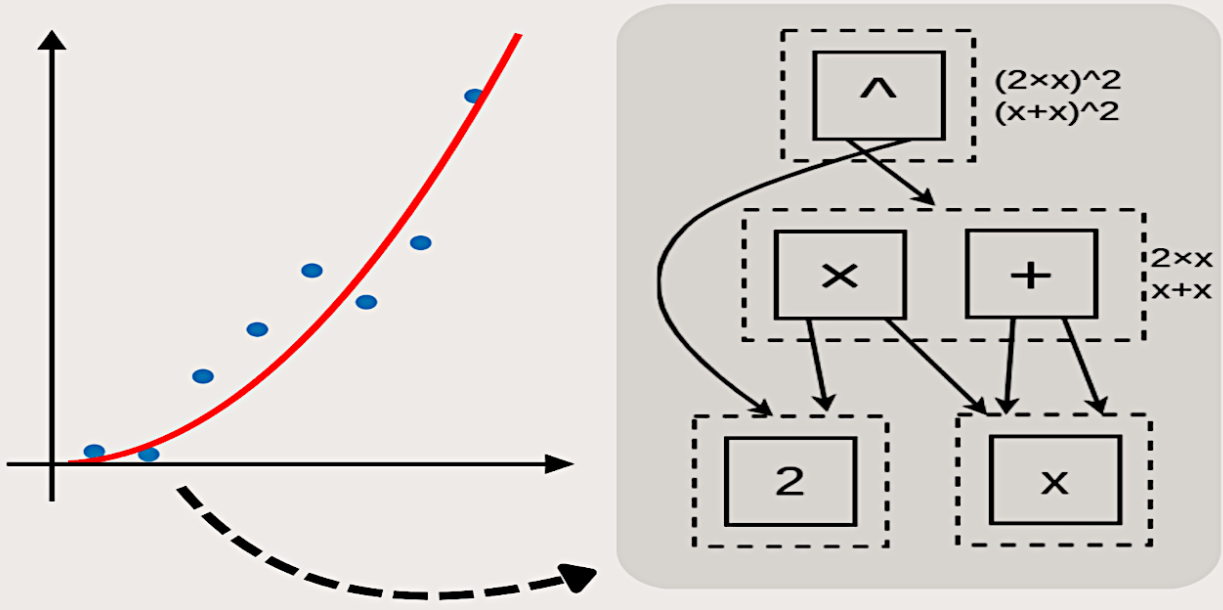
# Equality Graph Assisted Symbolic Regression




Fabrício Olivetti de França, Gabriel Kronberger

Federal University of ABC & University of Applied Sciences Upper Austria

29 April, 2025

# About me

# Who am I?


- Working with Symbolic Regression since 2018:
  - model interpretability
  - lightweight methods
  - improving navigability
  - incorporating domain knowledge
  - post analysis: identifiability, redundant parameters, simplification, confidence intervals

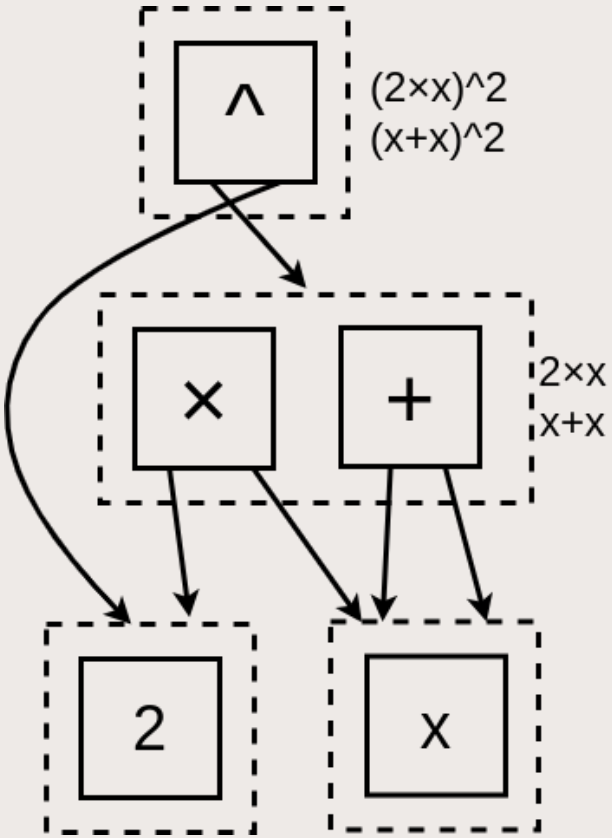
# Where am I?


Currently working at UFABC (Santo Andre - Brazil):

- Associated professor (completed the supervision of 22 undergrad, master, and Ph.D. students)
- Head of Heuristics, Analysis and Learning Laboratory (HAL)
- Coordinator of the gradute program in Computer Science.

# Equality Graphs for Babies

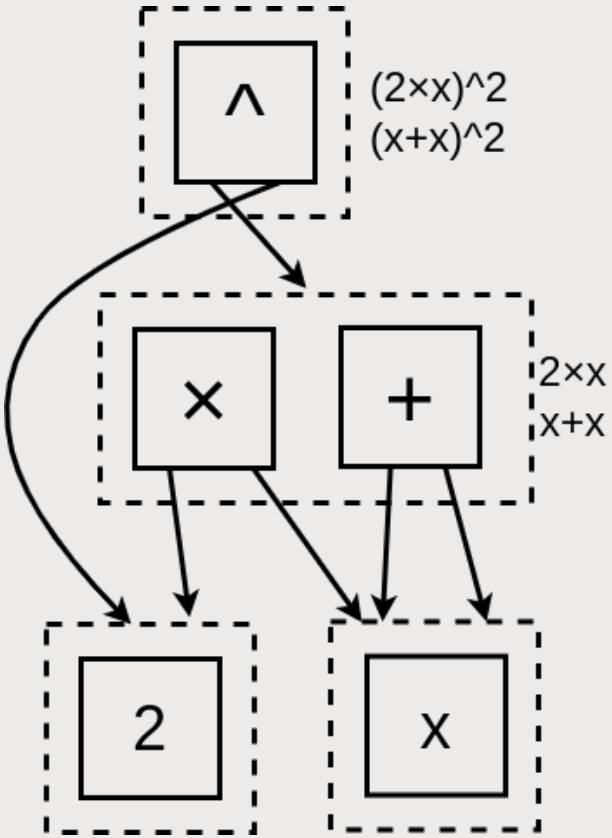
# Hi, I am an Equality Graph!




Hi, I’m an **equality graph**, but you can call me **e-graph**!

I look exactly like an ordinary graph, but with some added **stripes**!

