

# Intent Formalization

---

# PBE is not always enough

---

## Pros

- Ease of use
- Broader base of users
- Error detection

## Cons

- Ambiguity in intent
- Lack of Formal Guarantee

One needs to formally specify the desired behavior of the target program!

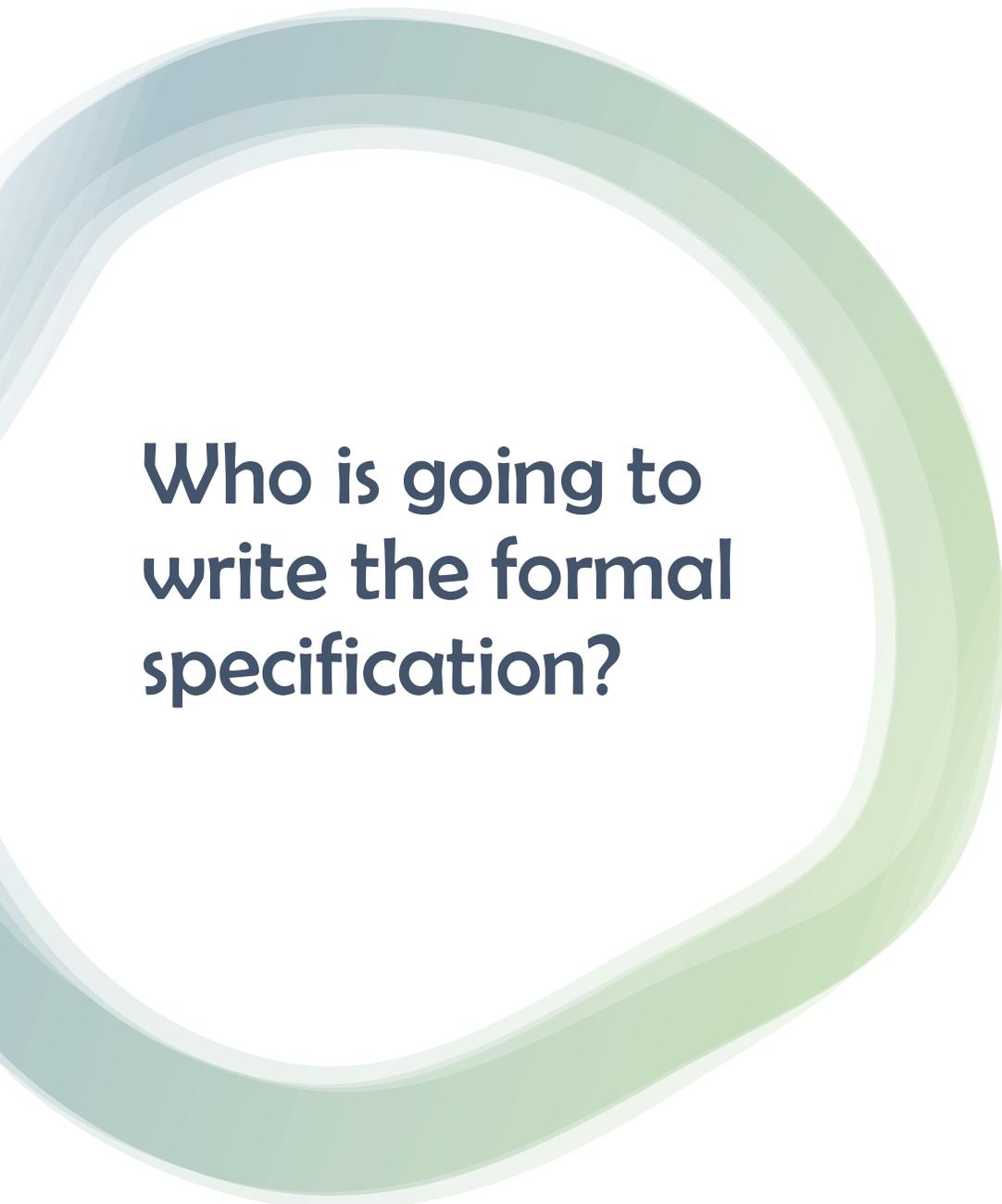
# Example

```
int f(int x, int y)
{
    z = 0;
    i = x;
    while (i) {
        z = z + y;
        i = i - 1;
    }
    return z;
}
```

PBE:

<i>x</i>	<i>y</i>	<i>out</i>
1	1	1
2	3	6
2	-1	-2
4	0	0
3	4	12

Formal spec:  $out = x \cdot y$



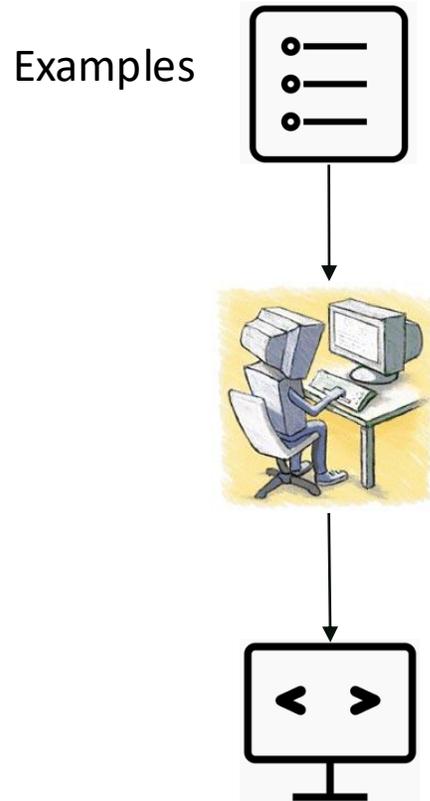
## Who is going to write the formal specification?

- Most users are not trained to write specifications using logics or other rigorous mathematical languages
- The synthesizer is sneaky and always finds a workaround (e.g., an infinite loop) for an incomplete specification

