# Explainable AI - Identifying Model Forms

Rohith Ravindranath rr3415, Andrew Tang at3456
*CRIS Lab - Columbia University*
*Github: https://github.com/rohithravin/CRIS_Lab_ExplainableAI*

## Abstract

Black-box models are known to have massive limitations with respect to explainability and interpretability. Such a constraint discourages many industries from using such accurate and powerful methodologies in high-risk challenges. Specifically in the chemical engineering industry where there is high human-risk, it is imperative to understand the data and the model forms derived from that data . We propose a methodology that uses an ontology—a knowledge base on chemical engineering model forms, relationships, and fundamental laws—in tangent with a novel search algorithm that structurally and contextually matches an equation to its corresponding first-principle definition.

## 1. Introduction

Nowadays, machine learning is used in almost every industry. The abundance of data has allowed machine learning and artificial intelligence to prosper. Many of the models that are used identify as black-box models. Black-box models have high accuracy and can obtain a good fit between input and output variables. However, the name "black-box" arises from not being able to quantitatively identify the relationship and mechanisms between the input and output variables. This is visibility apparent in neural networks. These models do amazing in computer vision, natural language processing, and game-playing. Yet when we attempt to understand why the model does so well, all we get are tensors of weights. To the human eye this means nothing. With respect to chemical engineering systems, black-box models are not able to identify and report back the physicochemical mechanisms. Engineering a methodology that can efficiently identify the underlying physicochemical mechanisms of the data  is the current challenge within the industry [1]. This is the premise of our problem, and we hope to tackle an aspect of that problem.

Currently, there are many researchers tackling this problem. One in particular is Professor Venkat Venkatasubramanian, Abhishek Sivaram, and Arijit Chakraborty and their machine learning system called AI-Darwin[2]. AI-Darwin[2] automatically discovers mechanism-based models from data for non-linear parametric systems. In simple terms, their system is able to extract the underlying chemical engineering fundamental principles and laws from the data and output a single equation. This equation describes the data based on the relationships between variables. The system however does not identify what specific model form the equation represents.

Our work this semester focuses on identifying the equations that AI-Darwin[2] generates from the equation. We want to be able to identify the equation to a specific chemical engineering model form, the variables used, and the context of each variable. In order to accomplish this task, we first need to develop a knowledge based on the inner workings of chemical engineering and the domain's fundamental laws. We use an ontology for this purpose. From the ontology, we can search through the known model forms and identify which one most resembles the input equation output form AI-Darwin. Our methodology that we propose utilizes an ontology, constraint satisfaction and search graph traversal to identify the correct model form.

The report is organized as follows. Section 2 describes the XAI approach where we discuss the various parts of our methodology and how they all fit into our search algorithm pipeline. We also discuss certain challenges and decisions we made with respect to the ontology to create a succinct knowledge base. We then report the result of our algorithm on various inputs (equations) in Section 3. We show the strengths and deficiencies in our algorithm that would be addressed next semester. We also show the need for providing context and how important it is to determine which model form identifies closest to the input equation. Finally in Section 4 and 5, we discuss our results and future work.

## 2. Methodology

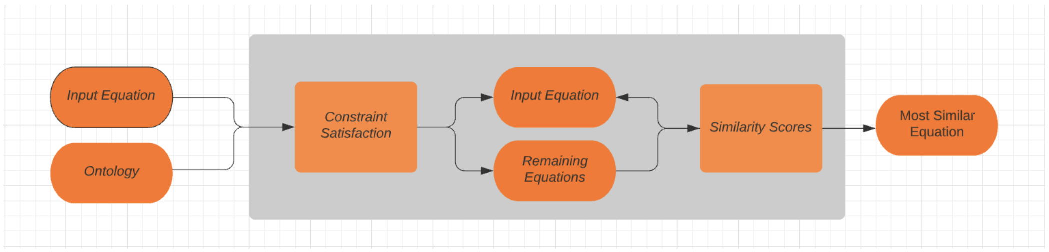
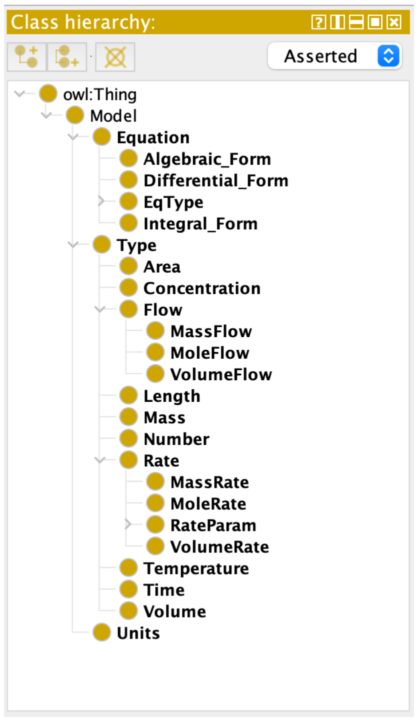