# I can store equivalences

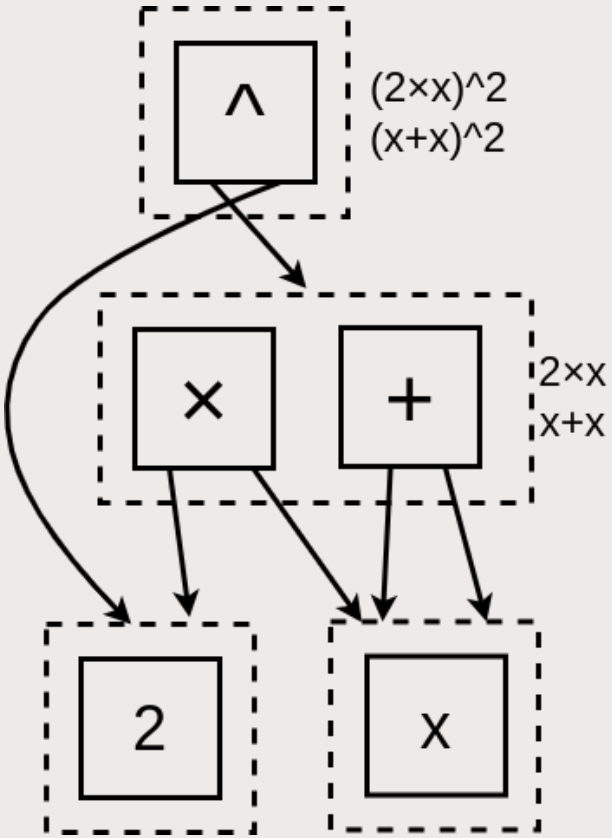



These “stripes” show the equivalence relationships I’ve learned so far.

If we follow the path from the middle ×, we will build the expression $2 \times x$. But if we follow the +, we will build $x + x$.

But they are the same!! I group them together so you can choose the best expression for the occasion.

# Terminology

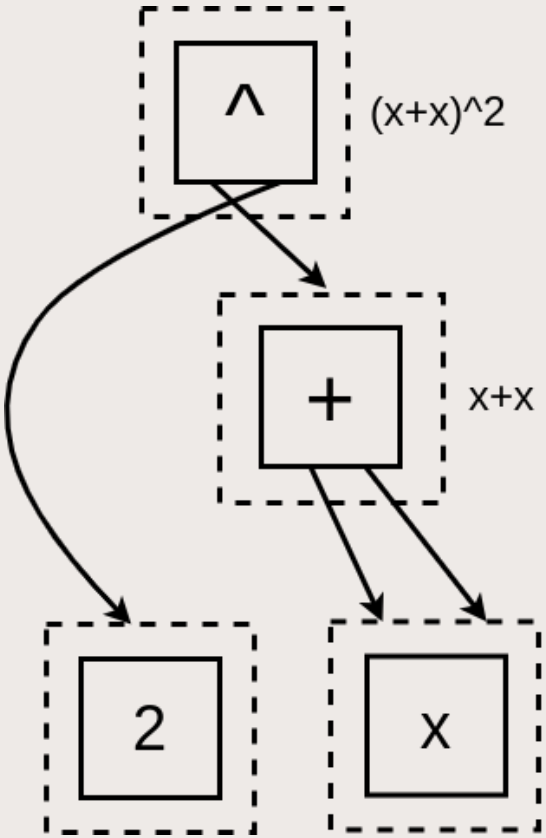



The dashed boxes, my stripes, are called **e**-classes, and they are assigned a numerical **id**.

The solid boxes are called **e**-nodes and are associated to a symbol.

The arrows, called **e**dges, points from one e-node to an e-class, they give the instructions on how to build equivalent expressions. We can read them as “if you go from this e-node, you can pick any e-node from the destination e-class and they will form the same expression (up to equivalence)”.

# Equality Saturation



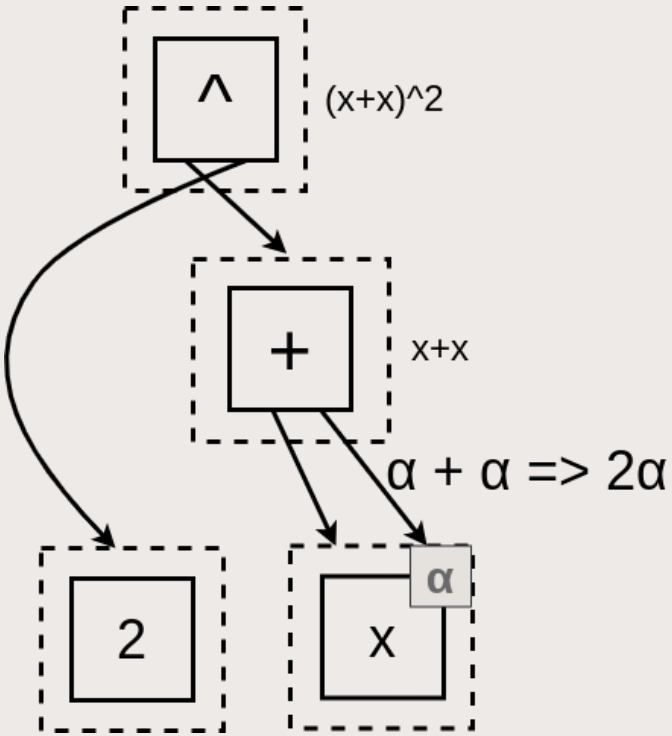

I can start small, resembling an acyclic directed graph.

# Equality Saturation

$$\alpha + \alpha \Rightarrow 2\alpha$$
$$\alpha \times \alpha \Rightarrow \alpha^2$$
$$\alpha \times (\beta + \gamma) \Rightarrow \alpha \times \beta + \alpha \times \gamma$$

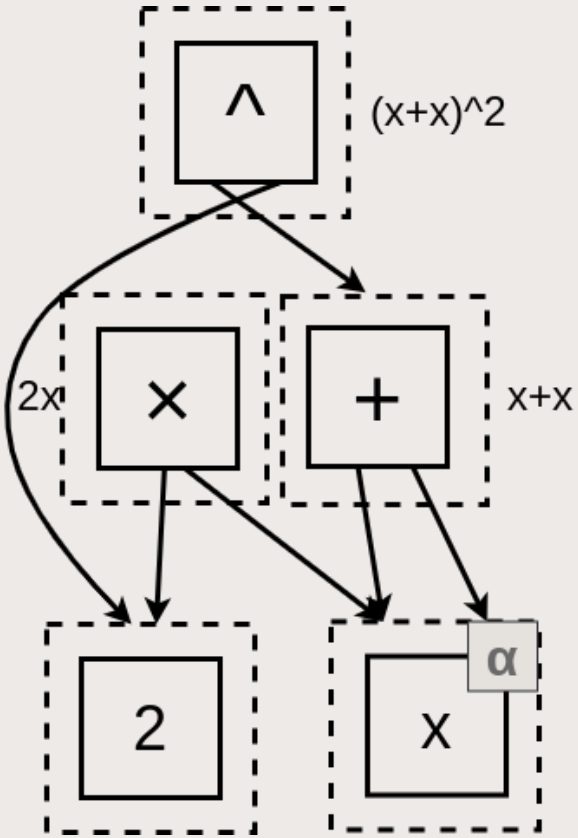
And I can learn some rules of how things are equal!

