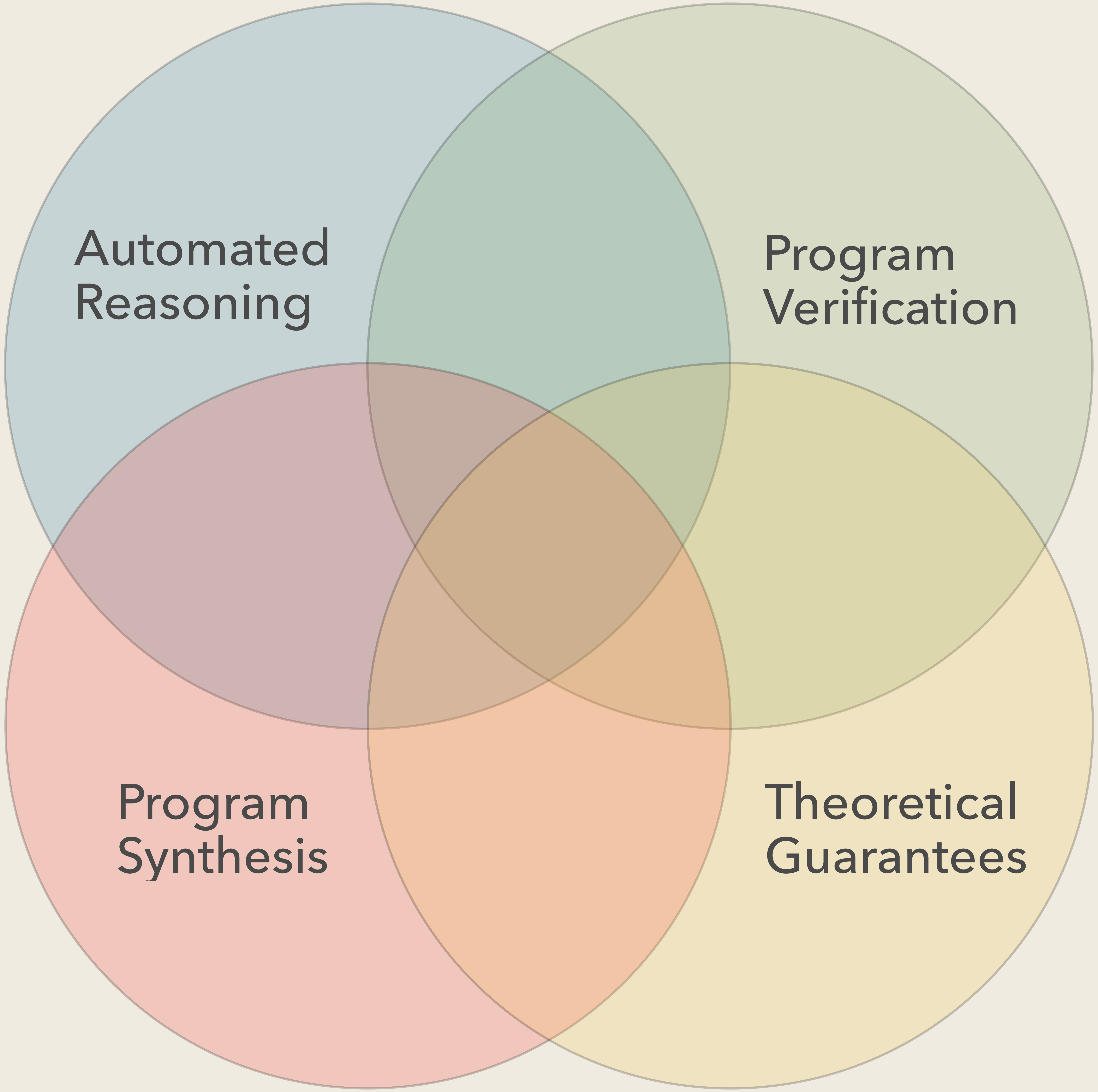
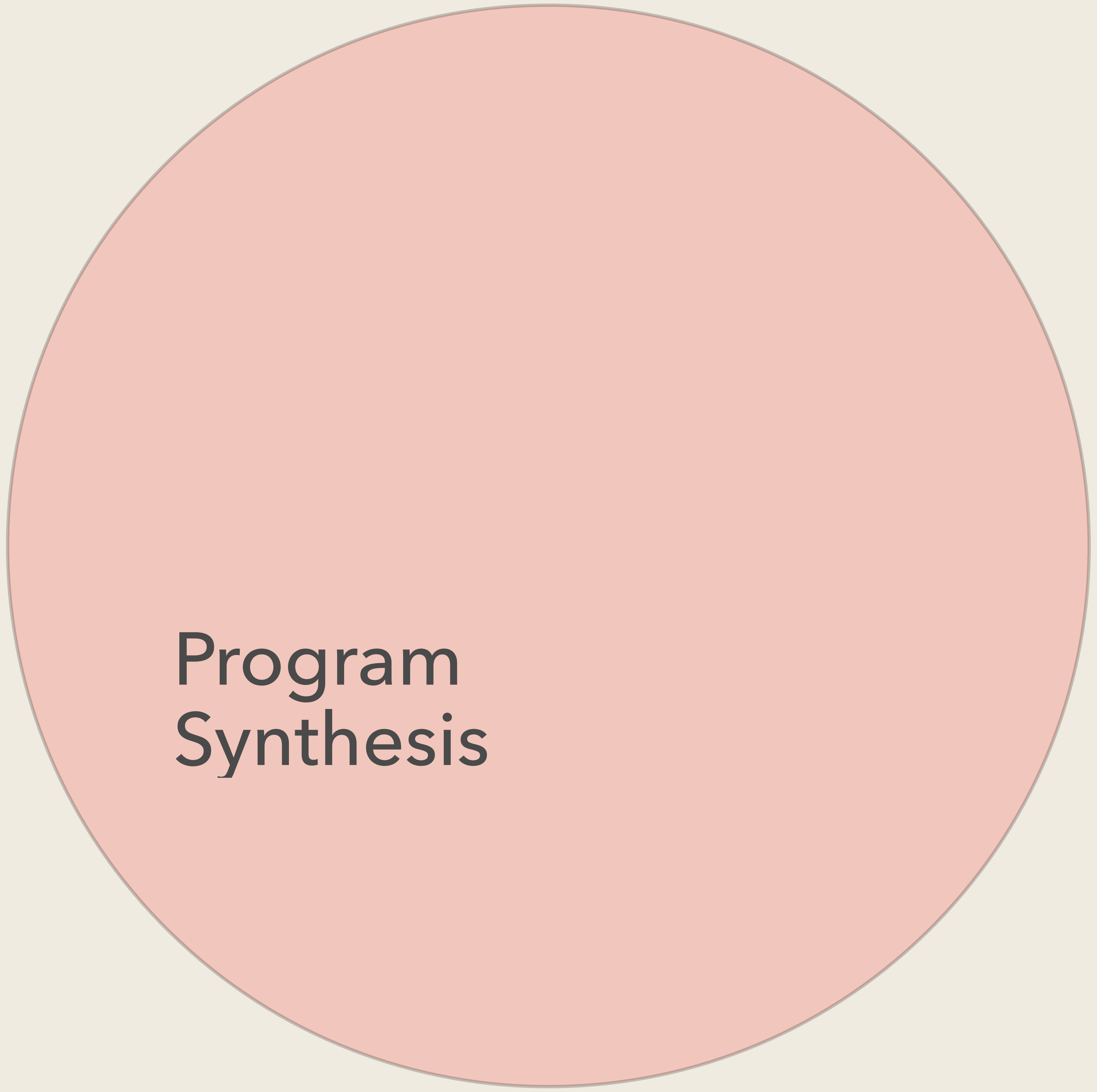
BRICS 2024 RUSSIA

ashoka UNIVERSITY

Одна голова хорошо, а две лучше: Symbolic Reasoning Meets AI

Aalok Thakkar


Automated
Reasoning
Program
Verification
Program
Synthesis
Theoretical
Guarantees


Program
Synthesis

Synthesis: Dreams $\Longrightarrow$ Programs

ZOHAR MANNA AND RICHARD WALDINGER

Abstract—Deductive techniques are presented for deriving programs systematically from given specifications. The specifications express the purpose of the desired program without giving any hint of the algorithm to be employed. The basic approach is to transform the specifications repeatedly according to certain rules, until a satisfactory program is produced. The rules are guided by a number of strategic controls. These techniques have been incorporated in a running program-synthesis system, called DEDALUS.

