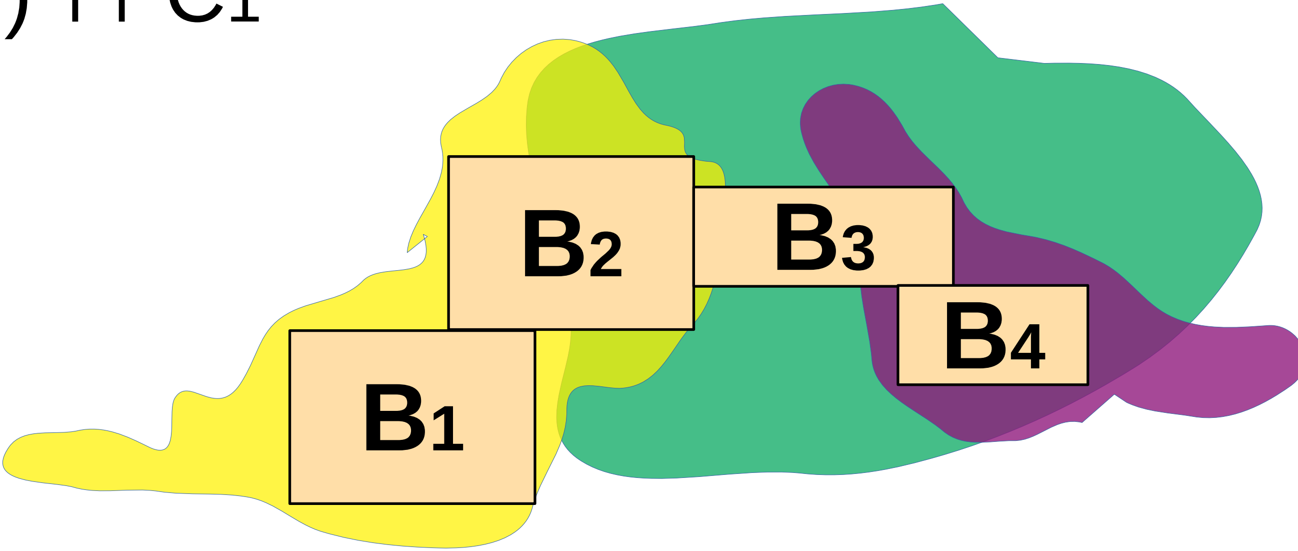
Boxes	Options
B_1	
B_2	
B_3	
B_4	



Start of *Second* Covering

- Options(B_i) = H(B_i) $\cap$ C_1

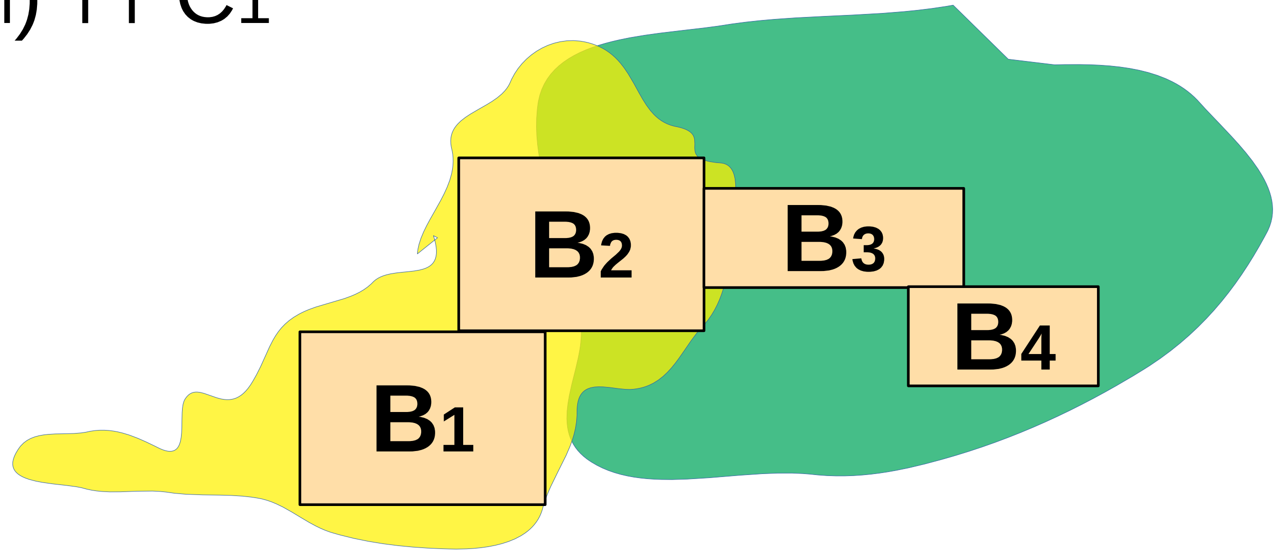
Boxes	Options
B_1	🟨
B_2	🟨
B_3	🟩
B_4	🟩 🟪



Start of *Second* Covering