**Fig. 1** Flowchart of XAI Methodology

We propose a methodology that incorporates a knowledge base of chemical model forms. This ontology is then used within a novel algorithm to identify an equation that best represents or is most similar to the input given. The algorithm has two parts. First, we trivally eliminate all equations from the ontology based on the general characteristics of the input equation. Second, with the remaining equations, we compute a similarity score against the input equation. The equation with the highest similar score - indicating the equation most closely resembles the input equation based on variables, context, and mathematical operations - is returned as the final output. We now discuss in detail each part of the methodology.

### 2.1 Ontology

An ontology is a formal norming and definition of categories, properties and relations between the concepts, data, and entities that substantiate the domain. Simply put, an ontology is a way of showing the properties of a subject area and how they are related, by defining a set of concepts and categories that represent the subject. We aim to use an ontology to formally store information about chemical engineering model forms. We structure our ontology in such a way where we can extract meaningful information about each equation, such variables and context.



- owl:Thing
  - Model
    - Equation
      - Algebraic_Form
      - Differential_Form
      - EqType
      - Integral_Form
    - Type
      - Area
      - Concentration
      - Flow
        - MassFlow
        - MoleFlow
        - VolumeFlow
      - Length
      - Mass
      - Number
      - Rate
        - MassRate
        - MoleRate
        - RateParam
        - VolumeRate
      - Temperature
      - Time
      - Volume
    - Units

**Fig. 2** Hierarchical Representation of XAI Ontology

There are three aspects to every model - the equation itself, the variables, and the units associated with each variable and the equation. From this knowledge, we create three main classes in our ontology - Equation, Type, and Units. The Equation class stores information about the form of the equation. There are three forms - Algebraic, Integral, and Differential. In most cases, these equations are disjoint from each other. However, there are equations such as Equation 1.4, where it falls under both derivative and integral. The type class are different categories of variables and parameters that are present in an instance of an equation. Notice that there may be subclasses within a type of variable, such as Flow and Rate, to allow for better representation of the variable within the equation. The Units class contains various instances of different units that are used within the variables and equations. This class is especially important as it allows us to explain what the respective equation is doing with respect to its variables. Figure 2 outlines the current hierarchy of our ontology. We create various instances of the classes and subclasses, all independent of each other. These are then connected via Object Properties to identify which variables are associated with equations and with what relationship.

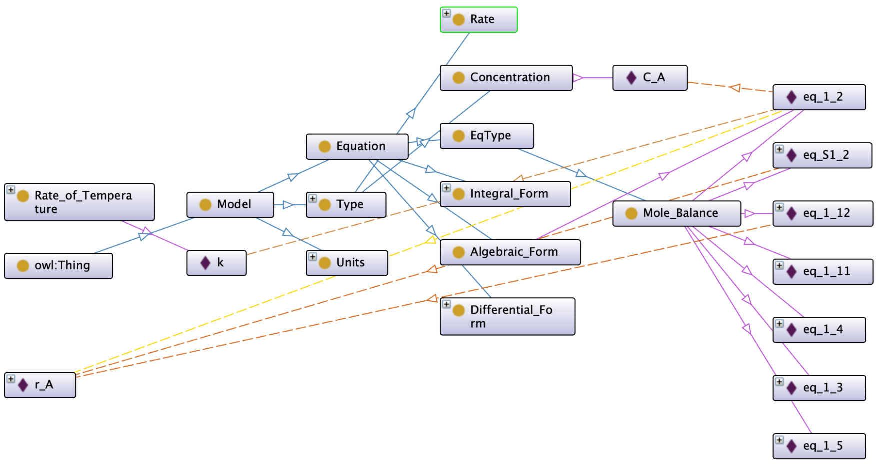
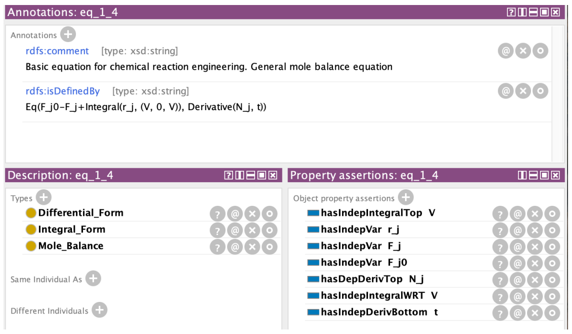