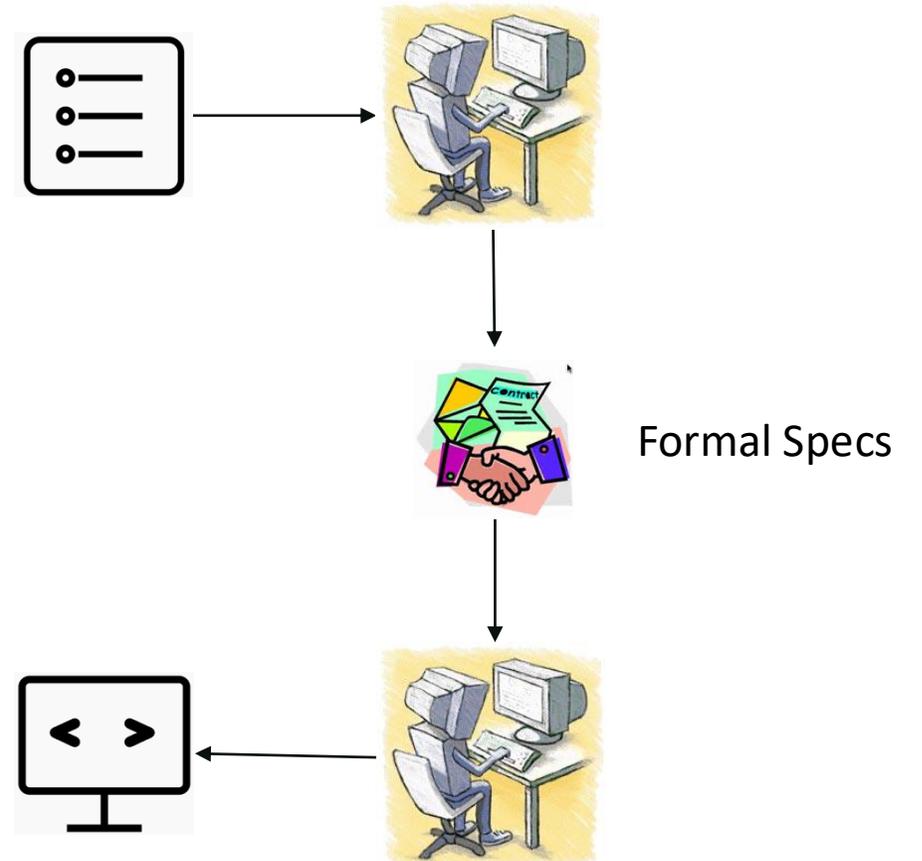
# Examples to the rescue!

---

## Traditional PBE



## Synthesize formal specs from examples



# **From Examples to RegEx**

---

# Regular Expressions

---

Regular expressions are a *syntactic tool* for defining *regular languages*

- Common feature in many languages; but the basics of regular expressions are much simpler than what you see in languages like Perl or Python
- String literals combined by **choice** and **star**
- **“Regular” languages:** Regular expressions can be represented as deterministic finite automata (DFA), and vice versa

[alice@example.com](mailto:alice@example.com)

[bob.smith@company.org](mailto:bob.smith@company.org)

[charlie123@mail.co.uk](mailto:charlie123@mail.co.uk)

[david@my-email.com](mailto:david@my-email.com)



`^[a-zA-Z0-9._%+-]+@[a-zA-Z0-9.-]+\.[a-zA-Z]{2,}$`

(123) 456-7890

987-654-3210

(555) 123-4567

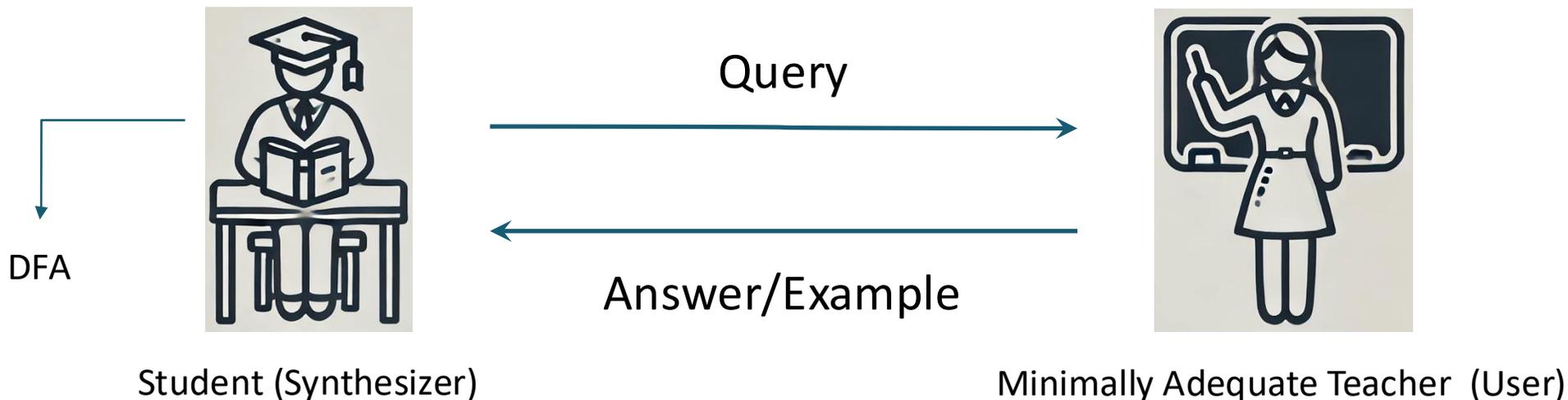
111-222-3333



`^((\d{3})\s?|\d{3}[-.\s])\d{3}[-.\s]?\d{4}$`

# Angluin's $L^*$ Algorithm (1987)

---



## Membership Query

- Is the string  $s$  accepted by the target language?
- Answer: yes/no

## Equivalence Query

- Does the candidate DFA match the target language?
- Answer: yes/counterexample
- Done if the answer is yes!

# Idea: maintain a candidate DFA

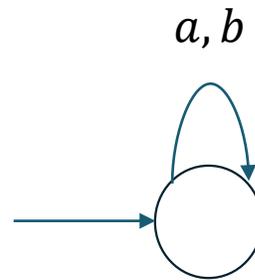
---

0. Represent the DFA as an Observation Table (ignored)
1. Start with a simple hypothesis DFA (usually an empty state machine).
2. Use equivalence queries to check if the current hypothesis matches the target DFA.
3. If not match, use membership queries to refine the DFA with respect to the counterexample.
4. Repeat from step 2 until it correctly recognizes the language.

# Example

---

Alphabet:  $\{a, b\}$



Equivalence?



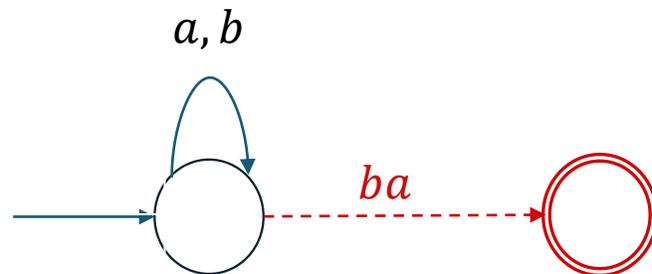
Cex:  $ba$



# Example

Alphabet:  $\{a, b\}$

$S$	$\epsilon$	Status
$\epsilon$	0	Current start state (Rejecting)
$b$	?	
$ba$	1	$b \in L$ (Accepting)
$S \cdot \Sigma$		
$a$	?	Matches $\epsilon$
$bb$	?	
$baa$	?	
$bab$	?	



But how?

Membership( $a$ )?  
Membership( $b$ )?