Many of the transformation rules represent knowledge about the program's subject domain (e.g., numbers, lists, sets); some represent the meaning of the constructs of the specification language and the target programming language; and a few rules represent basic programming principles. Two of these principles, the *conditional-formation rule* and the *recursion-formation rule*, account for the introduction of conditional expressions and of recursive calls into the synthesized program. The termination of the program is ensured as new recursive calls are formed.

Two extensions of the recursion-formation rule are discussed: a *procedure-formation rule*, which admits the introduction of auxiliary subroutines in the course of the synthesis process, and a *generalization rule*, which causes the specifications to be altered to represent a more general problem that is nevertheless easier to solve. Special techniques are introduced for the formation of programs with side effects.

The techniques of this paper are illustrated with a sequence of examples of increasing complexity; programs are constructed for list processing, numerical calculation, and array computation.

The methods of program synthesis can be applied to various aspects of programming methodology—program transformation, data abstrac-

Introduction

In recent years there has been increasing activity in the field of program verification. The goal of these efforts is to construct computer systems for determining whether a given program is correct, in the sense of satisfying given specifications. These attempts have met with increasing success; while automatic proofs of the correctness of large programs may be a long way off, it seems evident that the techniques being developed will be useful in practice, to find the bugs in faulty programs and to give us confidence in correct ones.

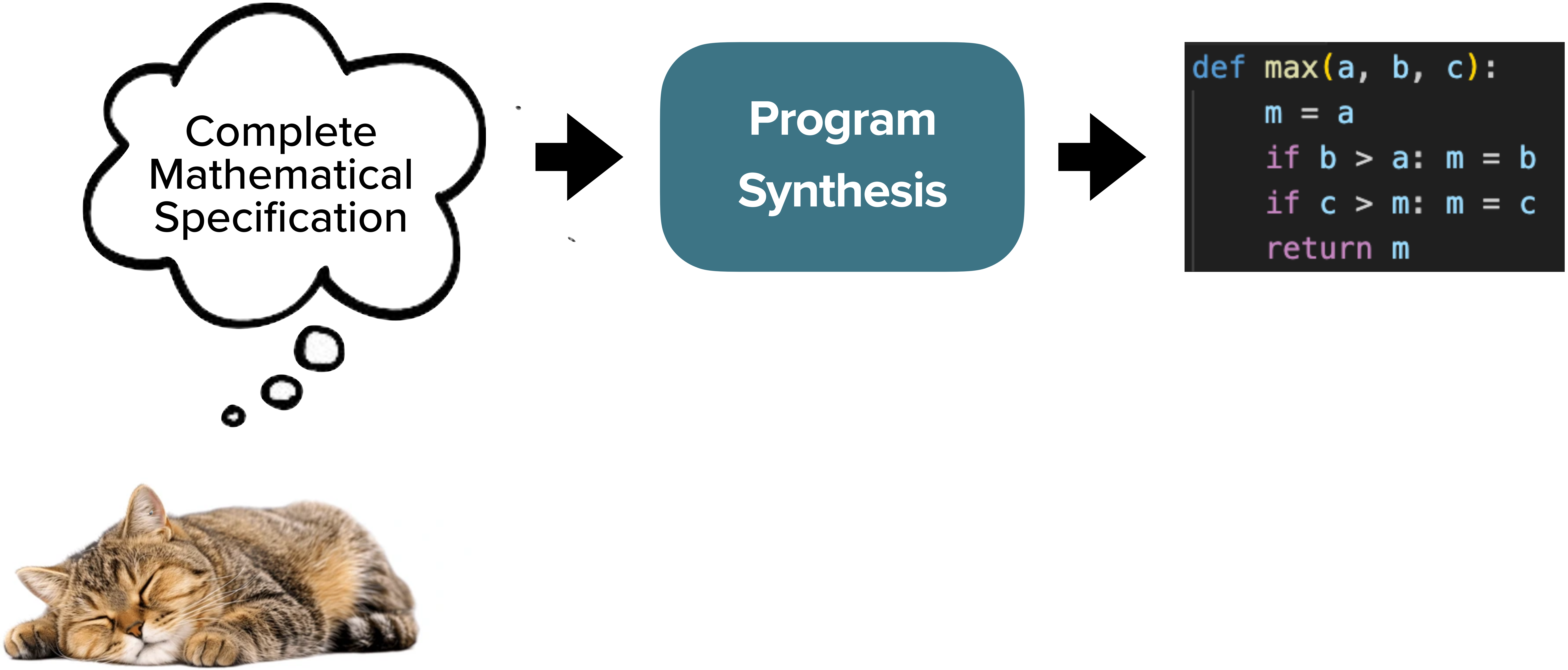
The general scenario of the verification system is that a programmer will present his completed computer program, along with its specifications and associated documentation, to a system which will then prove or disprove its correctness. It has been pointed out, most notably by the advocates of structured programming, that this is "putting the cart before the horse." Once we have techniques for proving program correctness, why should we wait to apply them until after the program is complete? Instead, why not ensure the correctness of the program while it is being constructed, thereby developing the program and its correctness proof "hand in hand"?

The point is particularly well-taken when we consider that

Specification	Synthesis	Verification
How to describe?	How to design?	How to ensure?

→ **Program Synthesis** →

```python
def max(a, b, c):
    m = a
    if b > a: m = b
    if c > m: m = c
    return m
```


Complete
Mathematical
Specification
Program
Synthesis
def max(a, b, c):
    m = a
    if b > a: m = b
    if c > m: m = c
    return m

Verification

```python
def max(a, b, c):
    m = a
    if b > a: m = b
    if c > m: m = c
    return m
```

$$\forall x, y, z \,.\, \big(max(x,y,z) \geq x\big) \wedge \big(max(x,y,z) \geq y\big) \wedge \big(max(x,y,z) \geq z\big)$$

$$\wedge \Big(\big(max(x,y,z) = x\big) \vee \big(max(x,y,z) = y\big) \vee \big(max(x,y,z) = z\big)\Big)$$

Verification

```python
def max(a, b, c):
    m = a
    if b > a: m = b
    if c > m: m = c
    return m
```

Function of three numbers that returns their max.

Write python code for a function of three numbers that returns their max.

Here’s the Python code for a function that returns the maximum of three numbers using comparisons:

python Copy code

```python
def max_of_three(a, b, c):
    if a >= b and a >= c:
        return a
    elif b >= a and b >= c:
        return b
    else:
        return c

# Example usage
result = max_of_three(5, 10, 3)
print(f"The maximum value is: {result}")
```

Mathematical Specification

Input-Output Examples

Natural Language

Input-Output Examples

```python
def max(a, b, c):
    m = a
    if b > a: m = b
    if c > m: m = c
    return m
```

$$max(1,2,4) = 4$$

$$max(8,3,-4) = 8$$

$$max(10,10,1) = 10$$

$$max(1,6,4) = 6$$

Easy to describe

Search Problem
Easy to automate

Easy to ensure!

$$max(1,2,4) = 4$$
$$max(8,3,-4) = 8$$
$$max(10,10,1) = 10$$
$$max(1,6,4) = 6$$

Program Synthesis

```python
def max(a, b, c):
    m = a
    if b > a: m = b
    if c > m: m = c
    return m
```

Correctness?