**Fig. 3** Snapshot of relationships in XAI Ontology

One challenge we had in our ontology was how do we explicitly state the mathematical relationship of each variable with respect to the associate equation. In the beginning we thought of creating variable instances where we specify the mathematical relationship as part of the name. However, this was not ideal since we ended up with duplicate variables. This made our ontology not succinct conceptually. We decided on defining various Object Projecties based on mathematical relationships, which would then then be used to connect variables to equations. This proved to be the best path forwarded, as we only had to instantiate the variable once  and could be used with different context in different equations based on the Object Property used in that equation. Figure 3 shows a graph representation of how the equations and variables are connected - specifically Eq_1_2.


Annotations: eq_1_4

Annotations

rdfs:comment [type: xsd:string]
Basic equation for chemical reaction engineering. General mole balance equation

rdfs:isDefinedBy [type: xsd:string]
Eq(F_j0−F_j+Integral(r_j, (V, 0, V)), Derivative(N_j, t))

Description: eq_1_4

Types
Differential_Form
Integral_Form
Mole_Balance

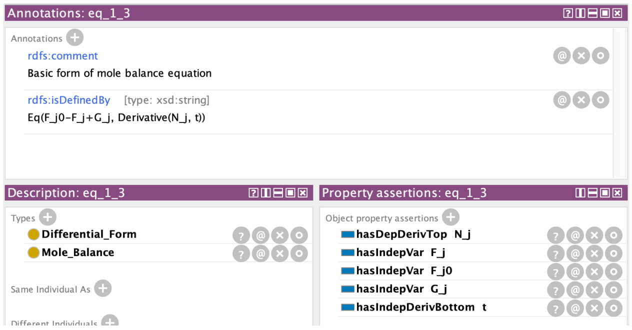
Same Individual As

Different Individuals

Property assertions: eq_1_4

Object property assertions
hasIndepIntegralTop V
hasIndepVar r_j
hasIndepVar F_j
hasIndepVar F_j0
hasDepDerivTop N_j
hasIndepIntegralWRT V
hasIndepDerivBottom t

**Fig 4.** Eq_1_4 from XAI Ontology


Annotations: eq_1_3

Annotations

rdfs:comment
Basic form of mole balance equation

rdfs:isDefinedBy [type: xsd:string]
Eq(F_j0−F_j+G_j, Derivative(N_j, t))

Description: eq_1_3

Types
Differential_Form
Mole_Balance

Same Individual As

Different Individuals

Property assertions: eq_1_3

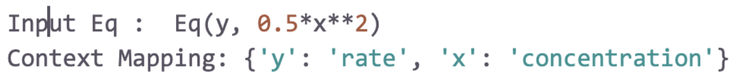
Object property assertions
hasDepDerivTop N_j
hasIndepVar F_j
hasIndepVar F_j0
hasIndepVar G_j
hasIndepDerivBottom t

**Fig 5.** Eq_1_3 from XAI Ontology

Figure 4 and 5 provide clear examples of how our variables, equations and Object Properties are used to define an equation and all the mathematical relationships within it. Notice how variable F_j0 is in both equations but is used differently in each equation. The respective Object Property nicely outlines how the variable is used differently in each equation. We can even go further and click on each variable and understand what is its context - flow, rate, concentration etc. This succinct and expansive information for a given equation will be useful when identifying similar equations for a given input equation. Also note the naming convention of our equation instances. Eq_1_3 represents the model form Equation-1-3 in the Fogler[3] textbook. This naming convention makes it easier for engineers to cross-reference information.

Given our ontology, we now have constructed a knowledge base that has well defined equations and variables. With each equation, we can identify the respective variables used, their context, and the mathematical relationships with respect to the equation. With this structured information, we can now begin to develop an algorithm that efficiently and accurately identifies the most similar equation given an input equation.