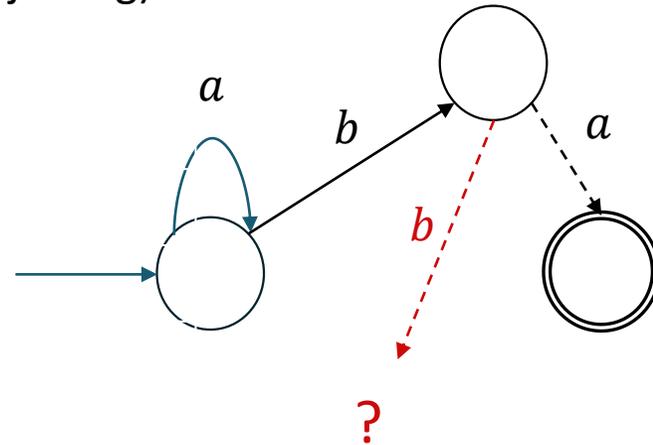
No, No



# Example

Alphabet:  $\{a, b\}$

$S$	$\epsilon$	Status
$\epsilon$	0	Current start state (Rejecting)
$b$	0	$b \notin L$ (Rejecting)
$ba$	1	$b \in L$ (Accepting)
$S \cdot \Sigma$		
$a$	0	Matches $\epsilon$
$bb$	?	
$baa$	?	
$bab$	?	



Membership( $bb$ )?



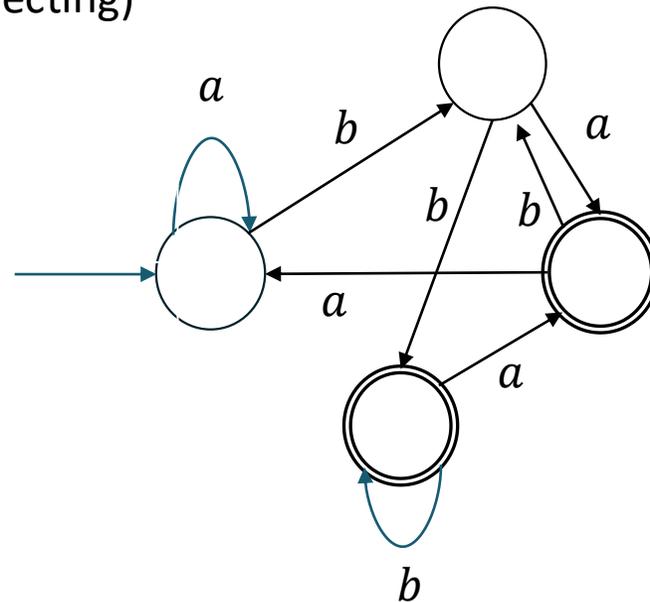
Yes



# Example

After several rounds of queries...

$S$	$\epsilon$	Status
$\epsilon$	0	Current start state (Rejecting)
$b$	0	$b \notin L$ (Rejecting)
$ba$	1	$b \in L$ (Accepting)
$S \cdot \Sigma$		
$a$	0	$b \notin L$ (Rejecting)
$bb$	1	$b \in L$ (Accepting)
$baa$	0	$b \notin L$ (Rejecting)
$bab$	0	$b \notin L$ (Rejecting)



Equivalence?  
→  
←  
Yes



# Model Learning

---

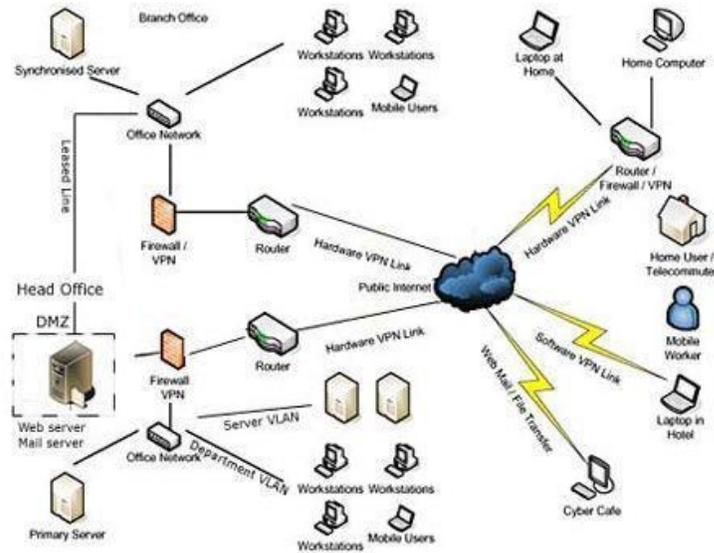
*“Even though faster algorithms have been proposed since then, the most efficient learning algorithms that are being used today all follow Angluin’s approach of a minimally adequate teacher (MAT).” – Frits Vaandrager*

# Comparative Synthesis

---

# Motivation: Network Design

---



- Plenty of source-destination pairs
- Multiple paths per s-d pair
- Multiple traffic classes
- Capacity/security restriction
- Given demand, figure out allocations

**It is not too hard to find a mediocre network design.**

# But what is the optimal solution?

How to trade-off multiple objectives (throughput and latency)?

## Software-driven WAN [1]

- Maximize throughput
- Minimize latency

Maximize

$$O(\text{throughput, latency}) \\ = 2 * \text{throughput} - \max(750 - \text{throughput}, 0) \\ - 5 * \text{latency} - 9 * \max(\text{latency} - 75, 0)$$

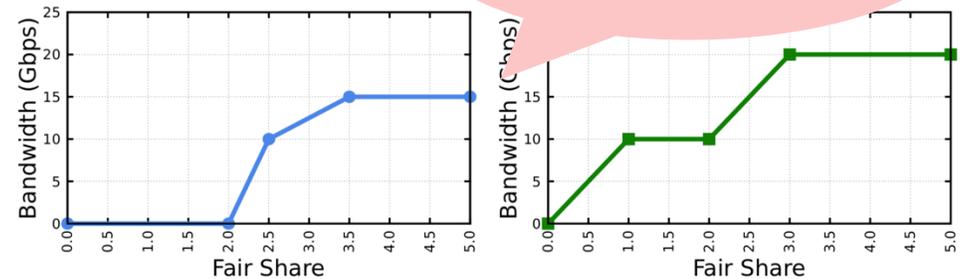
How to decide the knobs?

How to enforce fairness requirements for flows across different traffic classes?

## Bandwidth Enforcer (BwE) [2]

- Different classes have different bandwidth requirements and different priorities

How to decide fair shares?



Fair share as concave functions

[1] Achieving High Utilization with Software-Driven WAN. [Hong et al., SIGCOMM'13]

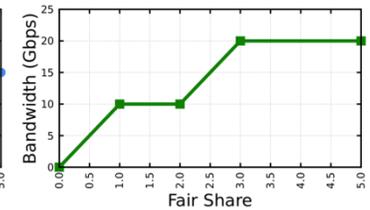
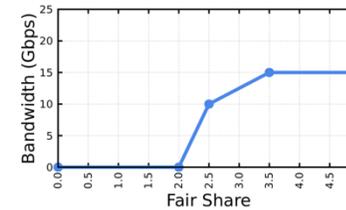
[2] BwE: Flexible, Hierarchical Bandwidth Allocation for WAN Distributed Computing. [Kumar et al., SIGCOMM'15]

# The synthesis conundrum remains

“Maximize throughput”  
“Minimize latency”

Give me an optimal program!

Give me the optimization target!