# Equality Saturation



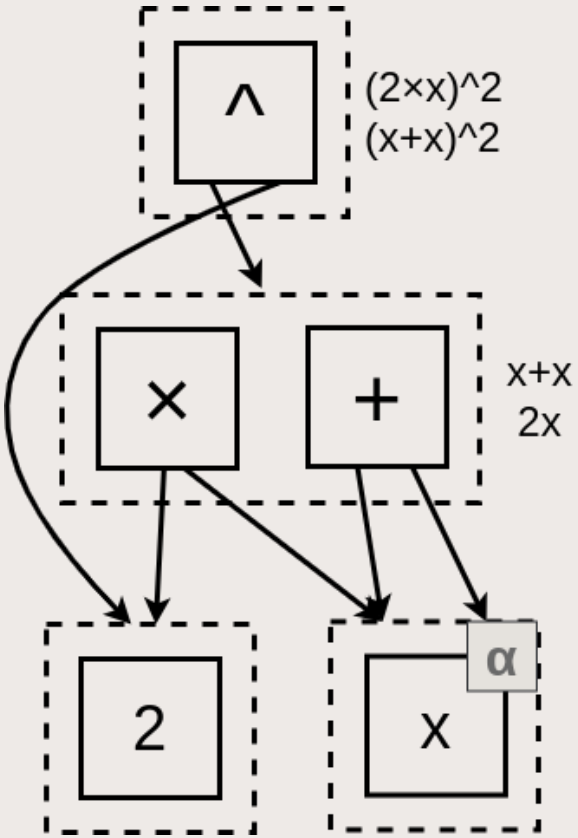

Using these rules, I can find the e-classes that matches the left-hand side of these rules…

# Equality Saturation



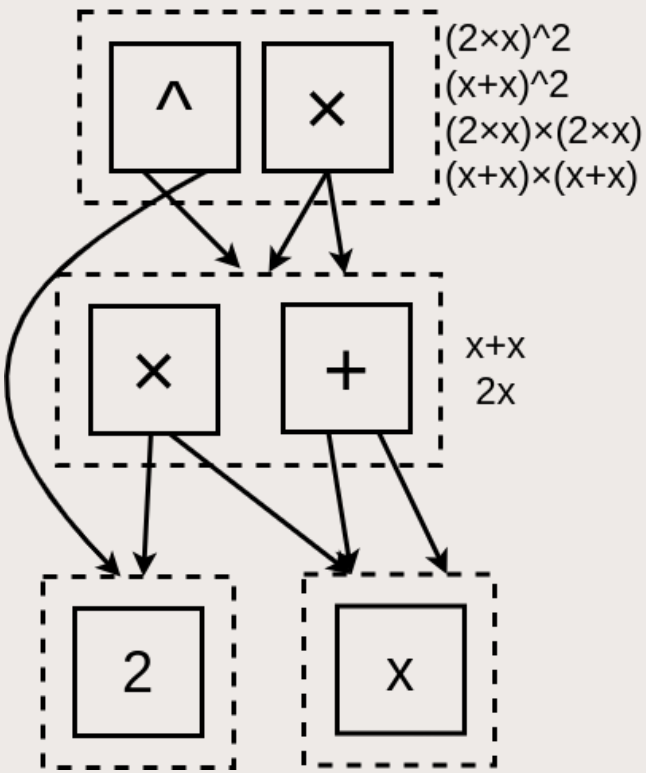

…and grow new e-nodes adding the right-hand side of the rule as a new e-node/e-class.

# Equality Saturation



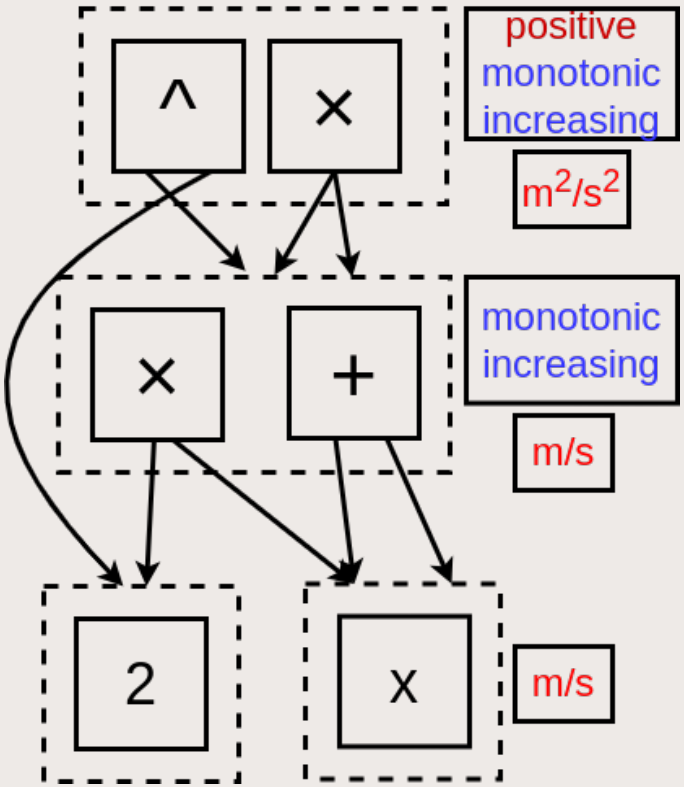

Since we know that they are equivalent, I can merge them into a single e-class.

# Equality Saturation




As I keep applying these rules, I can grow up to a point where no rule will create new e-nodes. At this point, I have reached **saturation** and every possible equivalent expression can be extracted from me!
(but reaching saturation often requires much time and memory)

# Equality Saturation

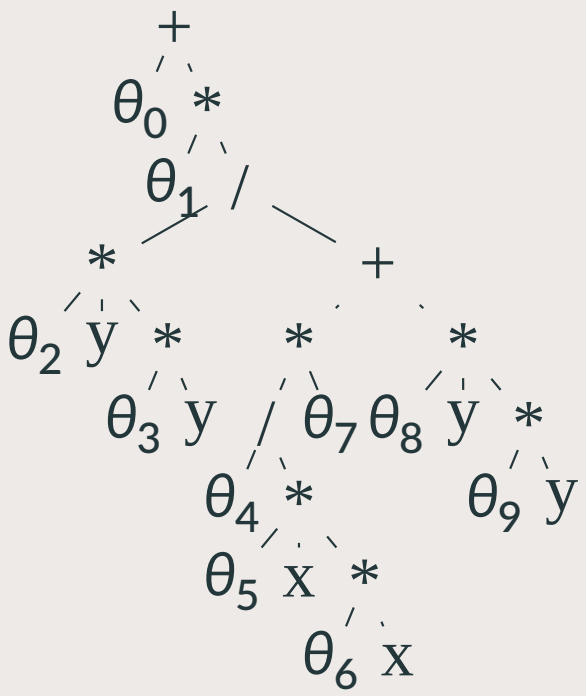



I can also store additional information about every e-class that can be propagated upwards, gaining new knowledge.

This information can be **universal** or **data dependent**.