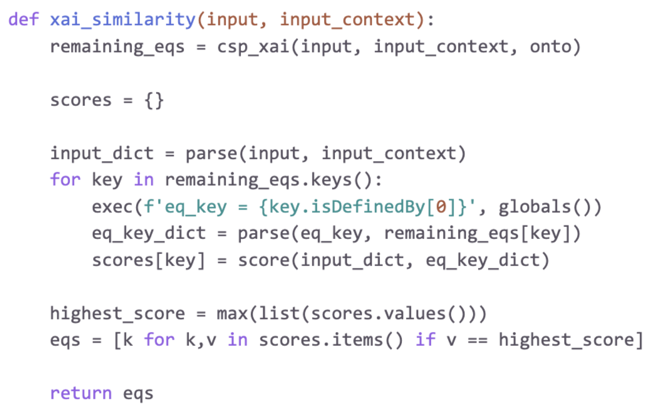
## 2.2 Algorithm



```
Input Eq :  Eq(y, 0.5*x**2)
Context Mapping: {'y': 'rate', 'x': 'concentration'}
```

**Fig 6. Sample Input Into XAI Algorithm**

Our algorithm has two parts. First, the constraint satisfaction task. Second is similarity scoring. The input to the algorithm is an equality equation and the context mapping of the input equation. Context mapping contains a list of variables present in the equation and their respective context.



```
def xai_similarity(input, input_context):
    remaining_eqs = csp_xai(input, input_context, onto)

    scores = {}

    input_dict = parse(input, input_context)
    for key in remaining_eqs.keys():
        exec(f'eq_key = {key.isDefinedBy[0]}', globals())
        eq_key_dict = parse(eq_key, remaining_eqs[key])
        scores[key] = score(input_dict, eq_key_dict)

    highest_score = max(list(scores.values()))
    eqs = [k for k,v in scores.items() if v == highest_score]

    return eqs
```

**Fig 7. XAI Algorithm**

With these two inputs, we then are able to trivally eliminate equations from our ontology based on equation form and context. From the subsetted equations, we compute the similarity score against the input equation and output the equation with the highest score; indicating that the input equation is most similar to the our selected equation.

### 2.2.1 Constraint Satisfaction

The first step in our algorithm is to trivally eliminate all equations that don't resemble our input equations. These characteristics include form of the equation (algebraic, derivative, integral, derivative & integral) and context matching.

With respect to the form of the equation, we eliminate possible equations from our ontology based on the type of equation it is. We identify whether the input equation is of algebraic, derivative, integral, derivative & integral by traversing down the graph representation of the equation. At each layer we search if there is a derivative or integral operation. If there is a derivative operation, we classify the equation as a differential form. If there is an integral operation, we classify the equation as an integral form. If there is both a derivative and an integral form, we classify the equation as an integral and differential form. If it is neither, then we classify the equation as an algebraic form. From our equation bank, we then subset only the equations that have the same classification as our input equation.

```
if not is_eq_derivative(input_eq):
    remove differential eqs from bank
if not is_eq_integral(input_eq):
    remove integral eqs from bank
if not is_eq_algebraic(input_eq):
    remove algebraic eqs from bank
```

**Fig 8. Pseudocode of removing eqs based on math form.**

Next, from our subsetted equation bank we further eliminate possible equations based on the context of the equation. Since we are also given the context of each variable in our equation, we can use that to identify the similar equations that have the same equation. We do this by computing the set difference between input equation context and every context specific to each possible equation.

$$input\_eq\_context\_set \; - \; eq\_context\_set$$

**Eq. 9.** Set difference between input equation context and arbitrary equation context

This set difference makes a certain assumption. One is that eq_context_set is a superset of input_eq_context. Hence, we are trying to see if every context in the input equation is present in the current equation context. If the output is 0, then we know that that current equation includes all the context present in the input equation. If it is greater than 1, then the contexts don't match, and we eliminate our current equation from our possible equations.

From the constraint satisfaction task, at least 50% of the equations from our bank (ontology) is eliminated, usually more. These are possible equations, and their context mapping are then scored against the input equation based on a similarity metric, which is discussed below.

### 2.2.2 Equation Similarity Metric

This part of the algorithm takes in two inputs, both of which are equations with a context mapping for each variable. Equations are represented as trees. Each internal node is a mathematical operator, and each leaf node is a symbol or number. We then parse each equation tree as a hashmap of tree levels, in which each key is the depth of a level, and its corresponding value is the set of nodes on that level. We represent each node as a tuple of its type with its contexts (if it has any).