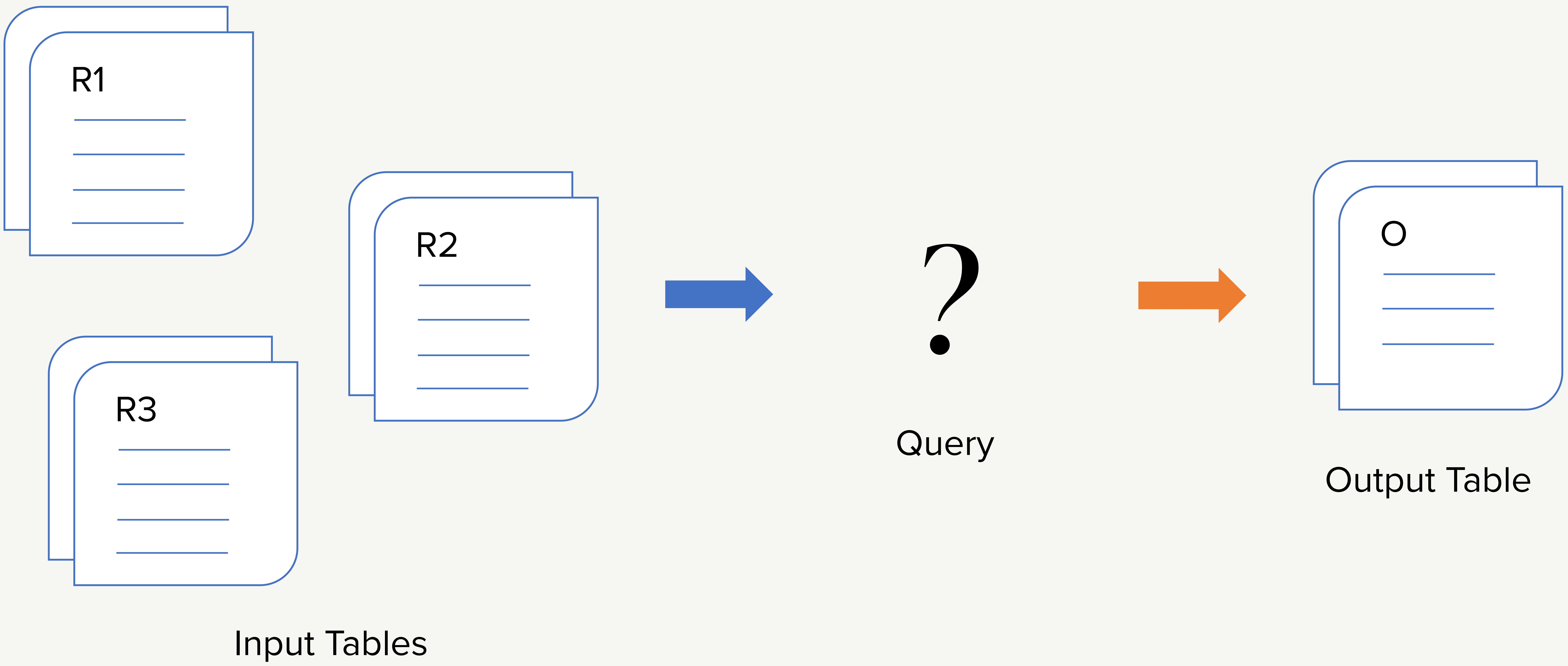
Ambiguity?

Overfitting?

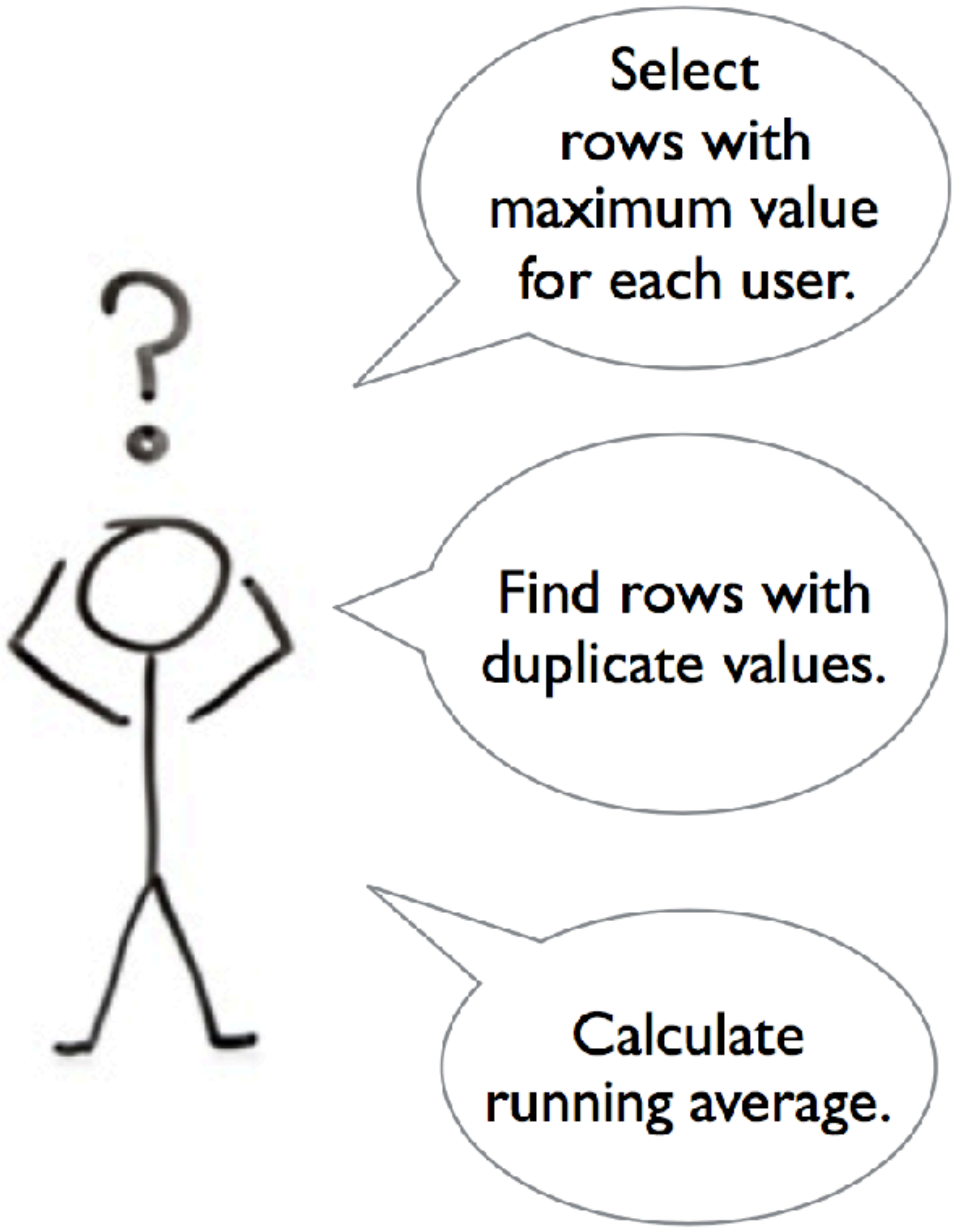
Scalability?