# Symbolic Regression + E-graphs =

# Simplifying expressions

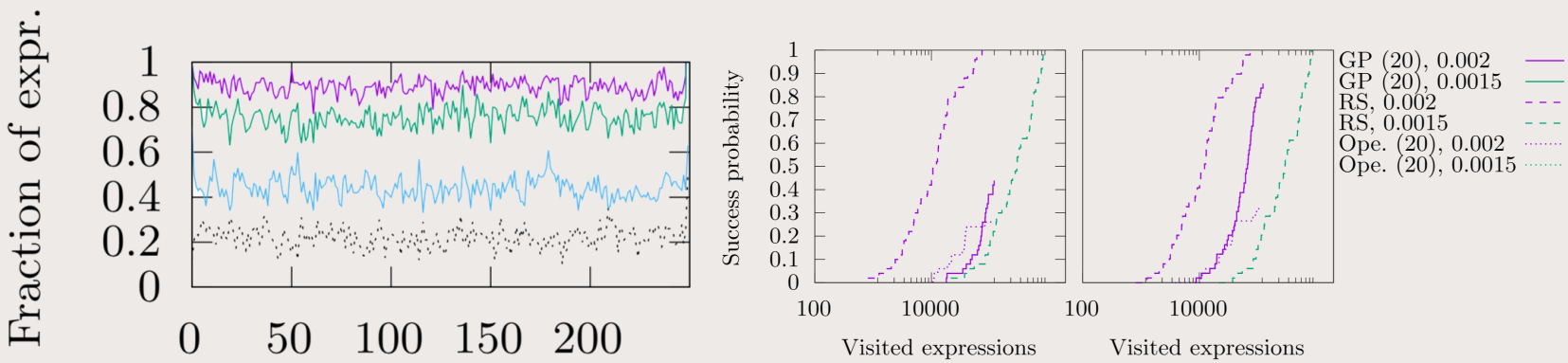



Equality saturation and e-graphs were used to reduce the overparameterization of symbolic expressions, leading to more interepretable expressions and easier to fit.

de Franca, Fabricio Olivetti, and Gabriel Kronberger. ”Reducing overparameterization of symbolic regression models with equality saturation.” Proceedings of the Genetic and Evolutionary Computation Conference. 2023.

Kronberger, Gabriel, and Fabrício Olivetti de França. ”Effects of reducing redundant parameters in parameter optimization for symbolic regression using genetic programming.” Journal of Symbolic Computation 129 (2025): 102413.

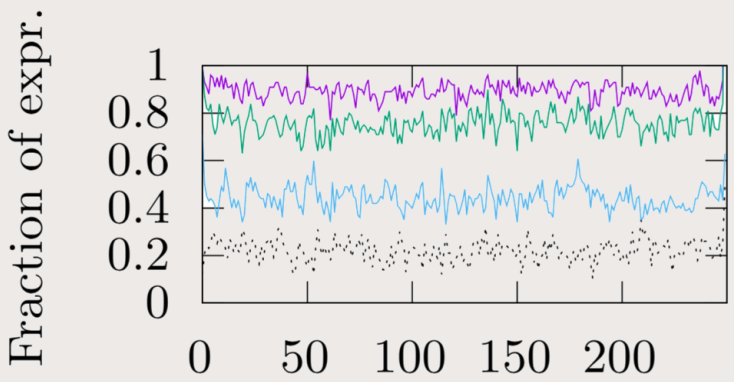
# Measuring inneficiency of genetic programming




Applying equality saturation, we can *normalize* the visited expressions and verify the inneficiency of genetic programming search.

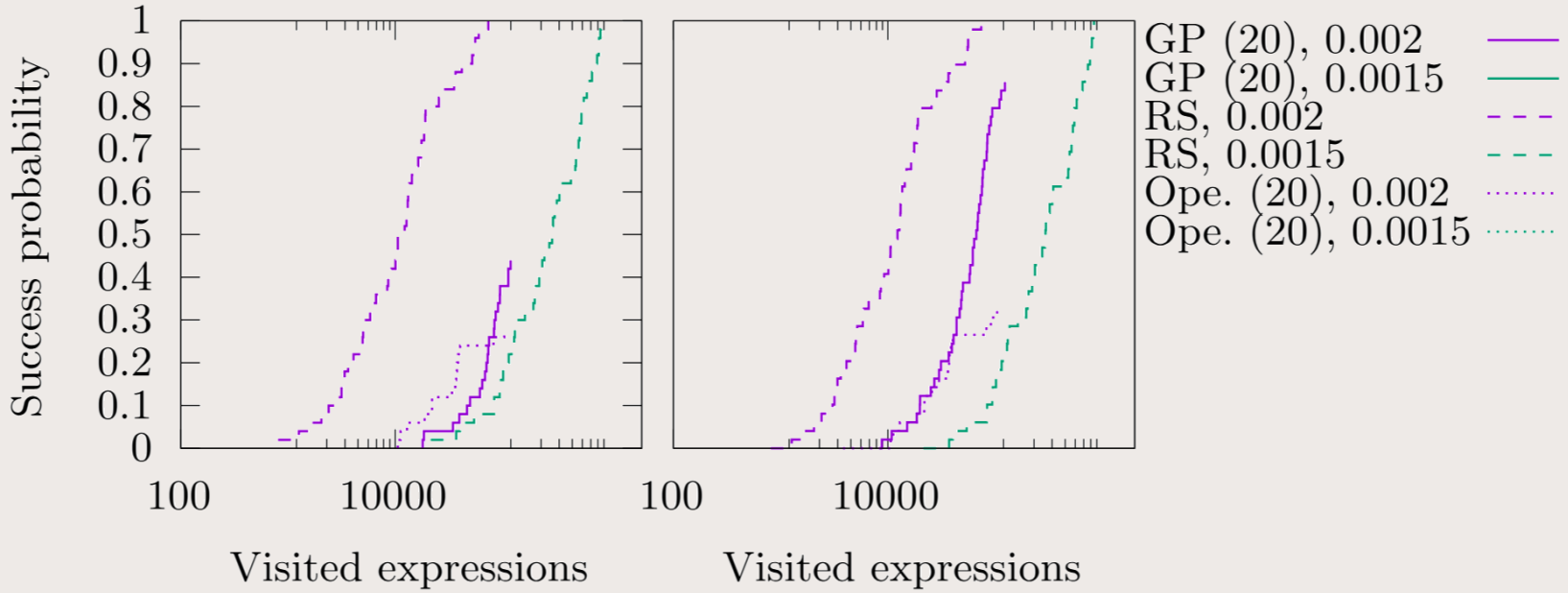

Kronberger, Gabriel, et al. "The Inefficiency of Genetic Programming for Symbolic Regression." International Conference on Parallel Problem Solving from Nature. Cham: Springer Nature Switzerland, 2024.

# Measuring inneficiency of genetic programming



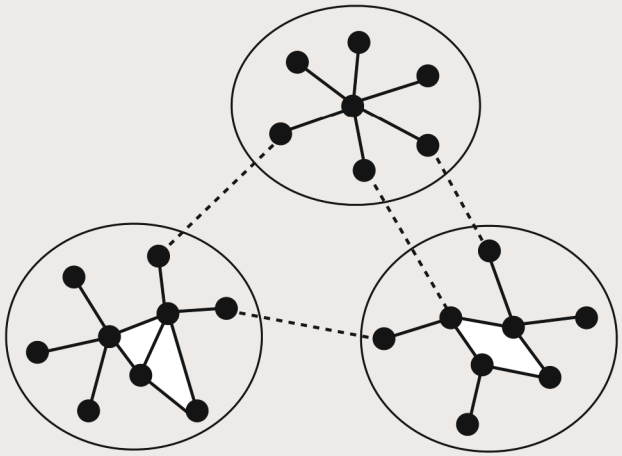

The lines are counts of: - unique expressions visited during GP search (purple) - unique structures (replacing constants with placeholders) (green) - unique structures after simplifying (blue) - dotted line is a trivial expression $c$

# Measuring inneficiency of genetic programming




An ideal **random search** (RS) can find the best solution much faster than GP and Operon. But it depends on ensuring that we can sample from all of the possible expressions **without replacement**.