Next, we compute the similarity metric between the two equations. This metric is computed via a level-by-level comparison of two equation trees (A and B). We only compare up to and including the smallest of the maximum level (depth) of either tree. For each level of trees A and B, we find the intersection of the set of nodes on that level between the two trees. We take the length of this intersection set, and add it to the similarity metric (which starts at zero).

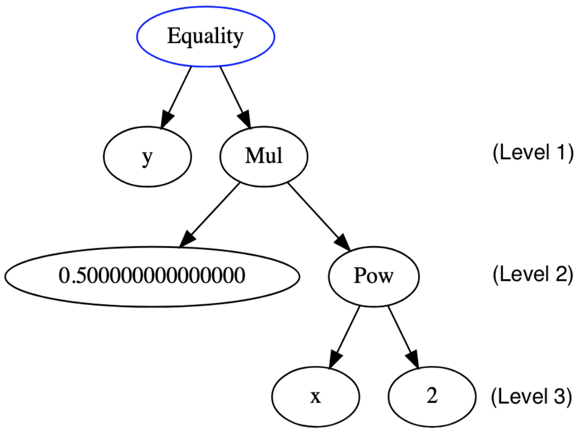


**Fig. 10.** Tree graph representation of $y = 0.5x^2$

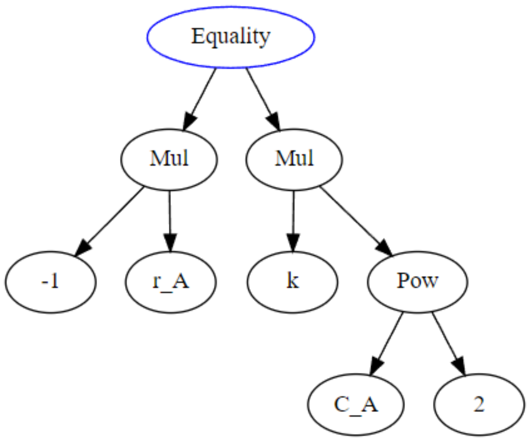


**Fig. 11.** Tree graph representation of $-r_A = kC_A^2$

| Key | Value |
|---|---|
| 1 | {(symbol, rate), (Mul,)} |
| 2 | {(float,), (Pow,)} |
| 3 | {(symbol, concentration), (integer,)} |

**Fig. 12.** Hashmap of Fig. 4. with *y* as rate and *x* as concentration.

| Key | Value |
|---|---|
| 1 | {(Mul), (Mul,)} |
| 2 | {(integer,), (symbol, rate), (symbol, rate), (Pow,)} |
| 3 | {(symbol, concentration), (integer,)} |

**Fig. 13.** Hashmap of Fig. 5. with $r_A$ and $k$ as rate, and $C_A$ as concentration.

Upon computing the similarity metric between Fig. 4. and Fig. 5, we would receive 4. Level one has one equivalent node of type "Mul", level two has one equivalent node of type "Pow", and level three has two equivalent nodes—one of type "symbol" with context "concentration", and the other of type "integer."

## 3. Results

### 3.1 Accuracy

We test the accuracy of our algorithm by inputting equations, with different symbols, that are already present in our ontology.

| Input Equation | Input Contexts | Output Equation(s) |
|---|---|---|
| $-y = 0.5x^2$ | $y$ −> rate<br>$x$ −> concentration | $-r_A = kC_A^2$ |
| $-a + a_0 + \int_0^x b\,dx = \frac{d}{dt}y$ | $y$ −> number<br>$x$ −> volume<br>$t$ −> time<br>$a, a_0$ −> flow | $-F_j + F_{j0} + \int_0^V r_j dV = \frac{d}{dt}N_j$ |

| | | |
|---|---|---|
| $\frac{d}{dx}y = z$ | $y$ −> flow<br>$x$ −> volume<br>$z$ −> rate | $\frac{d}{dV}F_j = r_j$<br>$\frac{d}{dV}F_A = r_A$ |
| $\frac{d}{dj}y = z \times x$ | $y$ −> number<br>$x$ −> volume<br>$z$ −> rate<br>$j$ −> time | $\frac{d}{dt}N_j = r_j \times V$ |
| $-a + a_0 + z \times x = \frac{d}{dj}y$ | $y$ −> number<br>$x$ −> volume<br>$z$ −> rate<br>$j$ −> time<br>$a, a_0$ −> flow | $-F_A + F_{A0} + r_A \times V = \frac{d}{dt}N_A$ |