- Options(B_i) = H(B_i) $\cap$ C_1

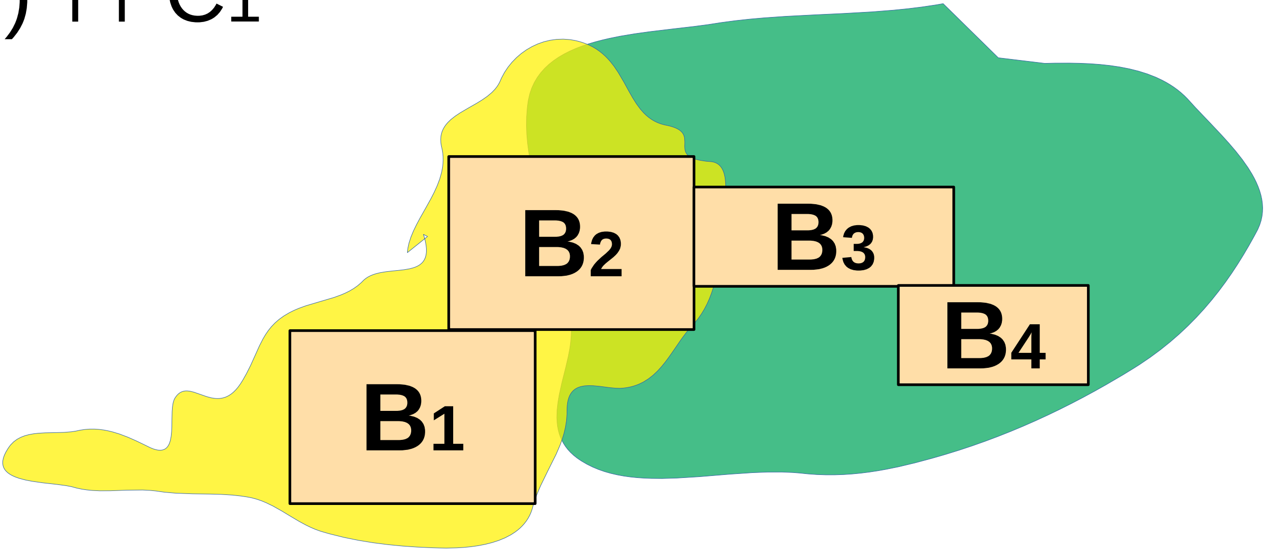
Boxes	Options
B_1	🟨
B_2	🟨
B_3	🟩
B_4	🟩 🟪



Start of *Second* Covering

- Options(B_i) = H(B_i) $\cap$ C_1

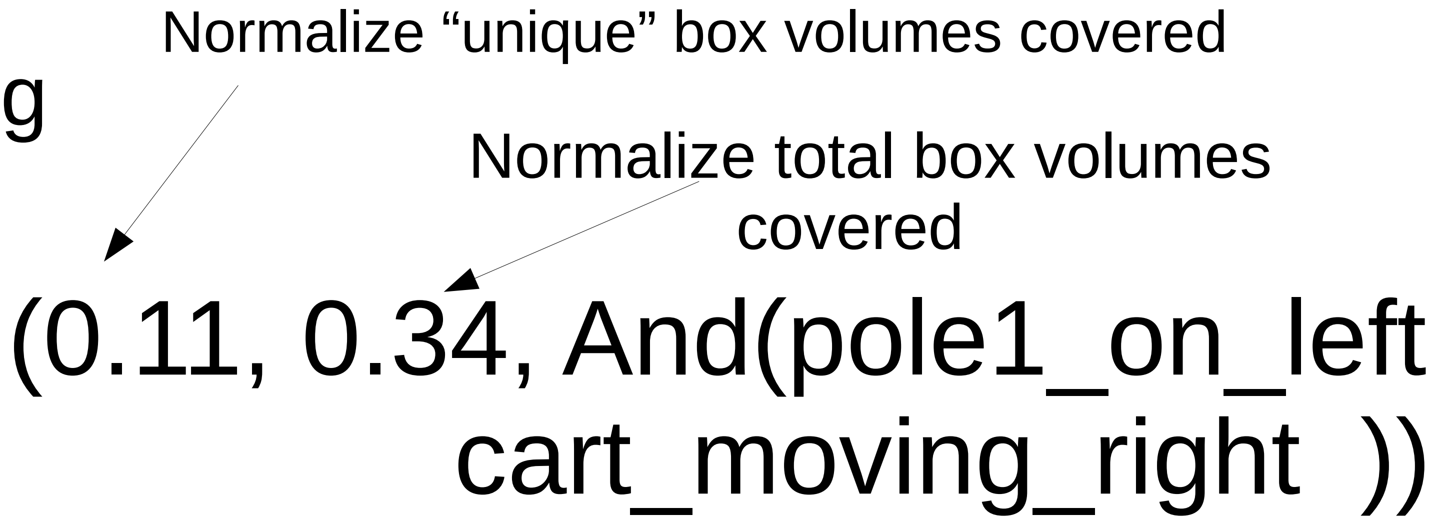
Boxes	Options
B_1	🟨
B_2	🟨
B_3	🟩
B_4	🟩 🟪



$C_2 = \{$ 🟨 , 🟩 $\}$

Cleaning and Presenting to User

- Some further post-processing
- Gather and show

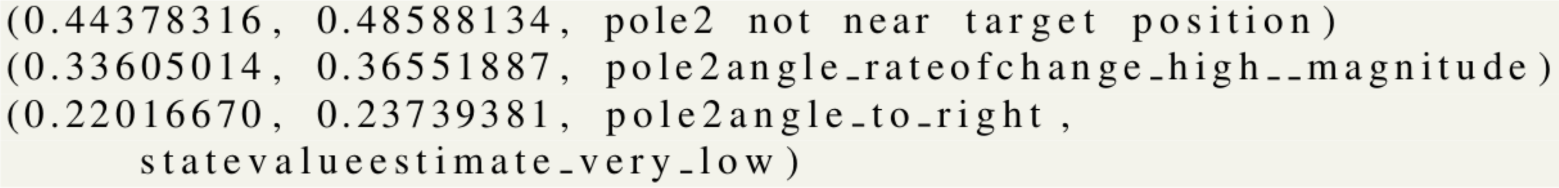


Normalize “unique” box volumes covered

Normalize total box volumes covered

(0.11, 0.34, And(pole1_on_left cart_moving_right))