# Neutrality

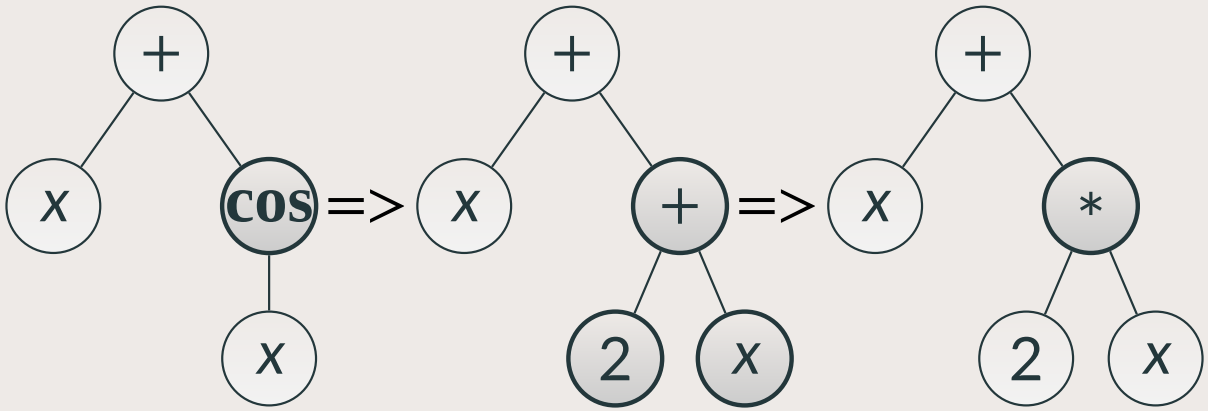


Genetic programming tends to create equivalent expressions to help with the navigability.
When it creates an equivalent and *bloated* expression, it will insert new bulding blocks into the search without impacting the fitness, thus creating opportunity to find different expressions at a later time.

Hu, Ting, and Wolfgang Banzhaf. ”Neutrality, robustness, and evolvability in genetic programming.” Genetic Programming Theory and Practice XIV (2018): 101-117.

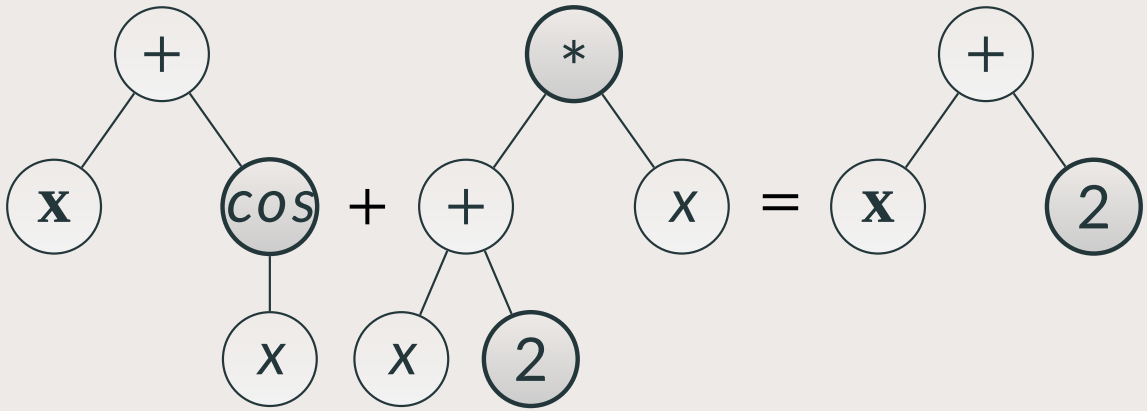
# Generating novelty

What would happen if we enforce the generation of unvisited expressions?




If we store all of the search history inside a single e-graph, we can verify all possible perturbations that generates an unvisited expression.

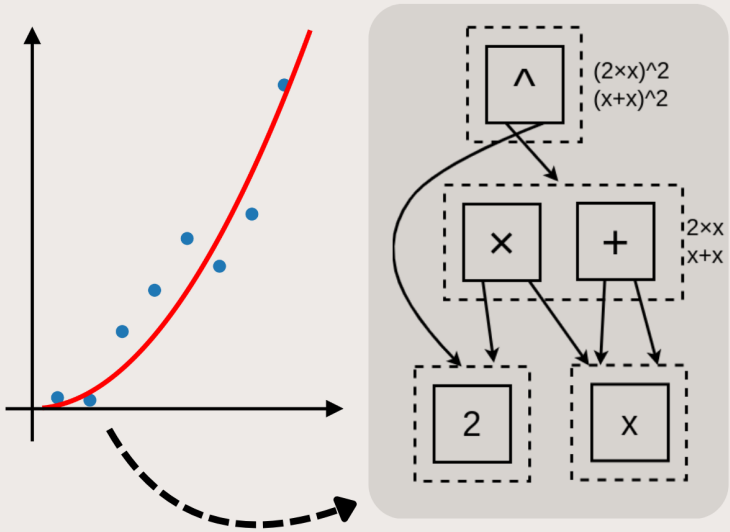
# Generating novelty




We can also verify all recombinations that will generate a new expression.

# Equality Graph Assisted Symbolic Regression

# Equality Graph Assisted Symbolic Regression




**symbolic regression + e-graphs = ❤**

Let us introduce **SymRegg**, an e-graph assisted symbolic regression. Main objectives:

- store all the search history in a single e-graph
- run few iterations of equality saturation to generate equivalent expressions
- exploit the e-graph to enforce the creation of unvisited expressions
- minimize the burden of choosing hyper-parameters