**Fig. 14.** Experimental results for inputs that are represented by our ontology

These results show that our algorithm can accurately predict the first-principles definition of an input equation, given that one of its model forms is present in our ontology and that each of its variables have a context mapping. Of note here is that we show the outputs as mathematical expressions for clarity purposes. In the implementation of the algorithm, the outputs are ontology objects that correspond to these expressions. From each object it is possible to query its type (the class it belongs to), its relationships with the variables and parameters, and, perhaps more importantly, query an explanation for the equation itself. For instance, the "annotation" for the second output equation in Fig. 14. is "Basic equation for chemical reaction engineering. General mole balance equation."

## 3.2 Missing or No Input Contexts

| **Input Equation** | **Input Contexts** | **Output Equation(s)** |
|---|---|---|
| $-y = 0.5x^2$ | None | $-r_A = kC_A^2$ |
| $-a + a_0 + \int_0^x b\,dx = \frac{d}{dt}y$ | None | $-F_j + F_{j0} + \int_0^V r_j dV = \frac{d}{dt}N_j$ |
| $\frac{d}{dx}y = z$ | $x$ −> volume<br>$y$ −> flow | $\frac{d}{dV}F_j = r_j$<br>$\frac{d}{dV}F_A = r_A$<br>$-F_A + F_{A0} + r_A \times V = \frac{d}{dt}N_A$ |
| $\frac{d}{dx}y = z$ | $y$ −> flow | $\frac{d}{dV}F_j = r_j$<br>$\frac{d}{dV}F_A = r_A$<br>$\frac{d}{dW}F_A = r'_A$<br>$-F_A + F_{A0} + r_A \times V = \frac{d}{dt}N_A$<br>$-F_j + F_{j0} + G_j = \frac{d}{dt}N_j$ |

| | | |
|---|---|---|
| $\frac{d}{dj}y = z \times x$ | None | $\frac{d}{dt}N_j = r_j \times V$ |

**Fig. 15.** Output equations given missing or no input contexts.

From Fig. 15., we can see that even without context, the first, second and last equations can output the correct equation. However, the third and fourth input equations output different results depending on the contexts it is missing. The third input outputs an extra equation, and the fourth input outputs three extra equations.

## 3.3. Inputs Not in Ontology

Now, we test with inputs that are not directly represented in our ontology.

| **Input Equation** | **Input Contexts** | **Output Equation(s)*** |
|---|---|---|
| $\frac{d}{dt}x = 0.9x^2 - 0.005xy + 0.05x$ | $x, y$ −> concentration<br>$t$ −> time | $-F_j + F_{j0} + G_j = \frac{d}{dt}N_j$<br>$-F_j + F_{j0} + \int_0^V r_j dV = \frac{d}{dt}N_j$<br>$-F_A + F_{A0} + r_A \times V = \frac{d}{dt}N_A$ |
| $\frac{d}{dt}C_A =- 0.01C_AC_B + 0.5C_B$ | $C_A$, $C_B$ −> concentration<br>$t$ −> time | Same as above |

**Fig. 16.** Output equations when inputs are not in ontology
* These outputs are calculated without using the constraint satisfaction algorithm described in section 2.2.1. We take all equations in the ontology with the highest equation similarity metric to the input equation.

The right-hand sides of both input equations are linear combinations of first order and second order rate laws. The left-hand sides are the derivative of the concentration with respect to time, which is in fact the rate. As such, the correct output should be $-r_A = kC_A^2$, since it is the only rate law represented in our ontology. In other words, from a first-principles perspective, the rate law is the most similar to the inputs and their given contexts. However, the outputs here are not as desired.

## 4. Discussion

For both sections 3.1 and 3.2, our results are as expected. Given an input equation with sufficient context, our method accurately predicts its first-principles chemical equation if it is in our ontology. Of note in Fig. 14 is the third equation, whose output contains two equations. This output is as expected; both equations are differential forms of steady state mole balance on a PFR[3], but with different species. Since the contexts are the same, it follows that the input and the two output equations have the exact same hashmap representation, thereby giving them the highest possible similarity metric.