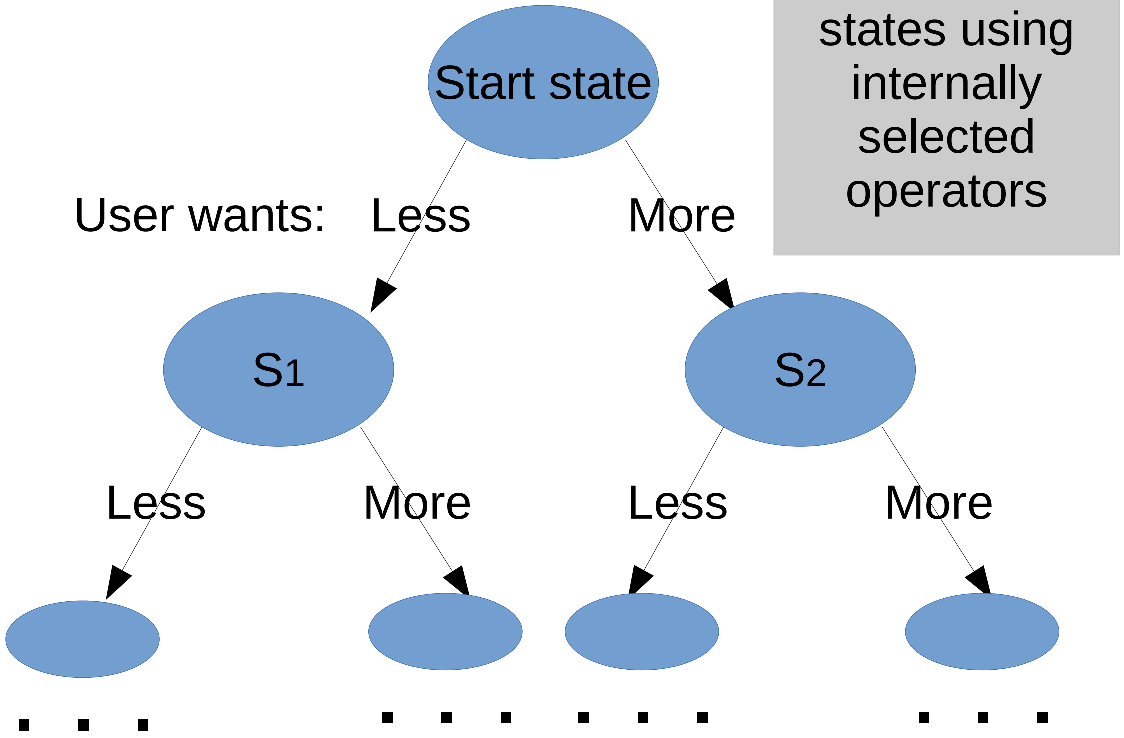
Output is a Weighted DNF



```
(0.44378316, 0.48588134, pole2 not near target position)
(0.33605014, 0.36551887, pole2angle_rateofchange_high__magnitude)
(0.22016670, 0.23739381, pole2angle_to_right ,
        statevalueestimate_very_low)
```

Using Feedback

- Fanoos has many internal parameters for:
  - CEGAR
  - Box-merging
  - Predicate
  - Etc.
- Use state-operator model
  - Feedback changes state and internal params
  - View as search for proper abstraction level