Example-guided Synthesis of Relational Queries

Query Synthesis



End User



stackoverflow

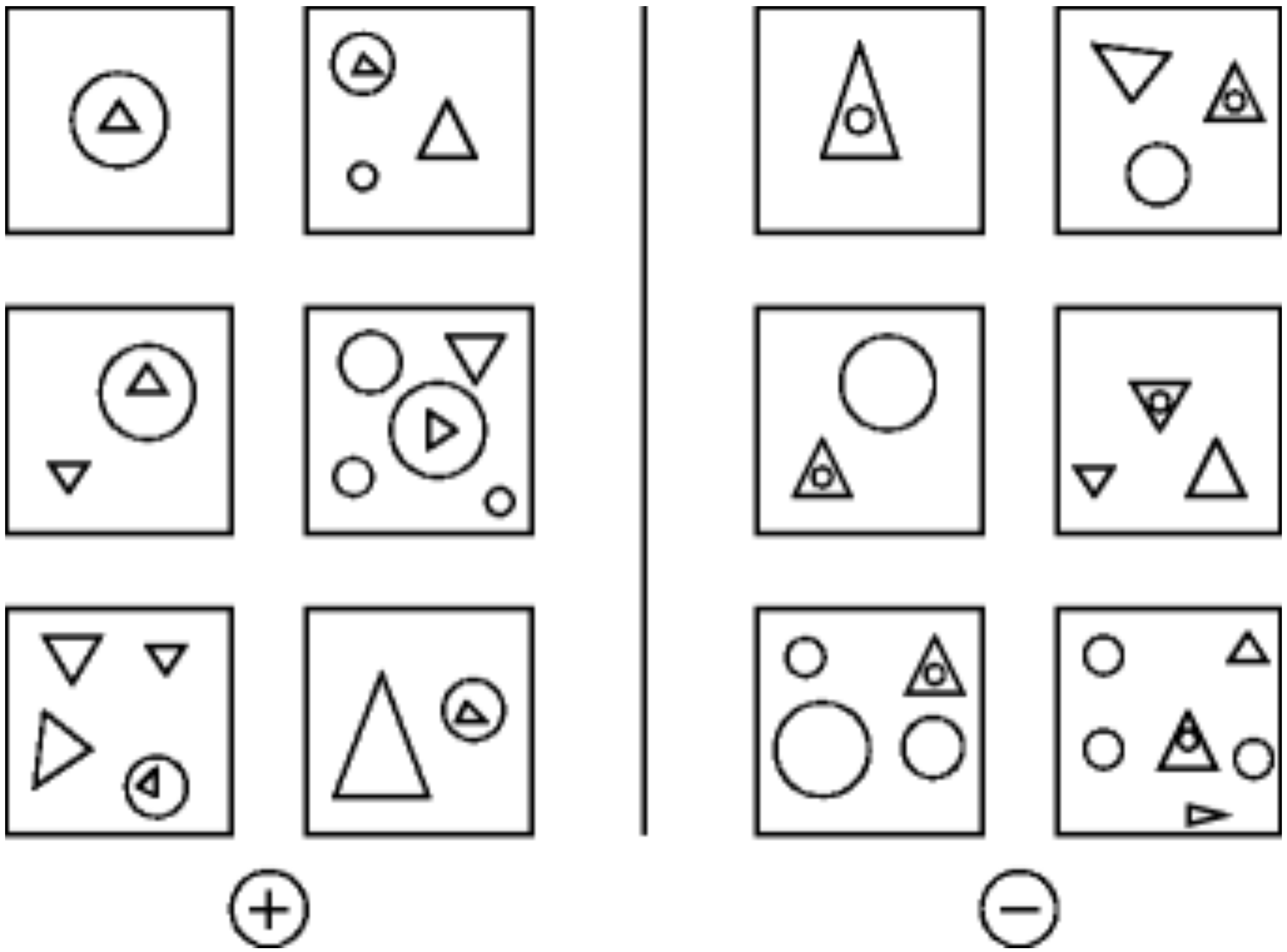
SQL

```sql
Select x.id, x.customer, x.total
From PURCHASES x
Join (Select p.customer,
             Max(total)
        From PURCHASES p
        Group By p.customer) y
On y.customer = x.customer
   And y.max_total = x.total
```

```sql
Select *
From  Users a
Where Exists
   (Select *
    From Users b
    Where (a.name = b.name
           Or  a.email = b.email)
          And a.ID <> b.id)
```

```sql
Select a.ord, a.val, Avg(b.val)
From t As a Join t As b
Where b.ord <= a.ord
Group By a.ord,a.val
Order By a.ord
```

Bongard problem 47



Supervised Learning

1. Small, human interpretable explanations
2. Sound inference:

   The learned hypothesis correctly *explains* the given data
3. Robust and Generalizable:

   Maintained performance against noise and outliers
4. Completeness:

   Tool returns an impossibility proof when there is no solution
5. Scalability:

   With respect to dimensionality (features) and sample size

Supervised Learning

Scalable

Sound

Generalizable

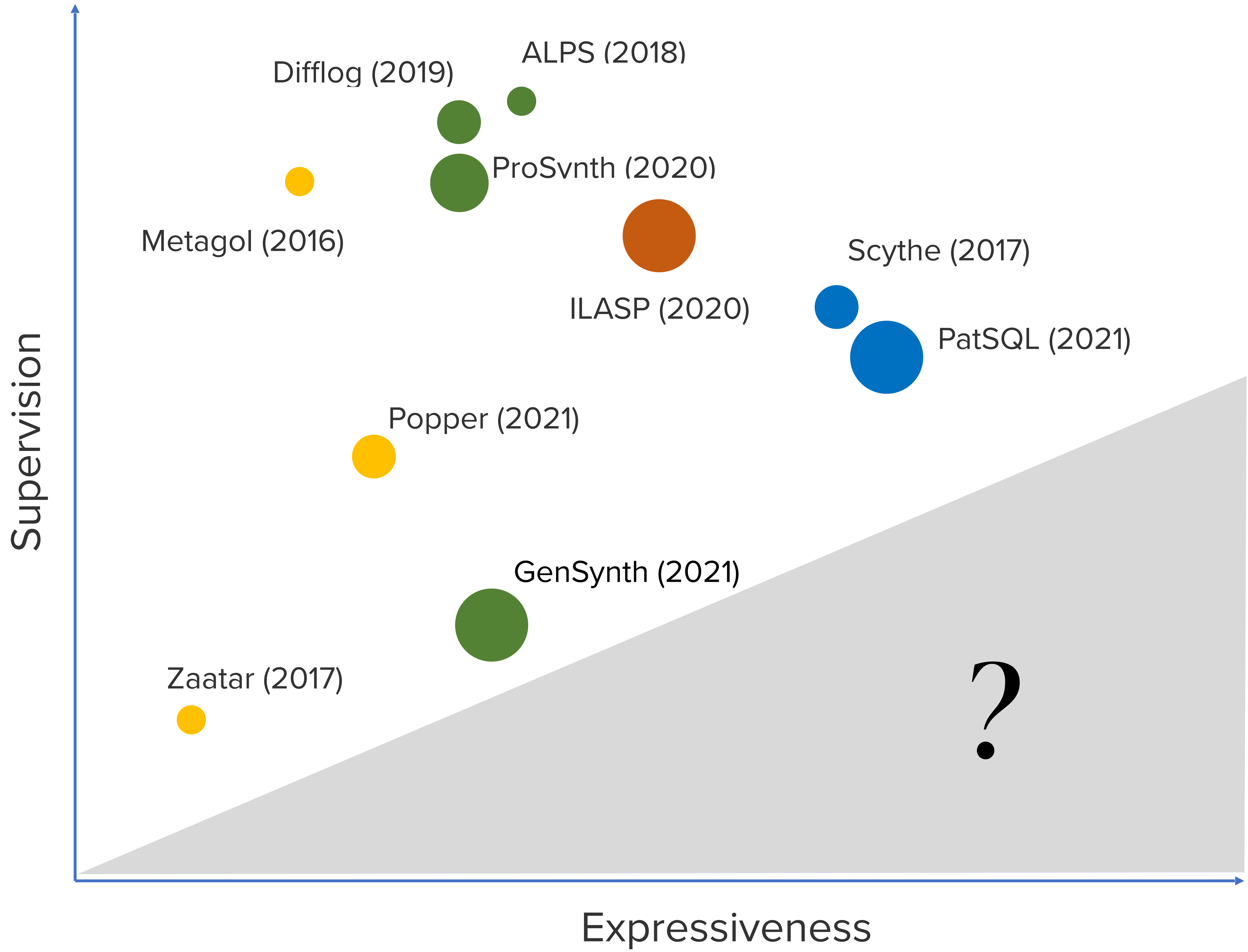
Inference + Guarantees

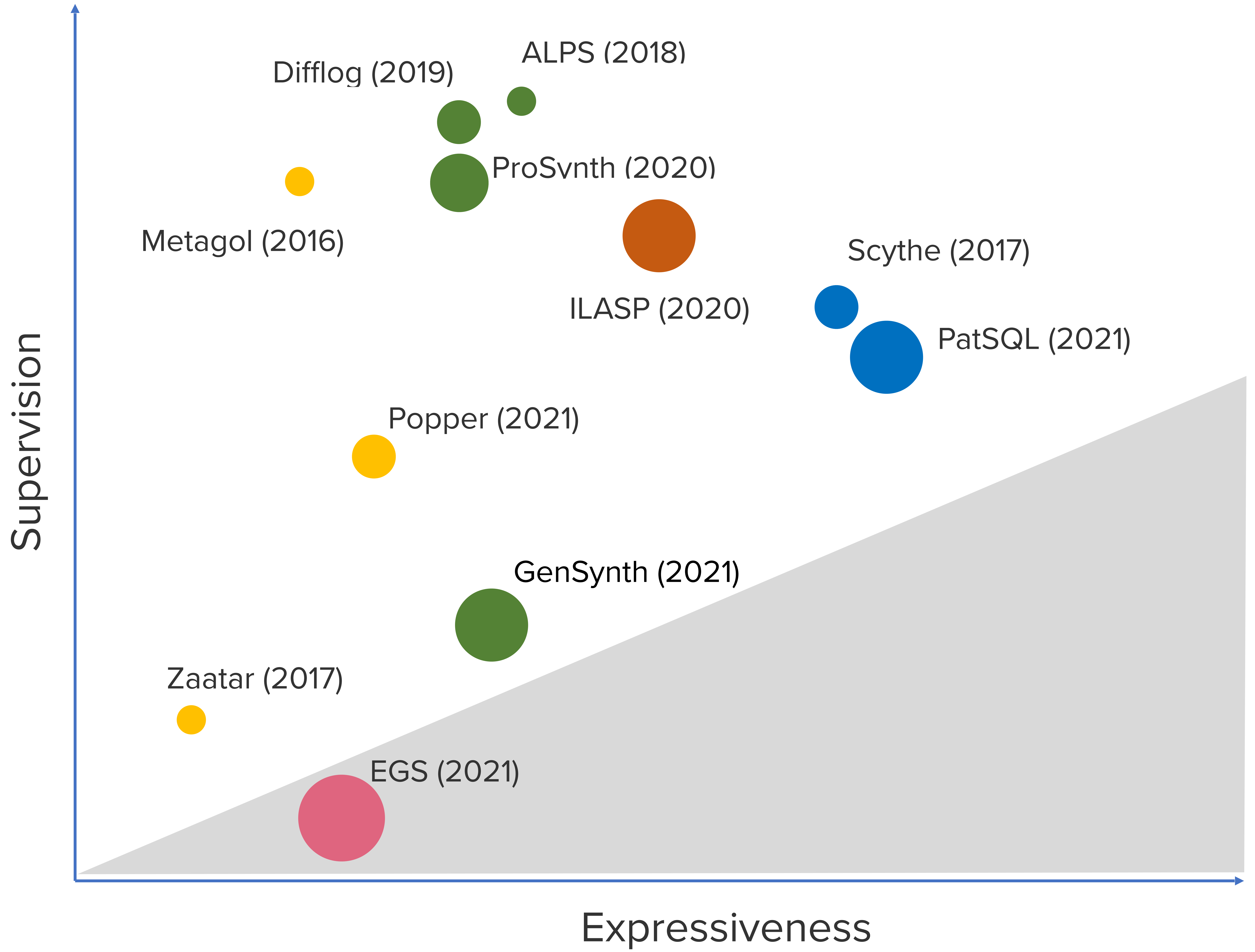
Explainable

Adaptable

Robust

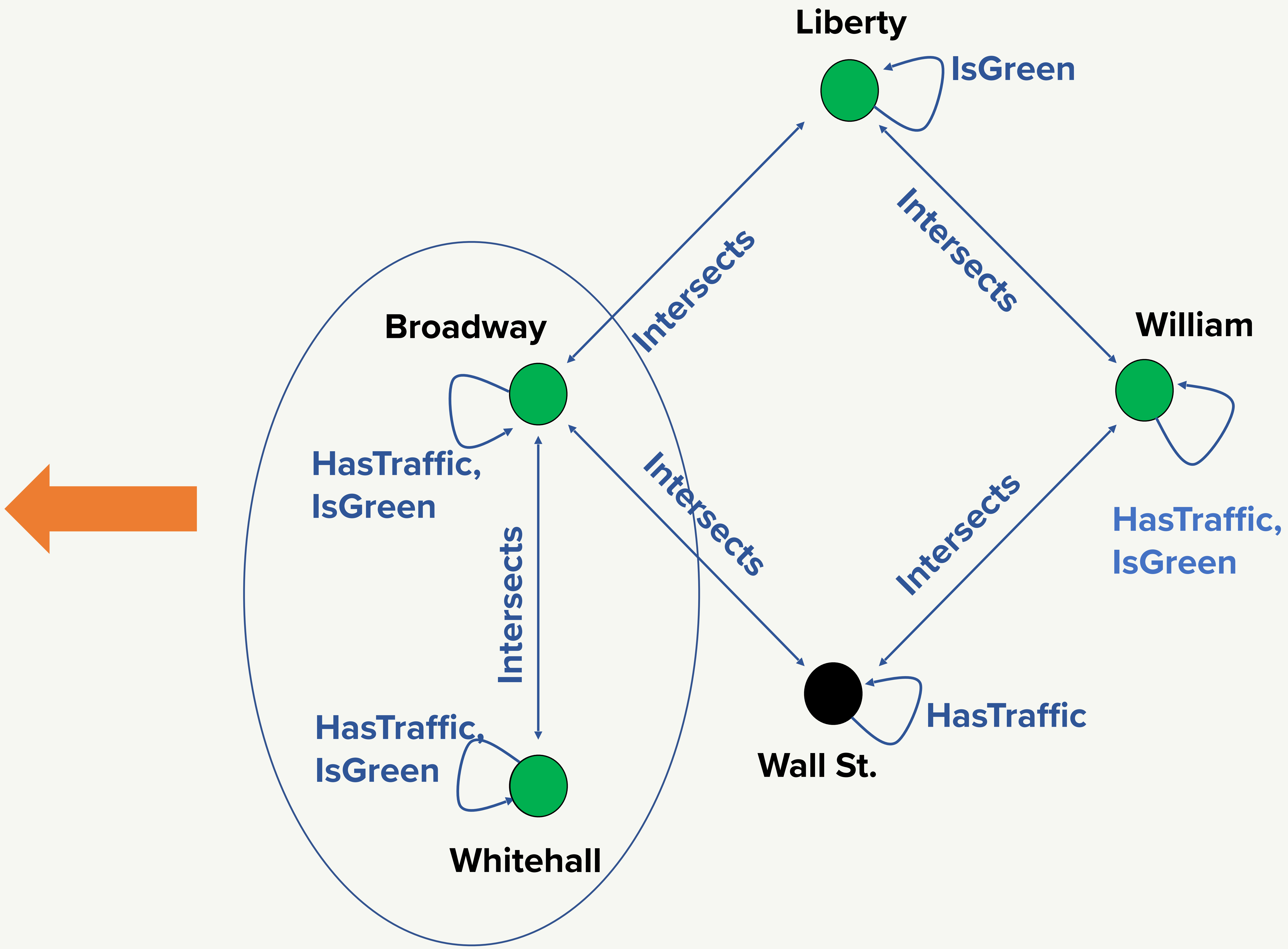
Одна голова хорошо, а две лучше

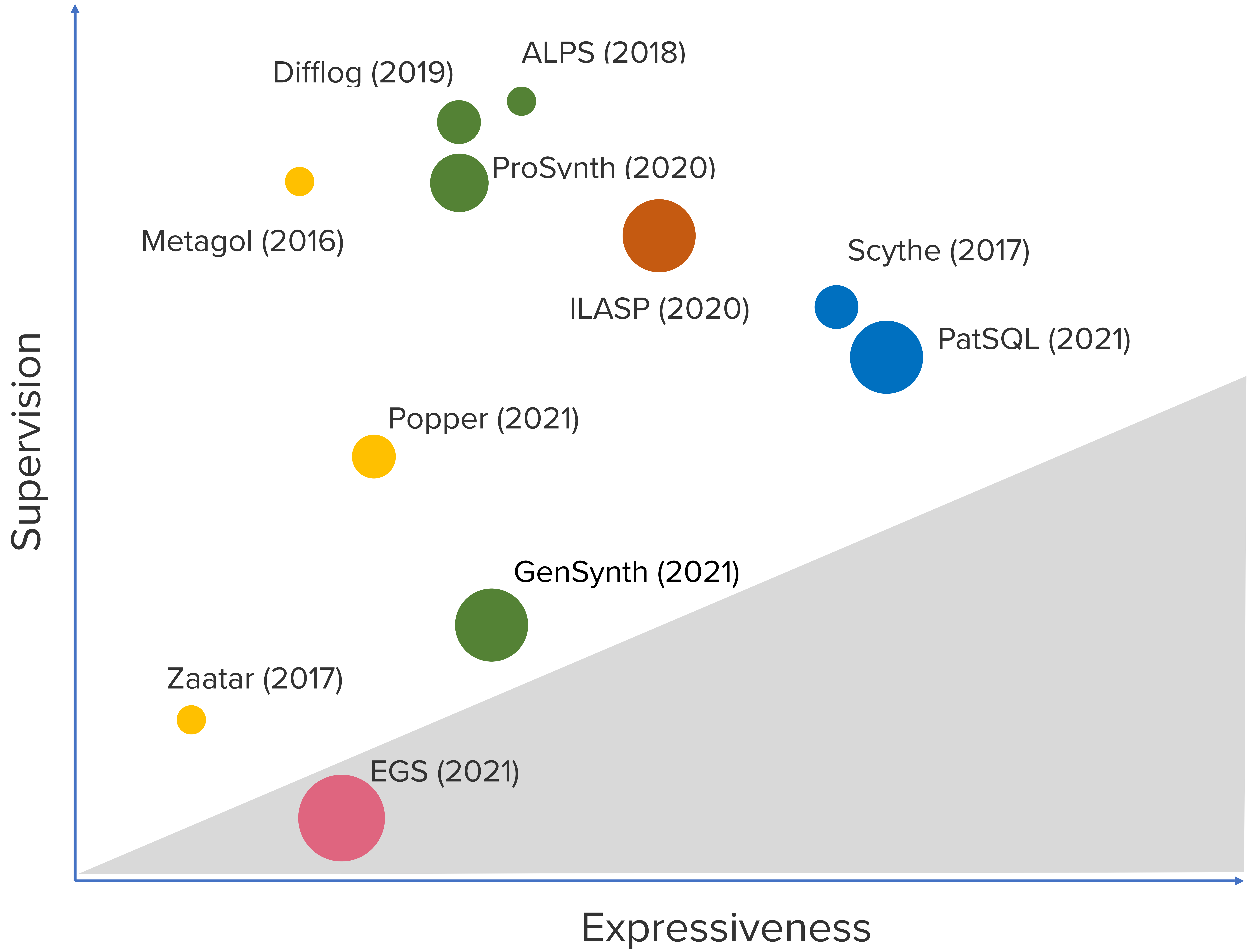

Difflog (2019)