Start ::= Start + Start/0.5  
| x | 0 | 1 | Expr

Expr ::= Expr + Expr/0.4  
| x | 0 | 1

$O(\text{throughput}, \text{latency})$   
 $= 2 * \text{throughput} - \max(750 - \text{throughput}, 0)$   
 $- 5 * \text{latency} - 9 * \max(\text{latency} - 75, 0)$

# But didn't we have PBE?

Input

Throughput: 10Gbps  
Latency: 300ms  
...

Throughput: 7Gbps  
Latency: 100ms  
...



Let's compare!

Throughput: 10Gbps  
Latency: 300ms  
...

vs.

Throughput: 7Gbps  
Latency: 100ms  
...

Output

??

??

Throughput: 10Gbps  
Latency: 300ms  
...



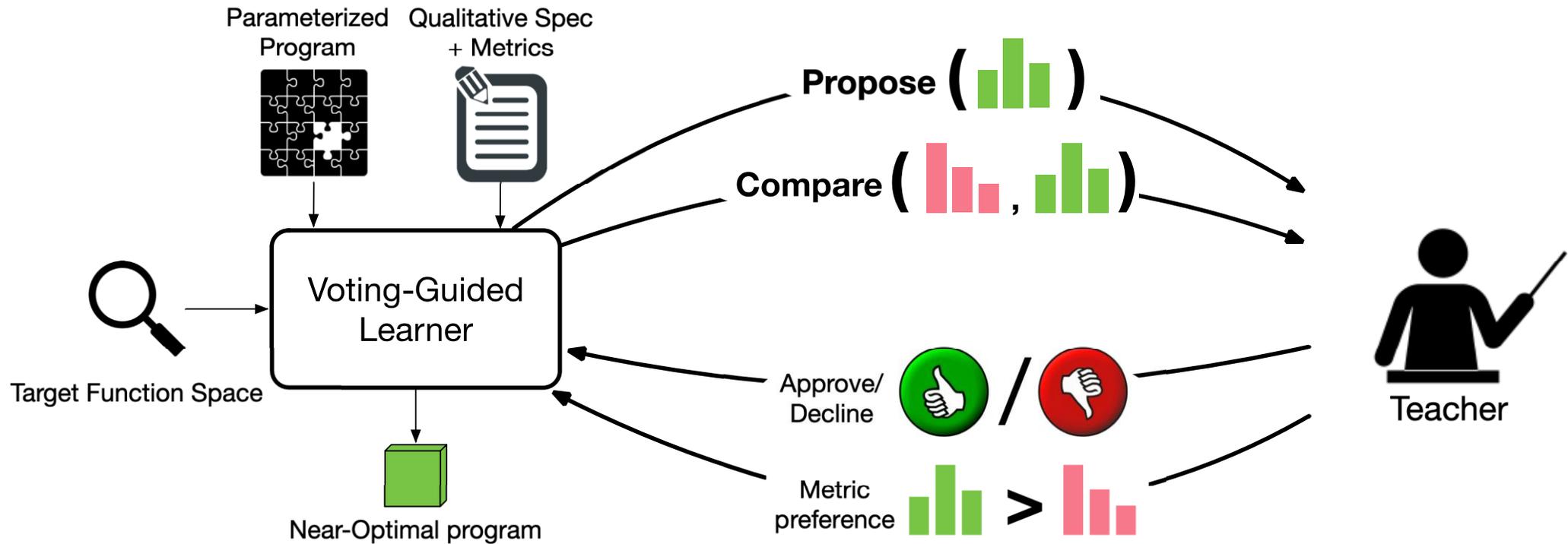
*"Comparison is the thief of joy."*

--Theodore Roosevelt

# Net10Q: Comparative Synthesis for Network Design

Learner's Goal:

Spend a **budgeted number of queries** and to produce a **near-optimal** program *from the perspective of the teacher*



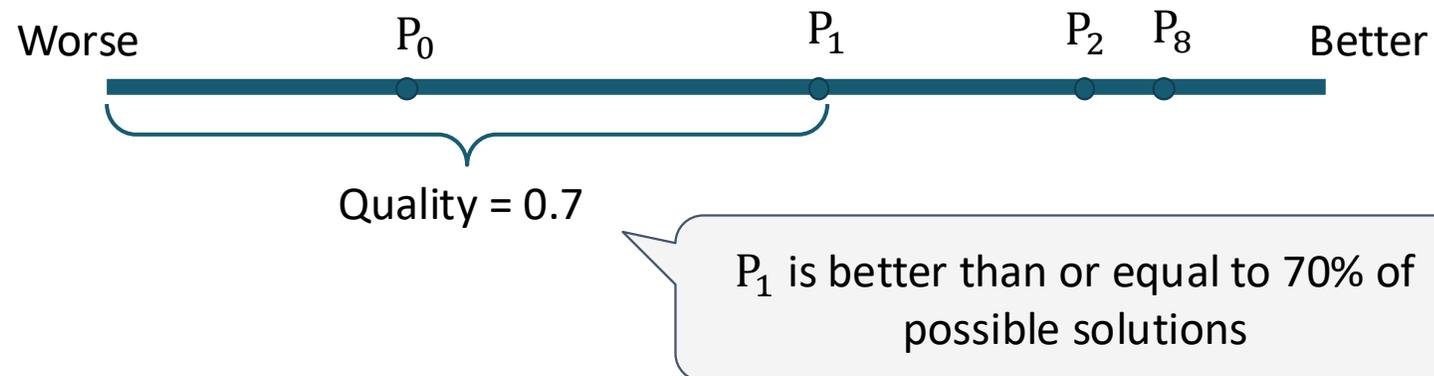
# Quality of Solution

---

How close a solution is to the ground truth optimal?