Get between states using internally selected operators

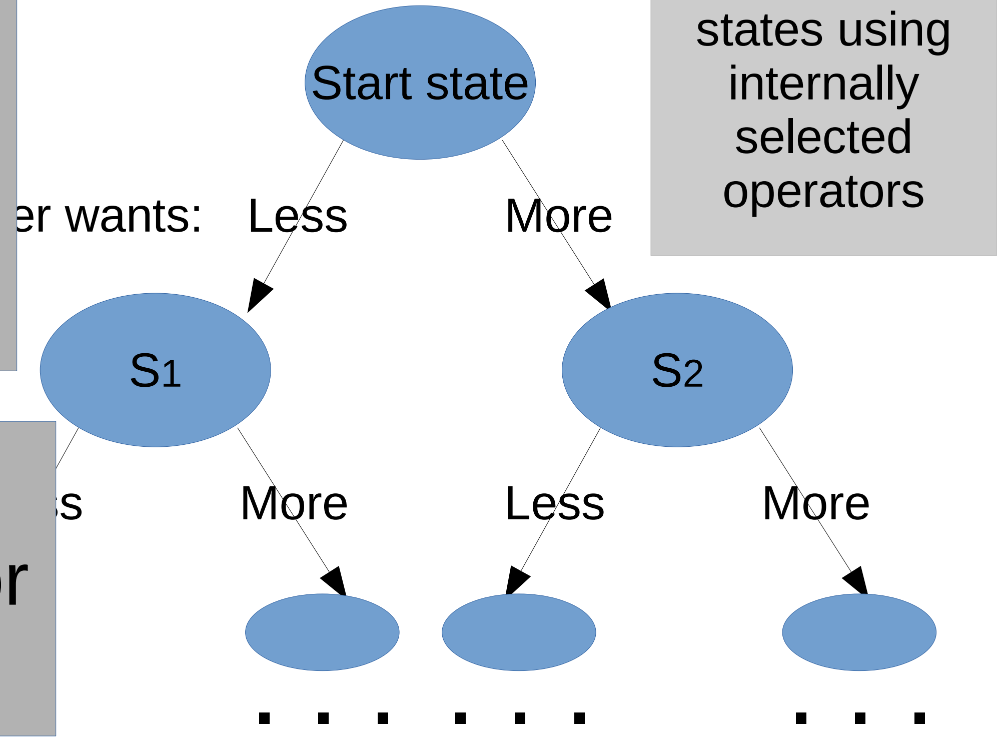
Using Feedback

- Fanoos l
  paramet
  - CEGAR
  - Box-me
  - Predica
  - Etc.

How select operators?
Hand-written **heuristics**.
Generally try to get smaller boxes, and looser descriptions for greater abstraction

- Use s
  - Fe
    and
  - Vie
    abstraction level

On-going future work: using ML-back operator selection


er wants:
Start state
Less
More
S1
S2
s
More
Less
More
. . . .

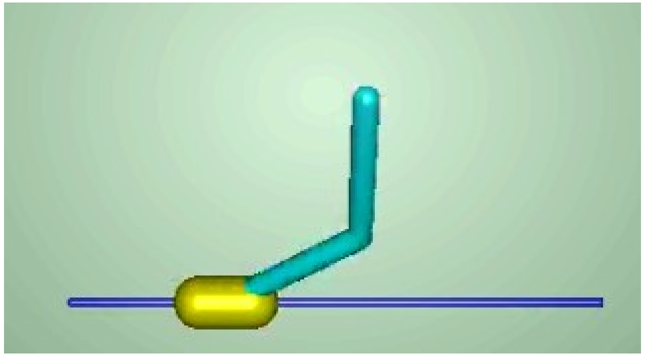
Get between states using internally selected operators

Experiments

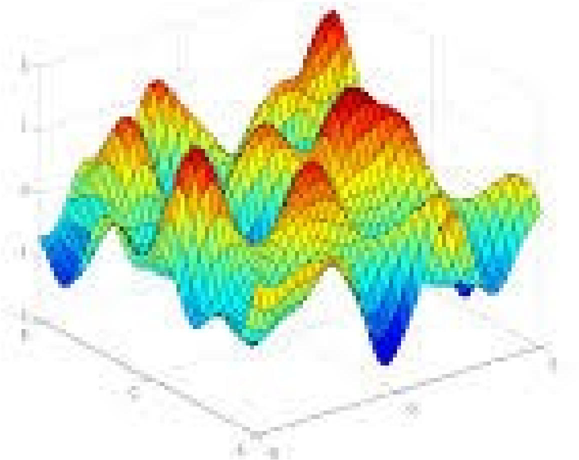
Fanoos

Experiments

- Ran on
  - Invertible double pendulum policy
    - 6D input, 2D output
  - A 3-degree polynomial regression for CPU Usage
    - 5D pre-featurization input, 3D output
- Preds. formed by mix of hand, data statistics, and templates