ALPS (2018)
ProSynth (2020)
Metagol (2016)
ILASP (2020)
Scythe (2017)
PatSQL (2021)
Popper (2021)
GenSynth (2021)
Zaatar (2017)
?
Supervision
Expressiveness

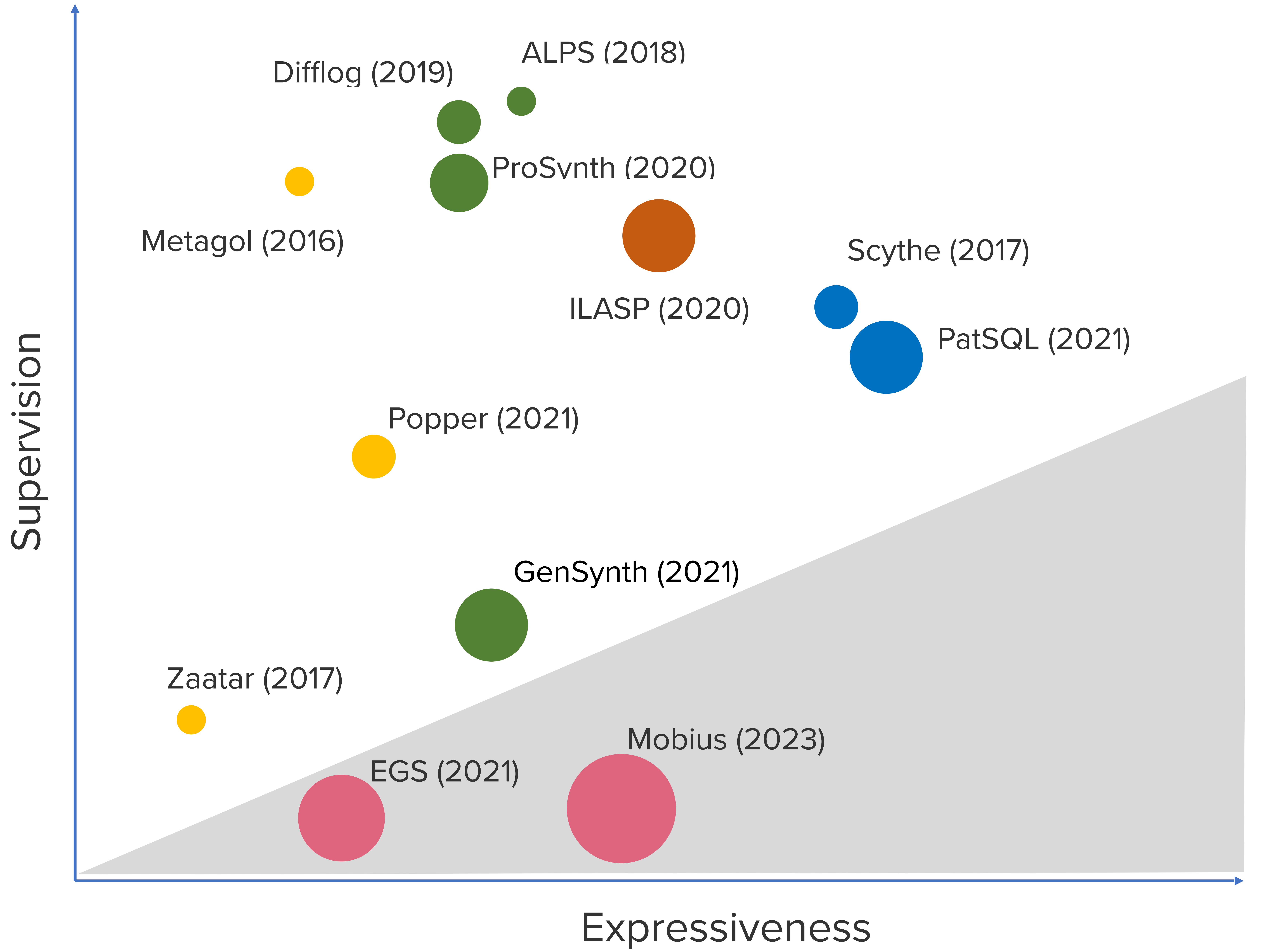

ALPS (2018)
Difflog (2019)
ProSynth (2020)
Metagol (2016)
ILASP (2020)
Scythe (2017)
PatSQL (2021)
Popper (2021)
GenSynth (2021)
Zaatar (2017)
EGS (2021)
Supervision
Expressiveness