## Quality of Solution:

The “**relative rank**” of the solution among all solutions

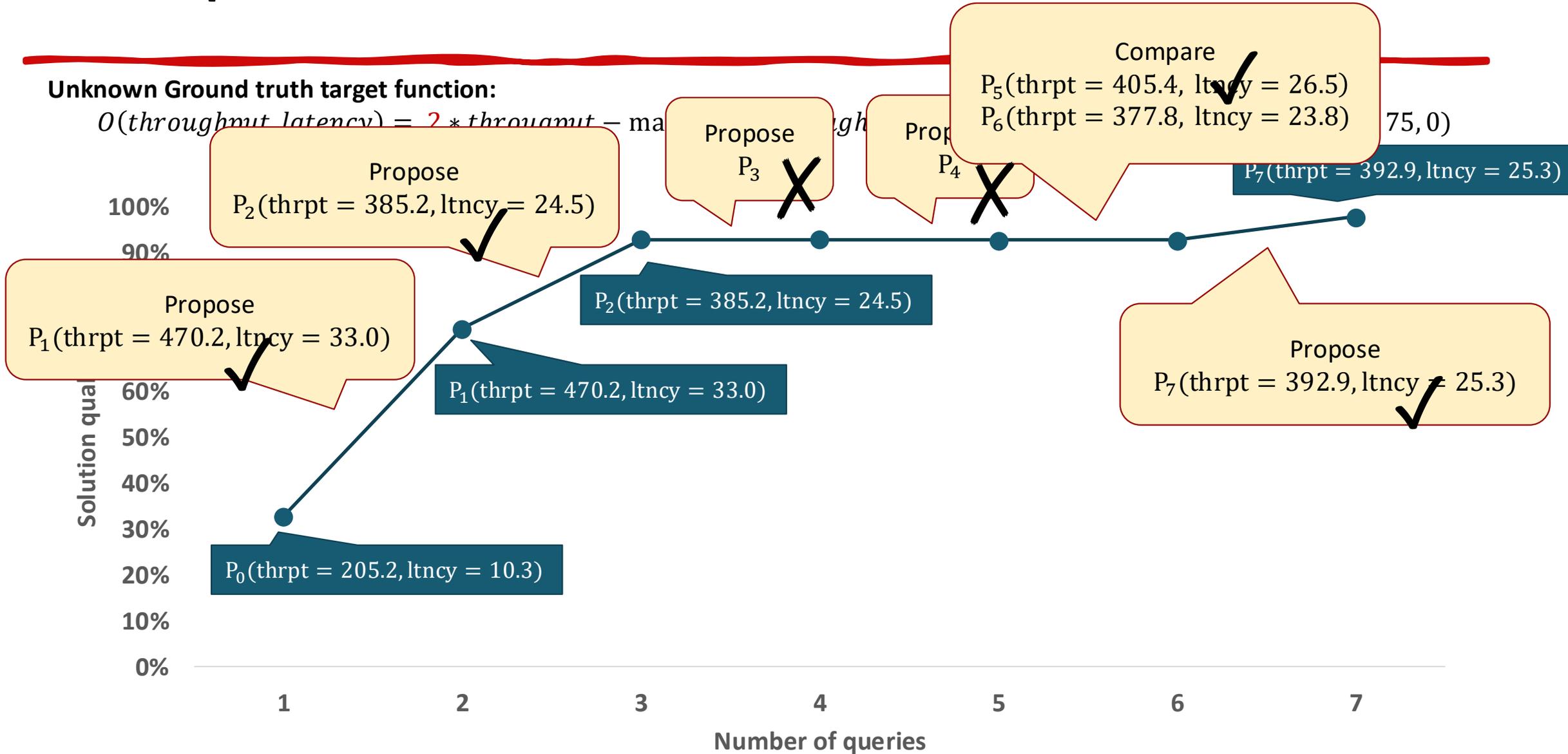


Computing the exact quality can be expensive, we estimate the quality by **sampling**

# Example

Unknown Ground truth target function:

$$O(\text{throughput latency}) = 2 * \text{throughput} - \text{max latency}$$



# How does the synthesizer work?

---

## A Voting-Guided Learning Algorithm

- Maintains a *Pareto candidate set*
- Each query can prune the search space, one way or the other
- Greedily prune the candidate set by making the *most informative* query (i.e., maximizing the worst-case space cut)
- Re-generate more candidates when the candidate set becomes too small

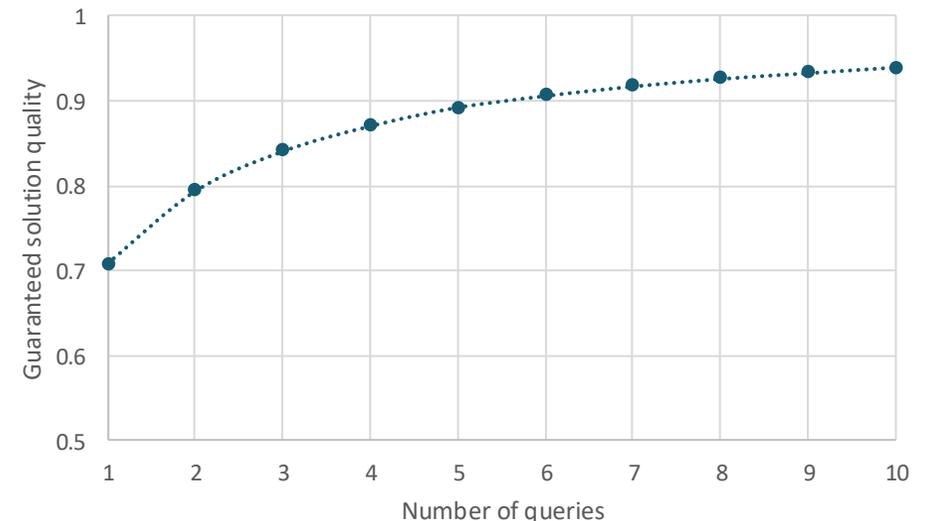
# Convergence Guarantee

*(How fast does the output solution approach the optimal?)*

## Theorem

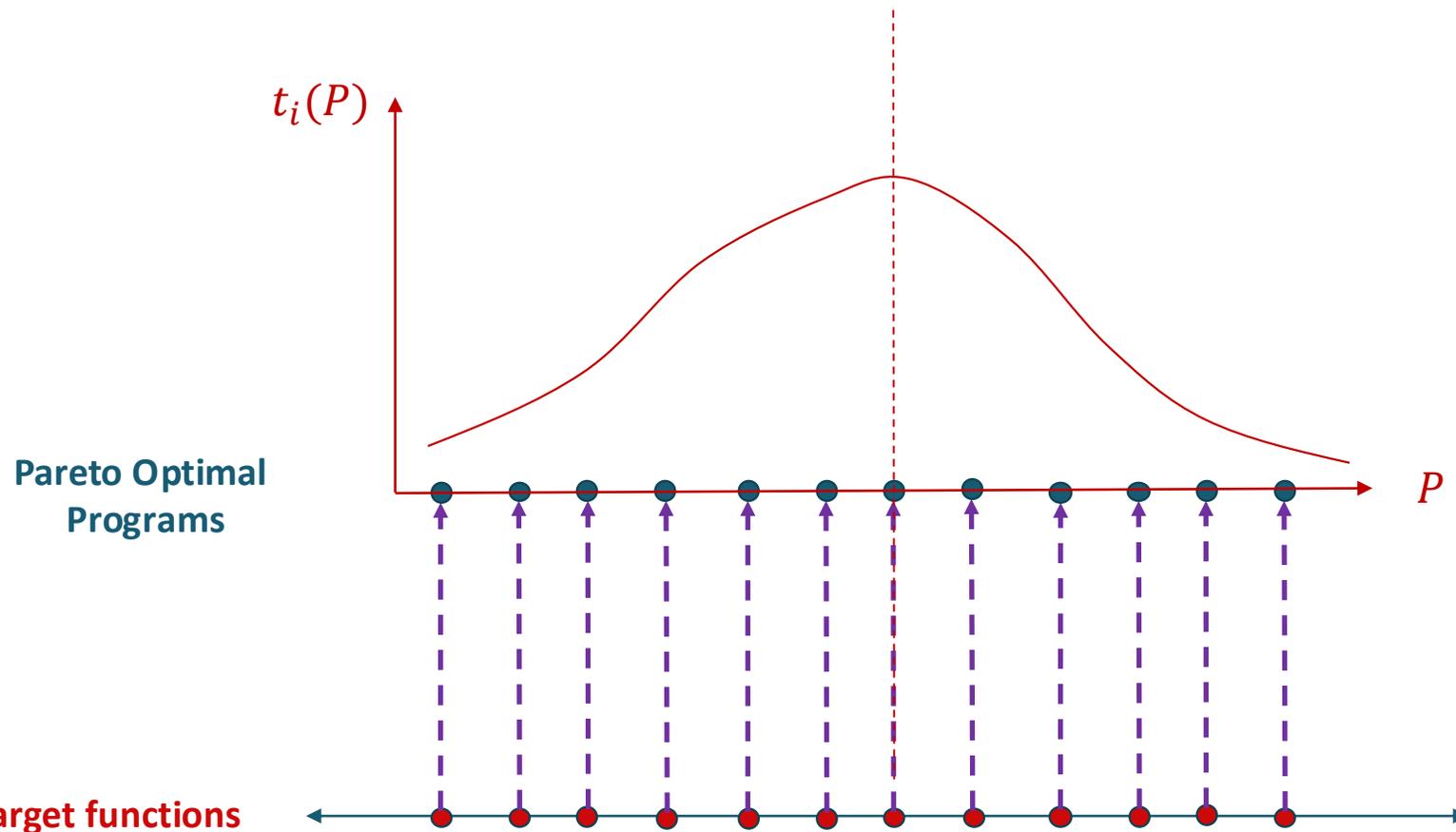
The voting-guided learning algorithm guarantees a **logarithmic** rate of convergence. (the median quality of solution is at least  $2^{\frac{-1}{n+1}}$  after  $n$  queries). The bound is tight.