# SymRegg

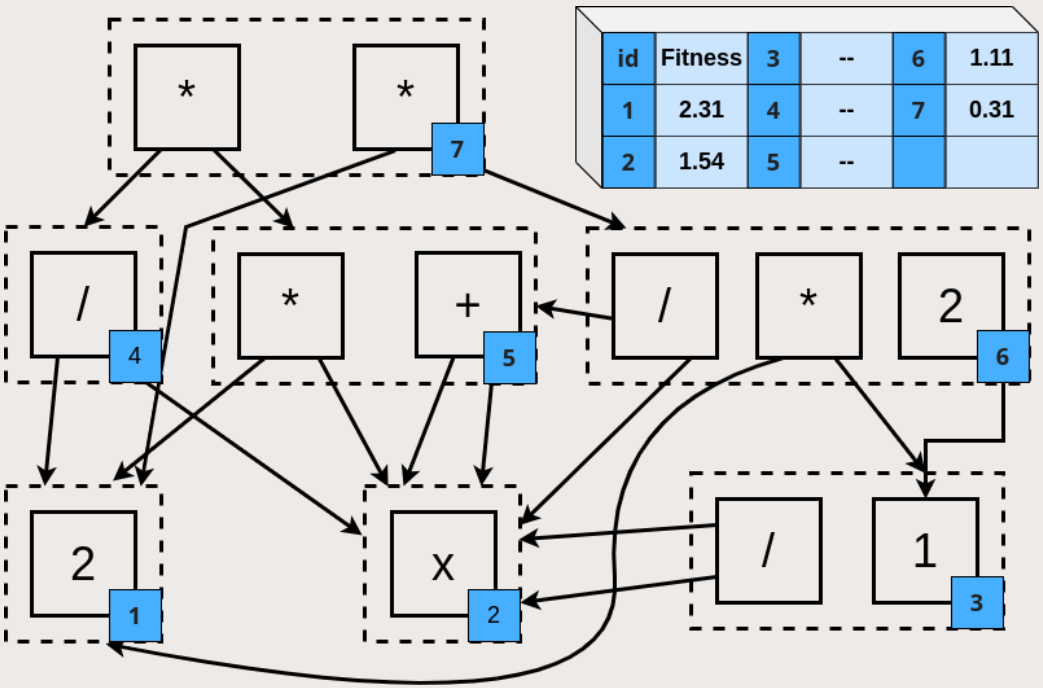
- Insert all terminals into e-graph and evaluate.
- Insert random expression up to a *max_size*.
- Repeat until we generate *n* unique expressions:
  - Try to generate an unvisited expression from the 50 max_size expressions using the 50 first dominance fronts.
  - If it fails, try to generate an unvisited expression from the 100 best expressions found so far.
  - If it fails, try to evaluate a random not-evaluated e-class from the e-graph.
  - If it fails, try to insert a completely random expression.
  - If we succeed to find a new expression, evaluate, insert into the e-graph and run 1 step of equality saturation.
- Return the e-graph

# SymRegg - Generating expressions from a selection

The first two attempts will perform either the e-graph assisted subtree perturbation or the e-graph assisted subtree recombination, each one with 50% chance applied to the pre-selected sets (the 50 first fronts or top-100 expressions).

This will exploit the promising building blocks generated during the search.

# SymRegg - Evaluating a subtree




| id | Fitness | 3 | -- | 6 | 1.11 |
|---|---|---|---|---|---|
| 1 | 2.31 | 4 | -- | 7 | 0.31 |
| 2 | 1.54 | 5 | -- | | |

If the first two steps fail, the algorithm retrieves all e-classes that were never evaluated, and evaluate one at random. This will guarantee the evaluation of a new expression unless all e-classes are evaluated.

# SymRegg - Inserting a new expression

In the worst case scenario, the algorithm will generate a new expression completely at random.

# SymRegg - Final steps

For every generated expression, we can verify whether it was never visited in $\log n$ for $n$ nodes in the expression.

If it already exists, the algorithm will not fit the parameters and evaluate (the computationally expensive part).

# Efficiency of SymRegg

# Datasets

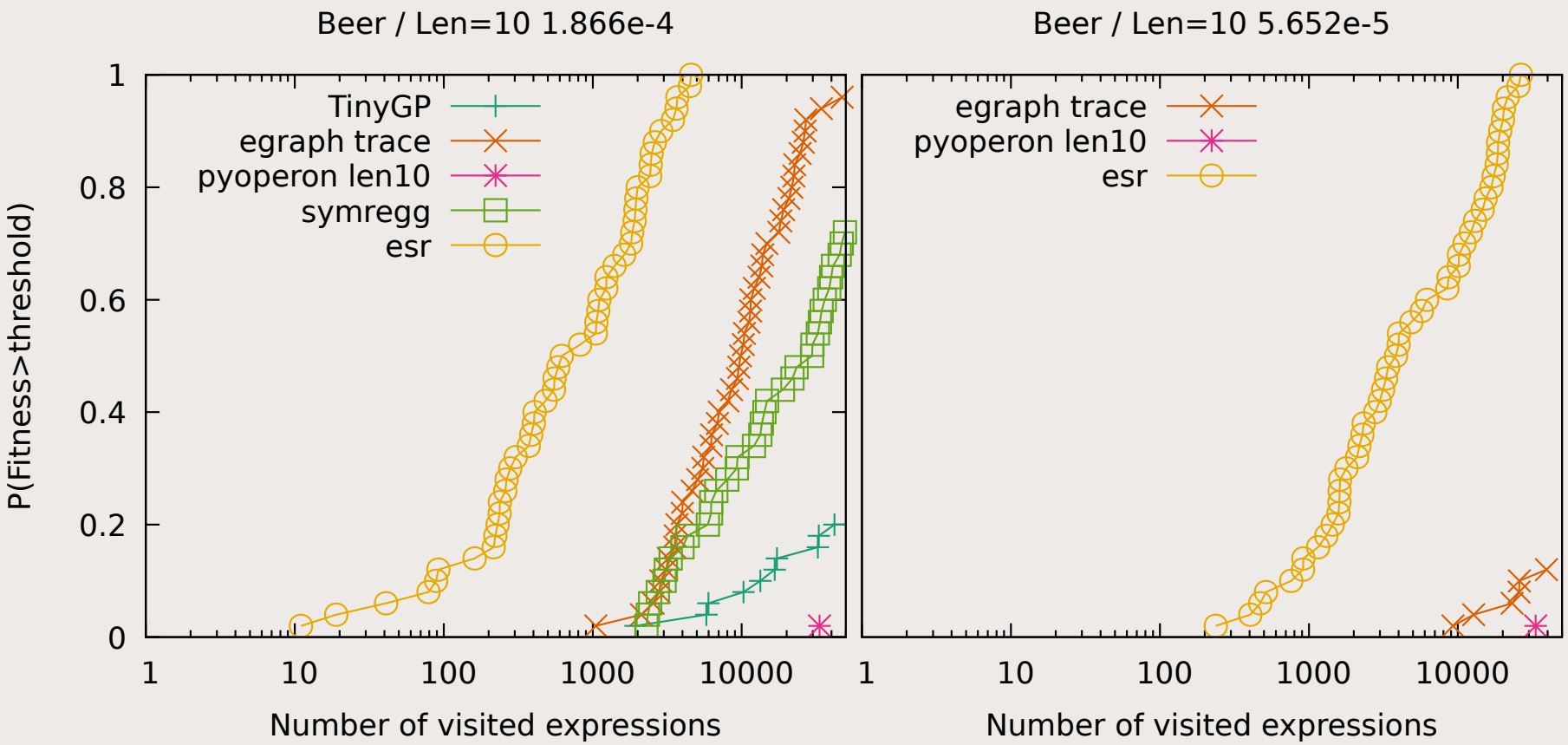
We have tested the efficiency of SymRegg using 4 datasets:

- Beer's law and Supernovae, extracted from[1]
- Nikuradse 1 and 2, extracted from[2] as well as the experimental methods.