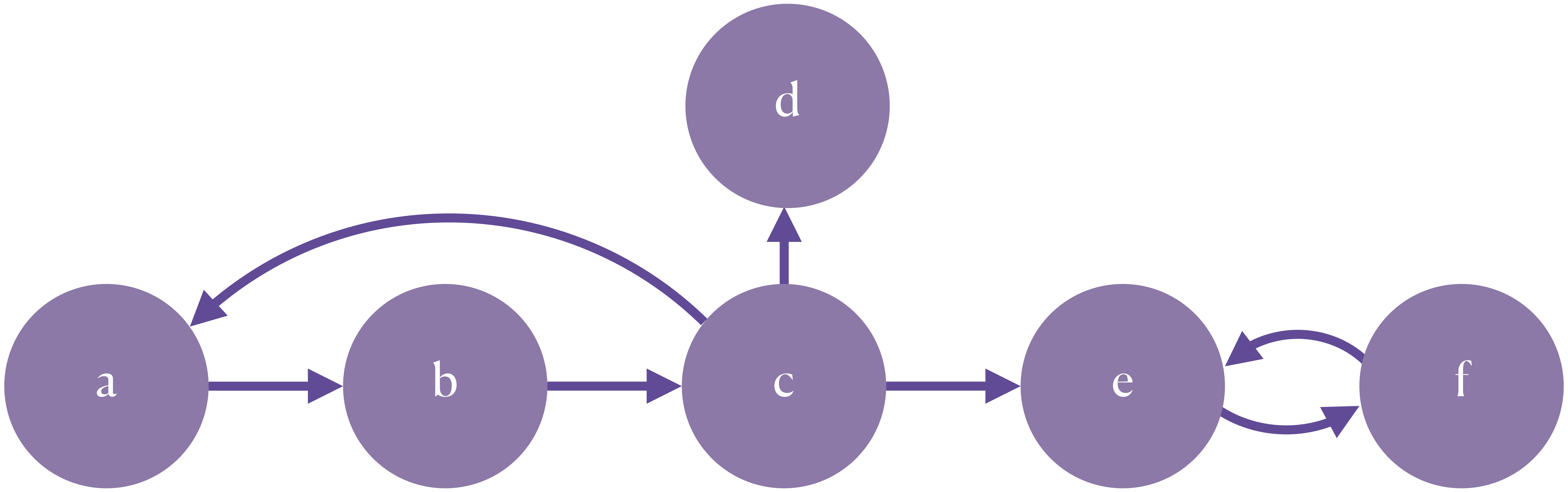
Example-guided Synthesis

Crashes(x) :– HasTraffic(x), isGreen(x),
Intersects(x, y),
HasTraffic(y), isGreen(y).

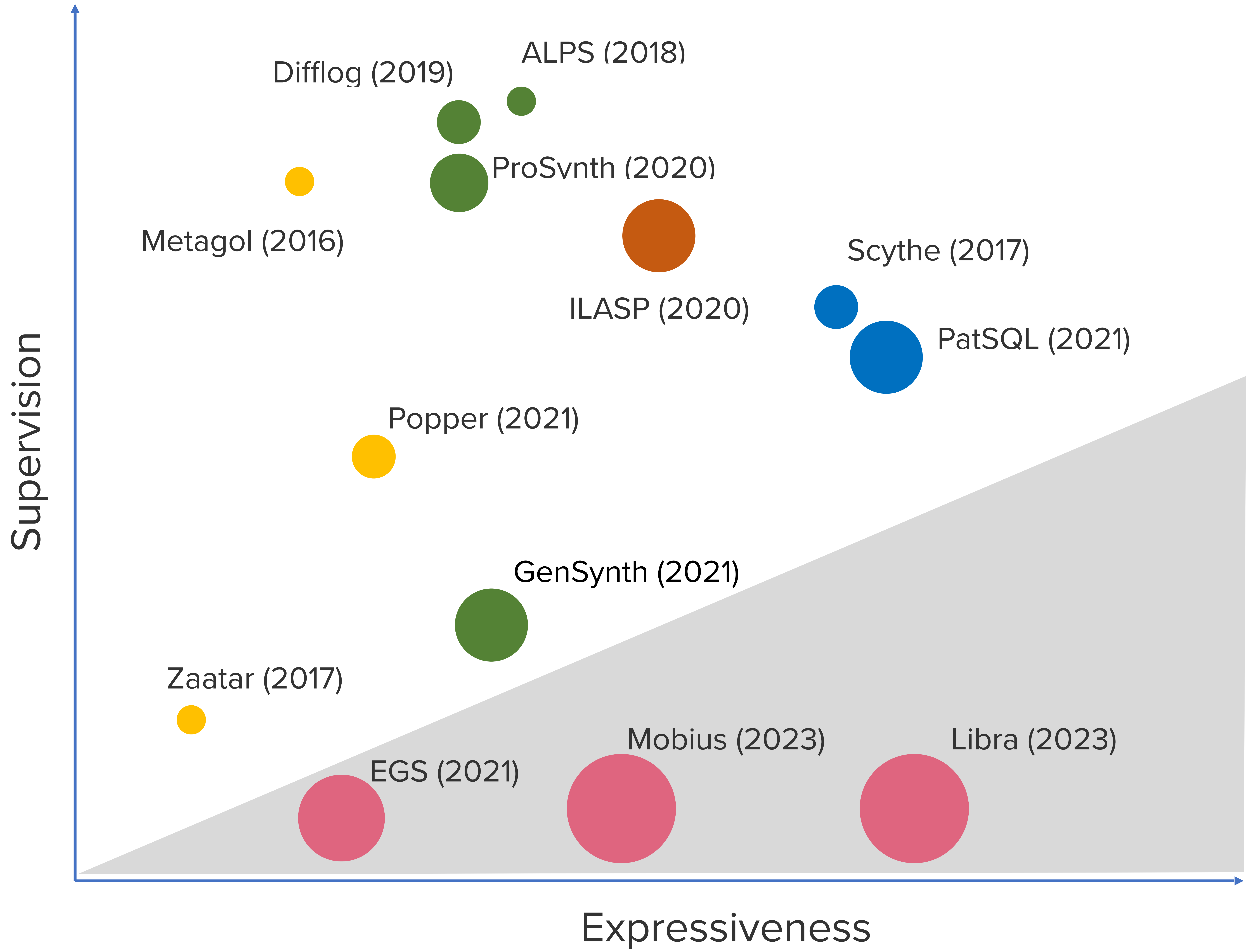



Difflog (2019)
ALPS (2018)
ProSynth (2020)
Metagol (2016)
ILASP (2020)
Scythe (2017)
PatSQL (2021)
Popper (2021)
GenSynth (2021)
Zaatar (2017)
EGS (2021)
Supervision
Expressiveness


ALPS (2018)
Difflog (2019)
ProSynth (2020)
Metagol (2016)
Scythe (2017)
ILASP (2020)
PatSQL (2021)
Popper (2021)
GenSynth (2021)
Zaatar (2017)
Mobius (2023)
EGS (2021)
Supervision
Expressiveness