**Proof idea:** Every query discards at least one randomly generated candidate.



# Can the algorithm converge faster?

(Yes, when the search space is *sortable*.)



## Example

If there are **two competing metrics** (e.g., *throughput* and *latency*) such that for each metric continued improvement leads to **diminishing marginal utility**, the search space is *sortable*.

# Can the algorithm converge faster?

---

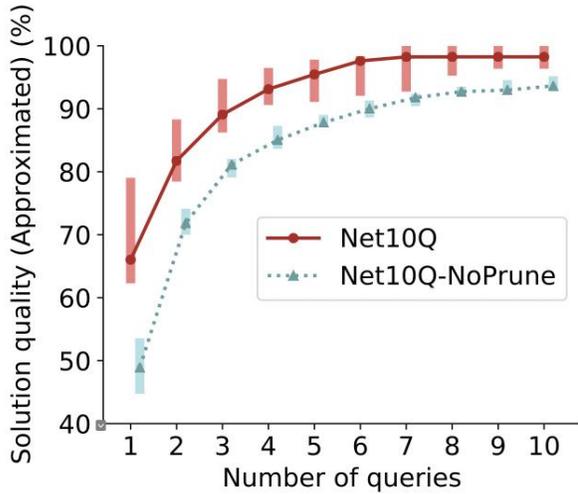
(Yes, when the search space is *sortable*.)

## Theorem

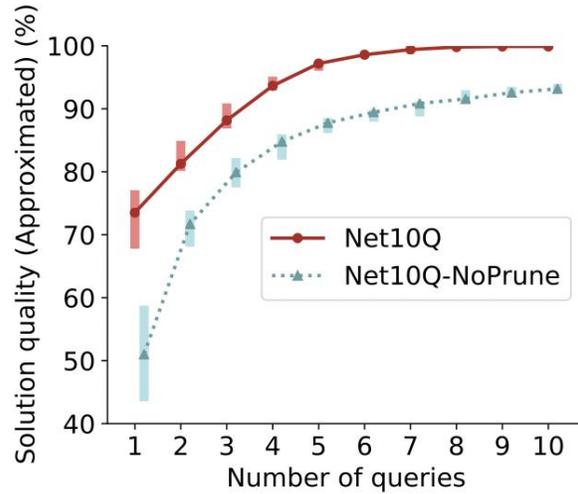
When the target function space is *sortable*, the voting-guided learning algorithm guarantees a *linear* rate of convergence. (the median quality of solution is at least  $1 - \frac{1}{\Omega(1.5^n)}$  after  $n$  queries). The bound is tight.

**Proof idea:** Every query discards at least **one third** of the candidates from the current PCS pool.

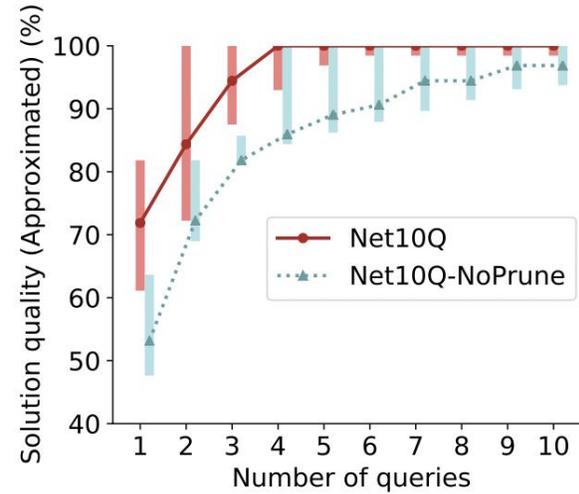
# Oracle-based Evaluation (Perfect Oracle)



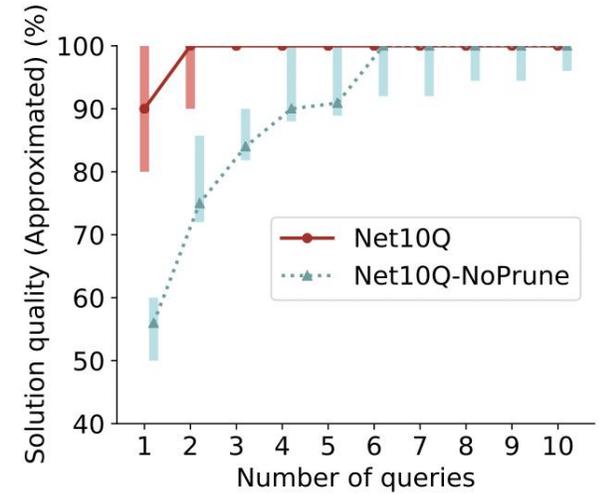
(a) MCF.



(b) BW.



(c) NF.



(d) OSPF.