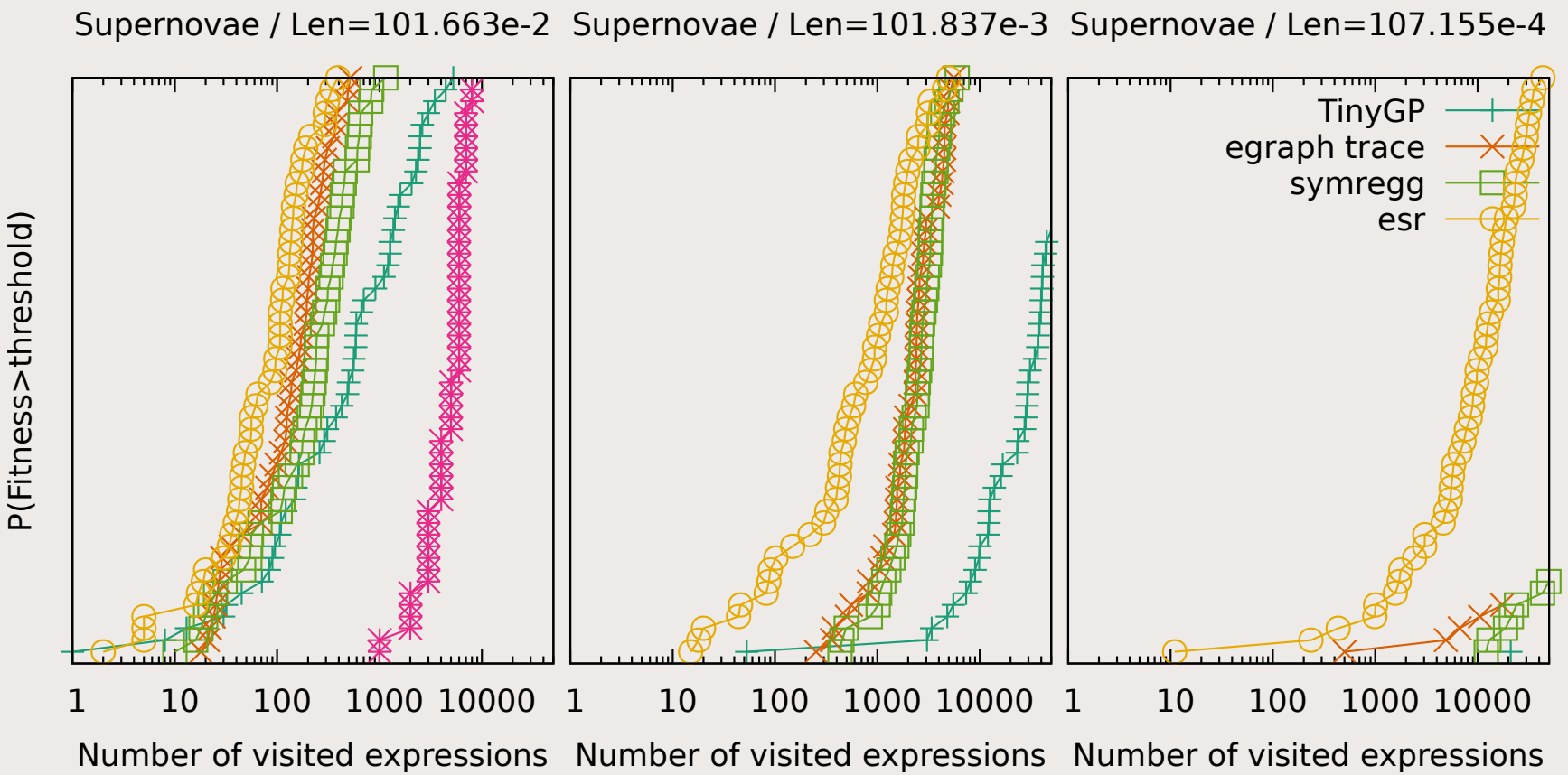
[1] Russeil, Etienne, et al. "Multiview Symbolic Regression." Proceedings of the Genetic and Evolutionary Computation Conference. 2024.

[2] Kronberger, Gabriel, et al. "The Inefficiency of Genetic Programming for Symbolic Regression." International Conference on Parallel Problem Solving from Nature. Cham: Springer Nature Switzerland, 2024.

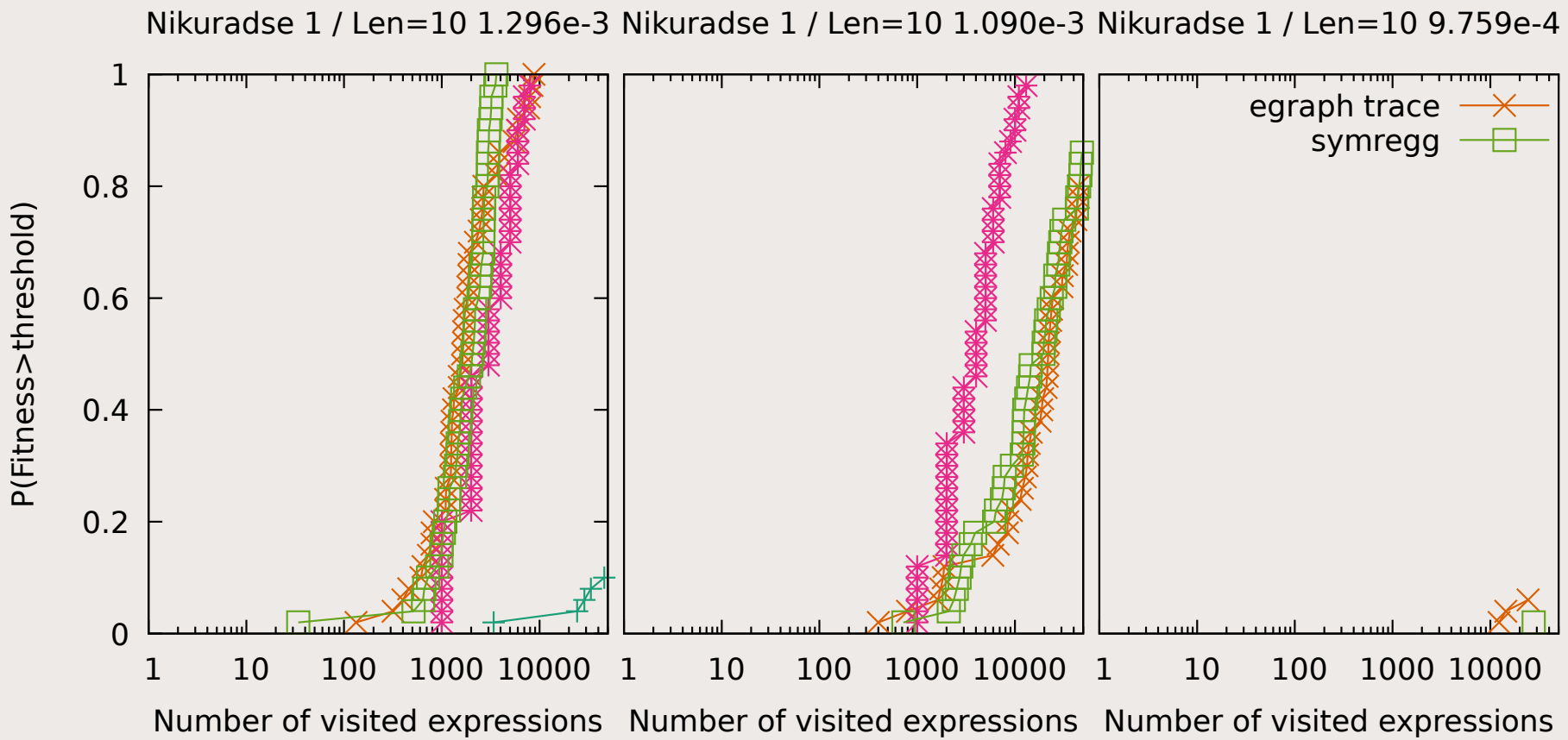
# Beer’s law


Beer / Len=10 1.866e-4
Beer / Len=10 5.652e-5
P(Fitness>threshold)
Number of visited expressions
Number of visited expressions
TinyGP
egraph trace
pyoperon len10
symregg
esr
egraph trace
pyoperon len10
esr
0
0.2
0.4
0.6
0.8
1
1
10
100
1000
10000