Openai gym
InvertedDoublePendulum-v2



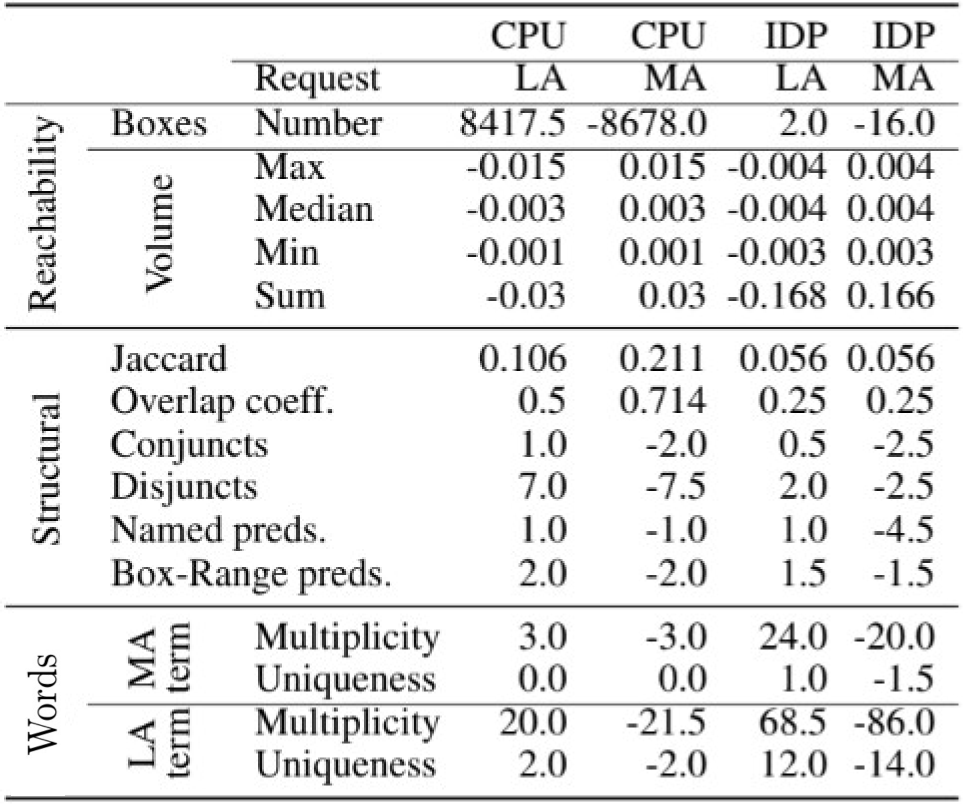
Experiments

- 130+ Start questions, several hundred replies total
  - Questions randomly generated based on some criteria
  - Half asked to make **more abstract (MA)**, half asked to make **less abstract (LA)**
- Compared befores-and-afters for:
  - Reachability
  - Result structure
  - Some approximation of agreement with human intuition

Experiments

- Reachability: MA tend to result in fewer, larger boxes. Opposite for LA
- Structural:
  - MA tend to be shorter, and have fewer conjuncts
  - Based on Jaccard and overlap score, not just becoming more verbose

Table 2: Median *relative* change in description before and after Fanoos adjusts the abstraction in the requested direction



			CPU	CPU	IDP	IDP
		Request	LA	MA	LA	MA
Reachability	Boxes	Number	8417.5	-8678.0	2.0	-16.0
	Volume	Max	-0.015	0.015	-0.004	0.004
		Median	-0.003	0.003	-0.004	0.004
		Min	-0.001	0.001	-0.003	0.003
		Sum	-0.03	0.03	-0.168	0.166
Structural	Jaccard		0.106	0.211	0.056	0.056
	Overlap coeff.		0.5	0.714	0.25	0.25
	Conjuncts		1.0	-2.0	0.5	-2.5
	Disjuncts		7.0	-7.5	2.0	-2.5
	Named preds.		1.0	-1.0	1.0	-4.5
	Box-Range preds.		2.0	-2.0	1.5	-1.5
Words	MA term	Multiplicity	3.0	-3.0	24.0	-20.0
		Uniqueness	0.0	0.0	1.0	-1.5
	LA term	Multiplicity	20.0	-21.5	68.5	-86.0
		Uniqueness	2.0	-2.0	12.0	-14.0

Experiments

- Approximate human judgment:
  - Labeled each predicate as higher or lower abstractness
  - “Grain of salt measure”: course labels and did not review whole output
  - As expected: LA requests tended for more lower abstraction terms, opposite for MA requests