**Net10Q** performed **constantly better** than **Net10Q-NoPrune**

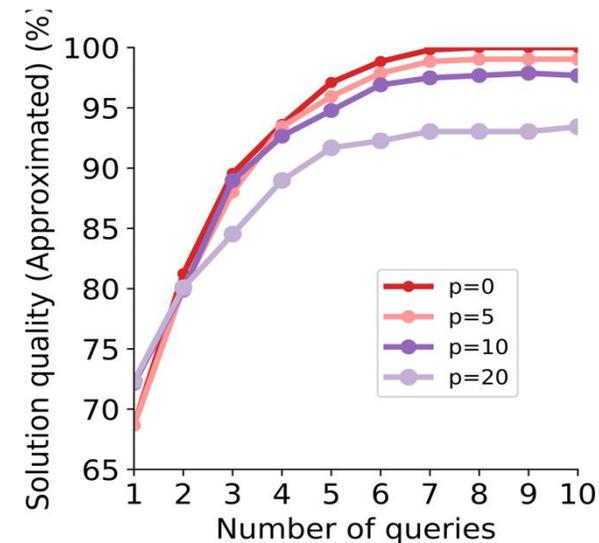
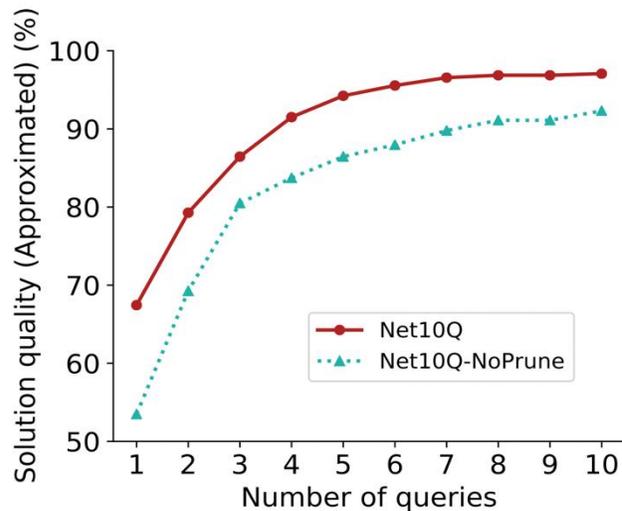
# Oracle-based Evaluation (Imperfect Oracle)

(BW on CWIX)

Imperfect oracle model:

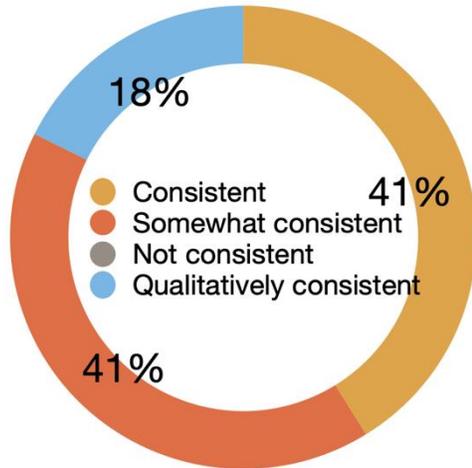
assigns a **random** reward that is sampled from a normal distribution (tunable by  $p$ )

Net10Q vs. Net10Q-NoPrune ( $p=10$ )

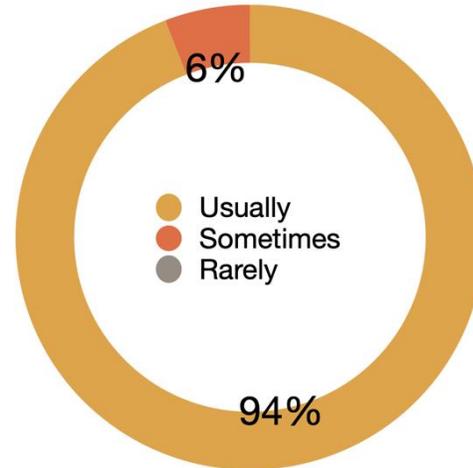


**Net10Q** can handle moderate feedback inconsistency

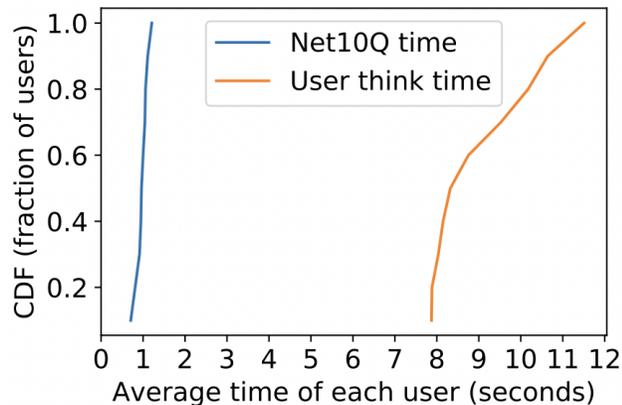
# User Feedback



Quality of recommendations



Was time taken acceptable?



Average time per query across users

## Qualitative Comments

- Most users are satisfied with **Net10Q's** recommendations and response time.
- *“The study was well done in my opinion. It put the engineer/architect in a position to make a qualified decision to try and chose the most reasonable outcome.”* – an expert user



# Can we learn formal specification from natural languages?

- Again, examples are not comprehensive, and good examples can be very expensive.
- Accurate specification may call for excessive examples
- *“Natural language will always remain the basic interpretation of, and reservoir for, the development of the artificial formalized languages of science.” – Doris Bradley*

# Pre-LLM Age

## FRET: Controlled Natural Language to Metric Temporal Logic (MTL)

### Create Requirement

Requirement ID	Parent Requirement ID	Project
REQ1-SEPARATION		ICAROUS

Rationale and Comments

Rationale

While in flight, a warning alert must be raised within 3 seconds of entering the warning hazard zone of an intruder (250 feet horizontal and 50 feet vertical).

Comments

Requirement Description

A requirement follows the sentence structure displayed below, where fields are optional unless indicated with "\*". For information on a field format, click on its corresponding bubble.

SCOPE   CONDITIONS   COMPONENT\*   SHALL\*   TIMING   RESPONSES\*

In flight mode, when (horizontal\_distance <=250) & (vertical\_distance <=50) the aircraft shall within 3 seconds satisfy warning\_alert

SEMANTICS

ASSISTANT   TEMPLATES   GLOSSARY

ENFORCED: in every interval where *flight* holds. TRIGGER: first point in the interval if  $((horizontal\_distance \leq 250) \& (vertical\_distance \leq 50))$  is true and any point in the interval where  $((horizontal\_distance \leq 250) \& (vertical\_distance \leq 50))$  becomes true (from false). REQUIRES: for every trigger, RES must hold at some point with distance  $\leq 3$  from the trigger (i.e., at trigger, trigger+1, ..., or trigger+3). If the interval ends sooner than trigger+3, then RES need not hold.