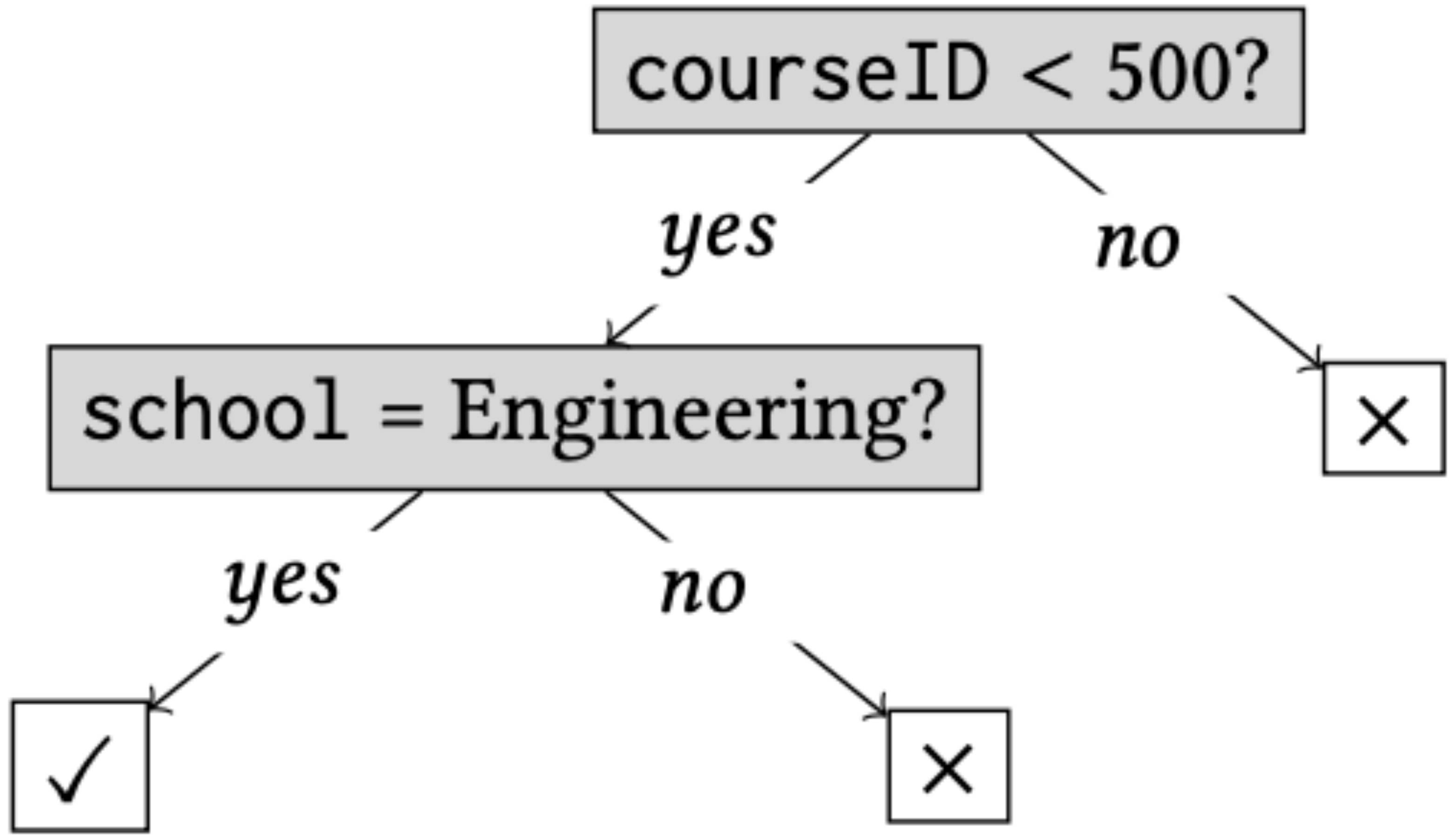
scc(x, y) :− path(x, y), path(y, x).
path(x, y) :− edge(x, y).
path(x, y) :− path(x, z), path(z, y).



ALPS (2018)
Difflog (2019)
ProSynth (2020)
Metagol (2016)
ILASP (2020)
Scythe (2017)
PatSQL (2021)
Popper (2021)
GenSynth (2021)
Zaatar (2017)
Mobius (2023)
Libra (2023)
EGS (2021)
Supervision
Expressiveness

```sql
SELECT registration.studentID
    FROM registration JOIN department
        ON registration.deptCode = department.deptCode
    WHERE registration.courseID < 500
        AND department.school = "Engineering"
```

studentID	deptCode	courseID	school
Alice	Comp.	201	Engineering
Alice	Chem.	310	Arts and Science
Alice	Mech.	550	Engineering
Bob	Mech.	320	Engineering
Bob	Mech.	550	Engineering
Charlie	Chem.	310	Arts and Science
David	Comp.	500	Engineering
David	Mech.	502	Engineering
Erin	Chem.	310	Arts and Science



CENTRE FOR
Data Sciences and Analytics

SAFEXPRESS CENTRE FOR
Data, Learning, and Decision Sciences

CENTER FOR
Digitalisation, AI, and Society

Koita Centre for Digital Health

CENTRE FOR

Data Sciences and Analytics

Comprehensive Data Lake Framework:

1. Repository of multimodal across interdisciplinary fields
2. Metadata of open source/public data
3. Unified access and integration
4. Inference, versioning, and provenance

All Centres will use CDA as its infrastructural foundation

Centre for Data, Learning, and Decision Sciences

Centre for Digitisation, AI, and Society

Centre for Health Analytics, Research and Trends (CHART)

Koita Centre for Digital Health

Centre for Economic Data and Analysis

- Climate
- Ecology
- Languages
- History
- Astronomy
- Nutrition and Food
- Health
- Epidemiology
- Traffic and Pollution
- Agriculture
- GIS

Building AI (with guarantees) as a tool

SAFEXPRESS CENTRE FOR

Data, Learning, and Decision Sciences

- Data-driven quantitative modelling (weather, epidemiology, cultural behaviour)
- Financial Mathematics (risk, pricing, optimisation)
- Reinforcement Learning
- **Automated Reasoning**

AI as an agent, and its interaction with society

Brazilian Artificial Intelligence Strategy (EBIA)

Russia: National AI Strategy

IndiaAI Mission, Responsible AI (2021)

China: New Generation AI Development Plan

South Africa: National AI Plan

CENTER FOR
Digitalisation, AI, and Society

- Voting Protocols and Their Properties
- Privacy and Integrity of Electoral Rolls
- Electronic Voting
- Applications of Blockchains
- Digitalisation in Healthcare
- Cryptocurrency Regulation
- Computational Techniques for Census
- AI for Social Good
- Robust, Fair, and Explainable AI
- Ethics of Computing

AADHAAR

120 crore
biometric records

Ayushman Bharat Digital Mission
Building Digital Health Ecosystem

25 crore
linked health records

UPI
UNIFIED PAYMENTS INTERFACE

36 crore
daily transactions

Personal Health & Wellness

- Generation and use of personalised health data to identify risks, promote wellness, and reinforce healthy behaviour
- Genetic disease screening
- Use of wearables & healthcare apps

Precision Public Health

- Integrating multi-modal information for multi-scale precision health
- Population cohorts, convenience cohorts, biobanks
- Precision Medicine and Precision Public Health

Intersections

- Assessing impact of food choices on health
- Promoting appropriate choices in foods
- Learning from history of medicine for digital health/ AI policy

AI + Health Data

- Developing a health data & analytics ecosystem for preventive and personalised medicine
- Ethical, purpose based, privacy preserving health data architectures that promote appropriate uses, while minimising risks to individuals
- Use of LLMs to empower citizens & public institutions with fit-for-purpose information

Koita Centre for Digital Health

CENTRE FOR
Data Sciences and Analytics

SAFEXPRESS CENTRE FOR
Data, Learning, and Decision Sciences

CENTER FOR
Digitalisation, AI, and Society

Koita Centre for Digital Health

Aalok Thakkar

aalok.thakkar@ashoka.edu.in

csdept@ashoka.edu.in

aalok-thakkar.github.io