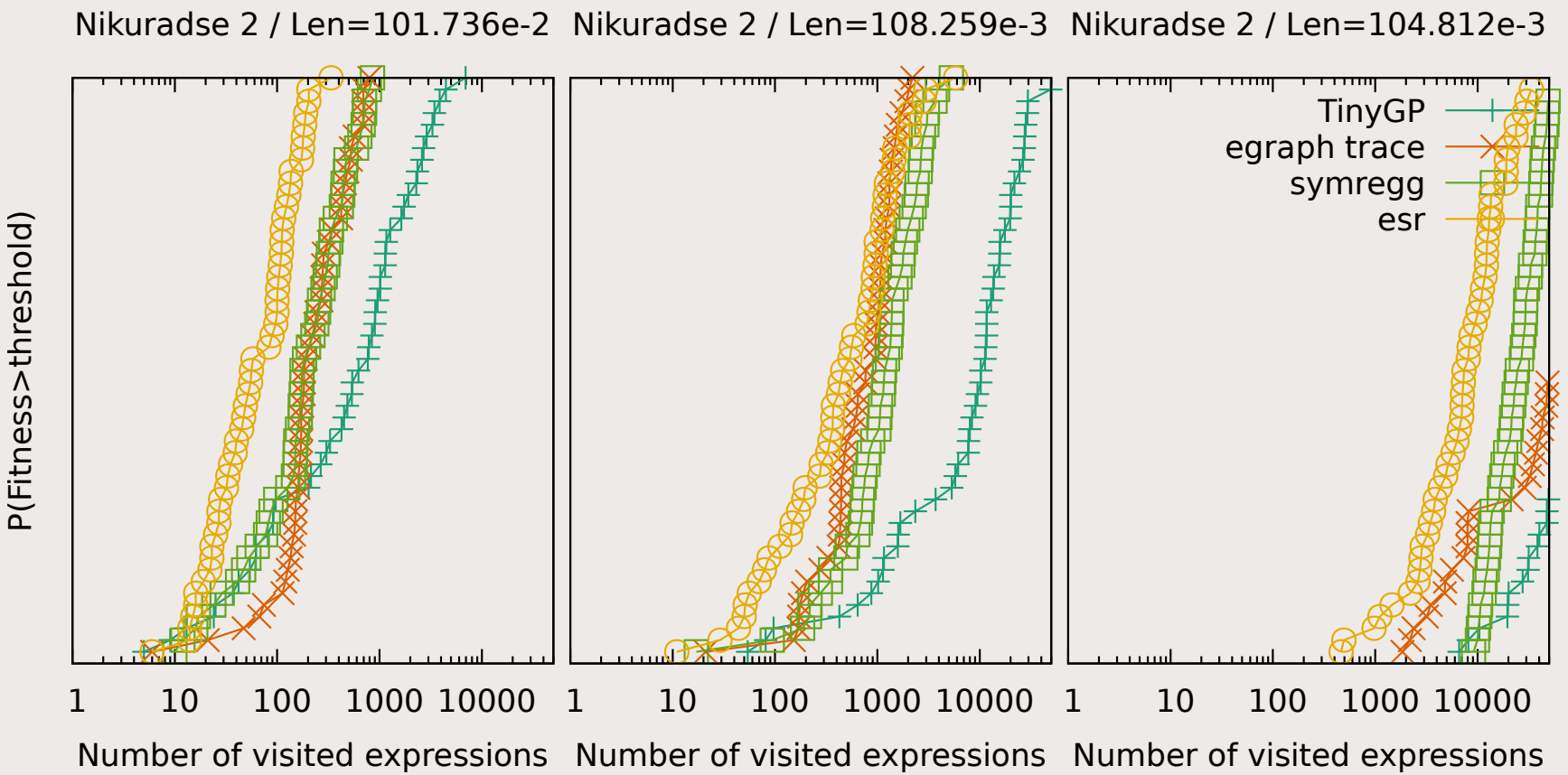
# Supernovae


Supernovae / Len=101.663e-2
Supernovae / Len=101.837e-3
Supernovae / Len=107.155e-4
P(Fitness>threshold)
TinyGP
egraph trace
symregg
esr
1
10
100
1000
10000
Number of visited expressions

# Nikuradse 1


Nikuradse 1 / Len=10 1.296e-3
Nikuradse 1 / Len=10 1.090e-3
Nikuradse 1 / Len=10 9.759e-4
P(Fitness>threshold)
0
0.2
0.4
0.6
0.8
1
1
10
100
1000
10000
Number of visited expressions
egraph trace
symregg

# Nikuradse 2


Nikuradse 2 / Len=101.736e-2
Nikuradse 2 / Len=108.259e-3
Nikuradse 2 / Len=104.812e-3
P(Fitness>threshold)
TinyGP
egraph trace
symregg
esr
1
10
100
1000
10000
Number of visited expressions

# What’s next?

# The future of this marriage

With SymRegg we are only scratching the surface of how e-graphs can assist the process of equation discovery.

There are many more we can do to exploit the information gathered during the search.

# rEGGression

rEGGression: an Interactive and Agnostic Tool for the Exploration of Symbolic Regression Models introduces a new tool for symbolic models that allow the user to go beyond the traditional Pareto front.

# Integrating Domain Knowledge

As shown at the beginning, we can store additional information about the e-classes in the e-graph,

either **universal** such as:

- Monotonically increasing, concave, convex, symmetric, etc.

or **data dependent** such as:

- Units information, totality of the operation (e.g., no division by zero), shape information dependent of the parameter

# Integrating Domain Knowledge

This information can be used to select the best expressions explored during the search that gurantees such properties.

During the search, we can create domain knowledge aware operators that tries to enforce the creation of feasible expressions.

# Incrementally building a large database

The e-graph is nothing more than a representation of a part of the search space. It can be stored in a database, thus avoiding having to rebuild it during a new search.

Every new search can store information about the fitness for that particular dataset into a relational database, and information such as average fitness of a certain building block can be used to generate expressions for a new dataset, creating the possibility of a transfer learning.

# Final Remarks

In short, this presentation introduced the idea of e-graphs and equality saturation and how they can improve the efficiency of equation discovery.

Not only that, but we showed that this data structure can be exploited to advance even further into creating a personalized experience for the user, enabling them to smoothly integrate their domain knowledge into the search or after the search.

# Questions

All the e-graph assisted tools for symbolic regression can be found at: https://github.com/folivetti/srtree