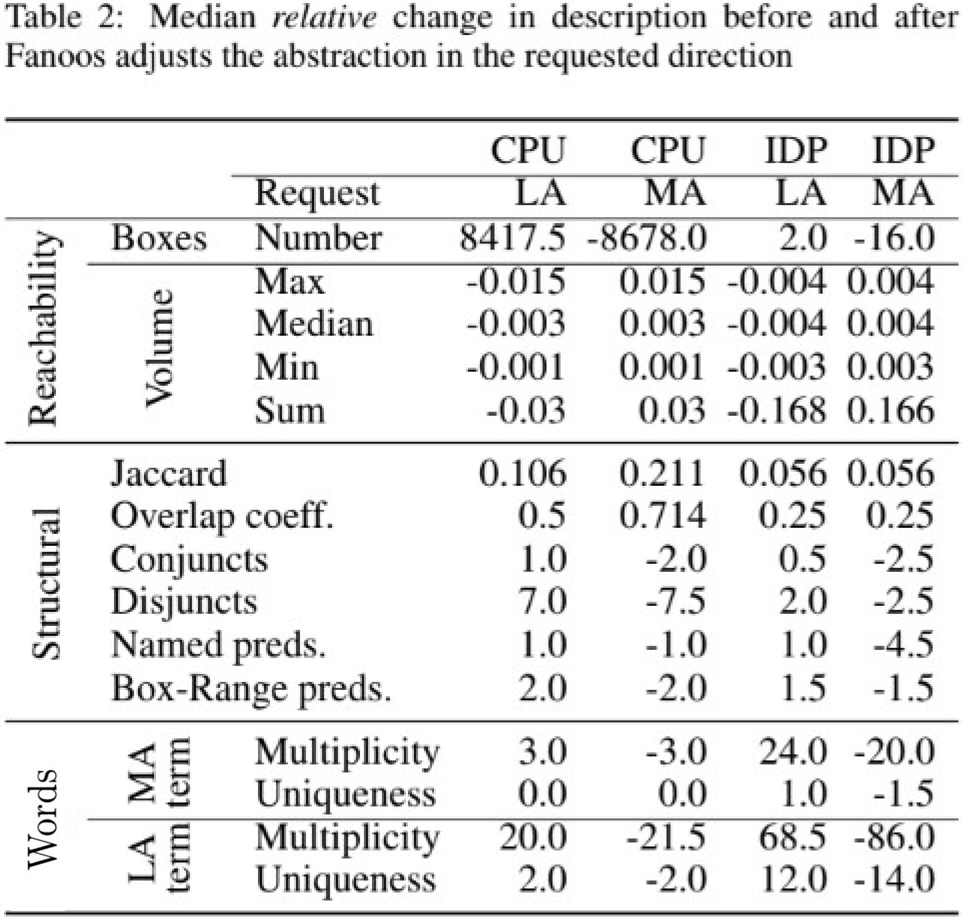


Table 2: Median *relative* change in description before and after Fanoos adjusts the abstraction in the requested direction

			CPU	CPU	IDP	IDP
		Request	LA	MA	LA	MA
Reachability	Boxes	Number	8417.5	-8678.0	2.0	-16.0
	Volume	Max	-0.015	0.015	-0.004	0.004
		Median	-0.003	0.003	-0.004	0.004
		Min	-0.001	0.001	-0.003	0.003
		Sum	-0.03	0.03	-0.168	0.166
Structural	Jaccard		0.106	0.211	0.056	0.056
	Overlap coeff.		0.5	0.714	0.25	0.25
	Conjuncts		1.0	-2.0	0.5	-2.5
	Disjuncts		7.0	-7.5	2.0	-2.5
	Named preds.		1.0	-1.0	1.0	-4.5
	Box-Range preds.		2.0	-2.0	1.5	-1.5
Words	MA term	Multiplicity	3.0	-3.0	24.0	-20.0
		Uniqueness	0.0	0.0	1.0	-1.5
	LA term	Multiplicity	20.0	-21.5	68.5	-86.0
		Uniqueness	2.0	-2.0	12.0	-14.0

Conclusion & Closing Thoughts

Fanoos

Fanoos:
Shining a Light on Black-Box AI

- Discussed the Fanoos system for XAI focused on ML
- Explanations from Fanoos are
  - Formally sound or probabilistic, depending on user preference
  - Interactive
  - Curtailable to user’s desired abstraction level
- Provided empirically demonstration
  - Fanoos recovered abstraction levels from semantics of the domain

Three Things I Want ***You*** to Have a Good Sense of After This:

Three Things I Want ***You*** to Have a Good Sense of After This:

1) the specific implementation in Fanoos

Three Things I Want ***You*** to Have a Good Sense of After This:

1) the specific implementation in Fanoos

2) the high-level ideas and motivations

Three Things I Want ***You*** to Have a Good Sense of After This:

1) the specific implementation in Fanoos

2) the high-level ideas and motivations

3) that the verification community can contribute a lot in XAI.