$M = flight, TC = ((horizontal\_distance \leq 250) \& (vertical\_distance \leq 50)), n = 3, Response = (warning\_alert).$

Diagram Semantics

Formalizations

Future Time LTL

Past Time LTL

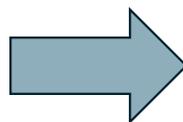
SIMULATE

# Pre-LLM Age

---

**Jdoctor:** translating Javadoc to JML-style specifications

```
/**
 * @param x the number to check, must be positive
 * @return true if x is prime, false otherwise
 * @throws IllegalArgumentException if x is negative
 */
public boolean isPrime(int x) { ... }
```



```
// Preconditions
assert x > 0;

// Exceptional Postconditions
if (x < 0) throw new IllegalArgumentException();

// Normal Postcondition
return isPrime(x);
```

# Pre-LLM Age

---

## Pros

- Pattern matching for semi-structured languages, which is predictable, understandable and adaptable
- Low computational cost

## Cons

- Limited generalization
- Limited handling of ambiguities
- Manually defined patterns

# NLP is the bread and butter of LLMs!

---

**Clarify When Necessary:  
Resolving Ambiguity Through Interaction with LMs**

**Can Large Language Models Write Good Property-Based Tests?**

**Finding Inductive Loop Invariants using Large Language Models**

**Can Large Language Models Transform Natural Language Intent into Formal Method Postconditions?**

**Tell Me More! Towards Implicit User Intention Understanding of Language Model Driven Agents**

---

**USER INTENT RECOGNITION AND SATISFACTION WITH LARGE LANGUAGE MODELS: A USER STUDY WITH CHATGPT**

---

*SpecGen*: Automated Generation of Formal Program Specifications via Large Language Models

---

**Can Large Language Models Reason about Program Invariants?**

---

# Open Question: How to evaluate LLM-generated specs?

---

Shuvendu Lahiri proposed the notions of soundness and completeness with respect to a set of input-output tests  $T$ :

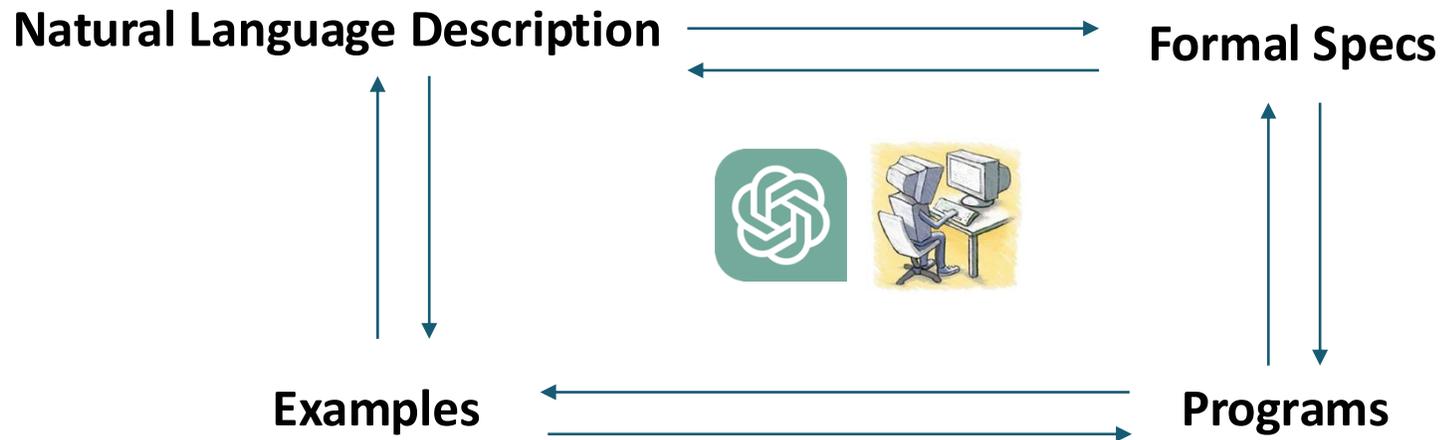
- **Soundness:** all tests in  $T$  satisfy  $\phi$
- **Complete measure:** if  $(i, o')$  is a mutation from a test  $(i, o) \in T$ , how likely  $(i, o')$  is *inconsistent* with  $\phi$

## Problems

- Higher completeness measure is not always better (e.g., output  $> 1.04562$  is likely overfitting; output  $> 1$  is more natural)
- What if test cases are not available (how about automatic generation of test cases?)

# Open Question: What's the best paradigm?

---



# Workflow 1

---

Natural Language Description



Programs

# Workflow 2

---

Natural Language Description



Formal Specs



Programs

# Workflow 3

---

Natural Language Description



Formal Specs

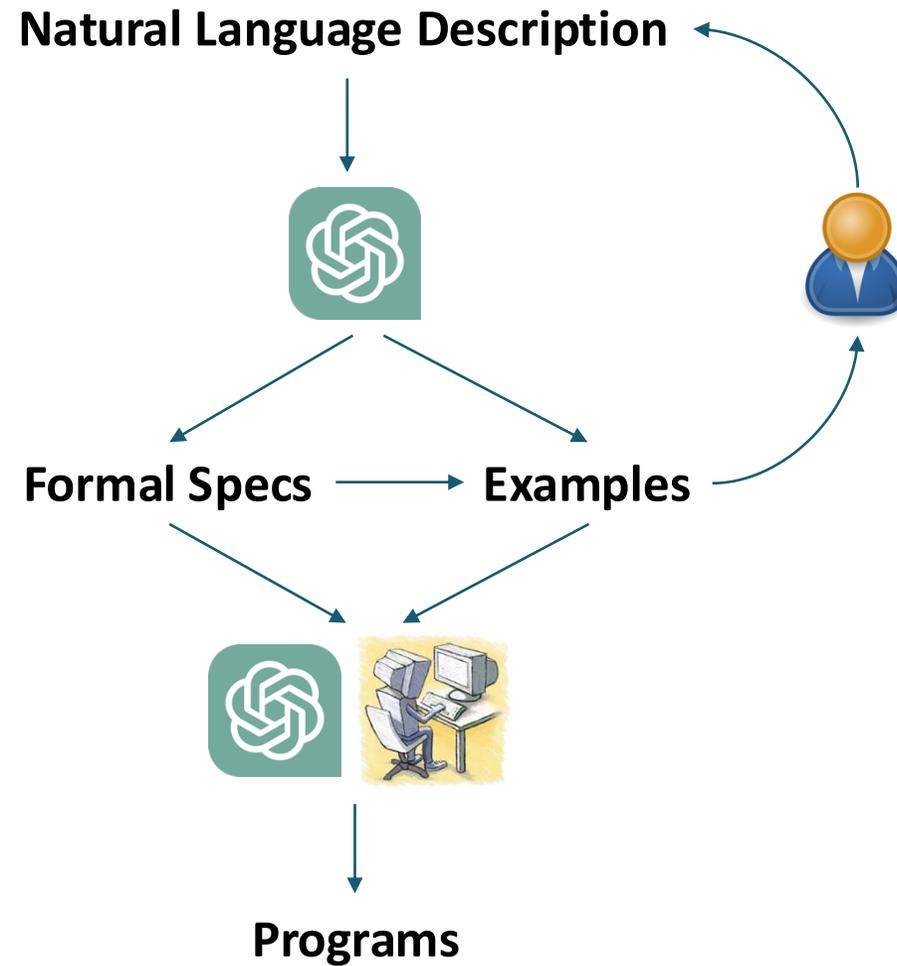
Examples



Programs

# Workflow 4

---



# Workflow n?

---



*Figure out your own  
workflow for your project!*