Three Things I Want ***<u>You</u>*** to Have a Good Sense of After This:

1) the specific implementation in Fanoos
2) the high-level ideas and motivations
3) that the verification community can contribute a lot in XAI.

Closing Thoughts

- Under-explored: ***Formal Verification + XAI***
  - A lot of complementary abilities and focuses
  - Current pushes to be aware of: see “Explainable AI: Beware of Inmates Running the Asylum” ([11])
- Need for flexibility and ***varying abstraction***
  - Examples of this working well for other tools and across CS

REFERENCES

Contact email: dcbayani@alumni.cmu.edu

[1] Amina Adadi and Mohammed Berrada. 2018. Peeking inside the black-box: A survey on Explainable Artificial Intelligence (XAI). *IEEE Access* 6 (2018), 52138–52160.

[2] Edmund Clarke, Ansgar Fehnker, Zhi Han, Bruce Krogh, Olaf Stursberg, and Michael Theobald. 2003. Verification of hybrid systems based on counterexample-guided abstraction refinement. In *International Conference on Tools and Algorithms for the Construction and Analysis of Systems*. Springer, 192–207.

[3] Edmund Clarke, Orna Grumberg, Somesh Jha, Yuan Lu, and Helmut Veith. 2000. Counterexample-guided abstraction refinement. In *International Conference on Computer Aided Verification*. Springer, 154–169.

[4] Patrick Cousot and Radhia Cousot. 1977. Abstract Interpretation: A Unified Lattice Model for Static Analysis of Programs by Construction or Approximation of Fixpoints. In *Conference Record of the Fourth ACM Symposium on Principles of Programming Languages, Los Angeles, California, USA, January 1977*, Robert M. Graham, Michael A. Harrison, and Ravi Sethi (Eds.). ACM, 238–252. https://doi.org/10.1145/512950.512973

[5] Bradley Hayes and Brian Scassellati. 2016. Autonomously constructing hierarchical task networks for planning and human-robot collaboration. In *2016 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 5469–5476.

[6] Bradley Hayes and Julie A Shah. 2017. Improving robot controller transparency through autonomous policy explanation. In *2017 12th ACM/IEEE International Conference on Human-Robot Interaction (HRI)*. IEEE, 303–312.

[7] Sandy H. Huang, David Held, Pieter Abbeel, and Anca D. Dragan. 2019. Enabling robots to communicate their objectives. *Autonomous Robots* 43, 2 (01 Feb 2019), 309–326. https://doi.org/10.1007/s10514-018-9771-0

[8] Jinkyu Kim, Anna Rohrbach, Trevor Darrell, John F. Canny, and Zeynep Akata. 2018. Textual Explanations for Self-Driving Vehicles. (2018), 577–593 pages. https://doi.org/10.1007/978-3-030-01216-8_35

[9] Anurag Koul, Alan Fern, and Sam Greydanus. 2019. Learning Finite State Representations of Recurrent Policy Networks. In *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019*. https://openreview.net/forum?id=S1gOpsCctm

[10] Zachary C Lipton. 2016. The mythos of model interpretability. *arXiv preprint arXiv:1606.03490* (2016).

[11] Tim Miller, Piers Howe, and Liz Sonenberg. 2017. Explainable AI: Beware of inmates running the asylum or: How I learnt to stop worrying and love the social and behavioural sciences. *arXiv preprint arXiv:1712.00547* (2017).

[12] Marco Túlio Ribeiro, Sameer Singh, and Carlos Guestrin. 2016. “Why Should I Trust You?”: Explaining the Predictions of Any Classifier. (2016), 1135–1144 pages. https://doi.org/10.1145/2939672.2939778