Interestingly, many of the equations in Fig. 15. seem to predict the correct output despite having no contexts given. There are two explanations for this: either our ontology is yet to be sufficiently populated, or there do not exist other first-principles equations mathematically similar to the input (which is highly unlikely). In other words, the mathematical structure (i.e. mathematical relationships between variables) of the input equation is sufficiently distinctive and unique to single out the corresponding equation in our ontology. This is logically consistent with our equation similarity metric (section 2.2.2), because similar equations, in particular equations with the same mathematical meaning but with different symbols for variables, have similar if not the same tree graph representation. This is because tree representations of equations implicitly capture the mathematical structure of an equation. As such, their similarity metric would be the highest value possible.

When there do exist mathematically similar equations in our ontology, however, contexts for variables become crucial in predicting the correct output. By "correct," we mean outputs that are logically consistent from a first-principles perspective. We can see the importance of contexts from row three of Fig. 15. Upon removing the context map "$z$ —> rate", an extra (wrong) equation appears in the output. Such behavior is expected. Without defining the context of $z$, the constraint satisfaction algorithm does not eliminate equations that do not contain a variable with the rate context. This introduces a lot of randomness into the prediction due to the nature of our similarity metric. An instance of such randomness is shown by Fig. 17., in which we highlight the equivalent nodes. Nodes *y*, *F_A*, and *F_A0* are equivalent because they all have context "flow", and nodes *x*, *V*, *r_A* are equivalent because they all have context "volume". Notice, however, that nodes *F_A0* and *V* of the extra output are not children and grandchildren, respectively, of the *Derivative* node. Our algorithm only checks whether nodes are on the same level, and does not take into account the parent-child relationships in a tree graph. Since any level in an equation tree can theoretically contain any node with any context, it is thus necessary to limit the search space via contextual constraints.

In the case of Fig. 17., if the input had the mapping "$z$ —> rate", even if the extra output were not eliminated by constraint satisfaction, the score of the correct output would be one higher than it is here. This is because *z* and *r_A* would be equivalent nodes as they both have context "rate".

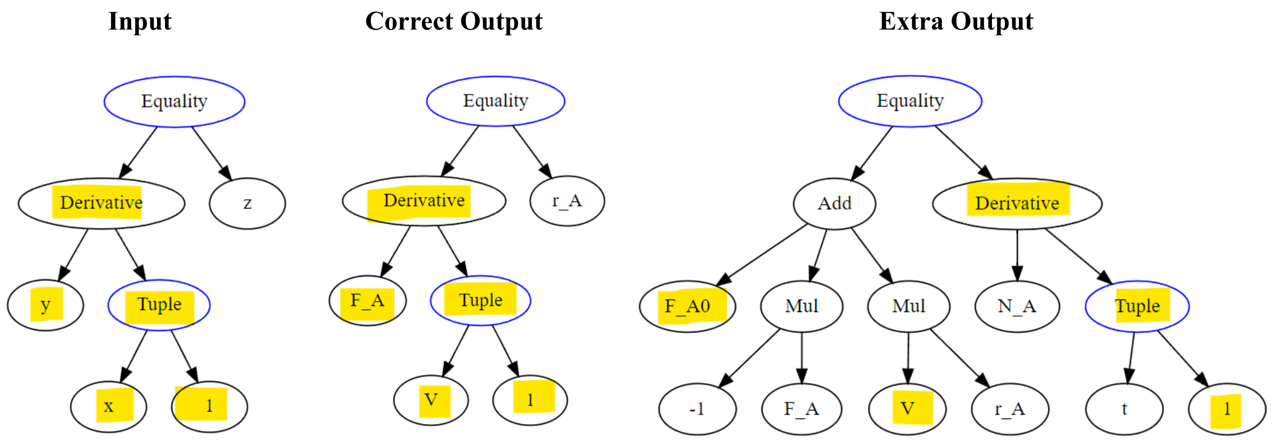


**Fig. 17.** Graph comparison of row three of Fig. 15. with equivalent nodes highlighted. Similarity metric is 5.

Row four of Fig. 15.'s output is also expected. We can do an analysis similar to what we did for Fig. 17.. The first three outputs are intuitive because they are also the derivative of a flow variable with respect to some variable, equal to another variable. Since $z$ and $x$ have no input context, we can treat all variables other than $y$ as arbitrary symbols. It can thus be said that these first three outputs are mathematically consistent with the input equation—they have the same form. Similarly, the last two outputs are included because certain nodes in the tree representation of the equations happen to be on the same level, despite having different parent-child relationships than the input nodes. In particular, the similarity metric is 4 because we simply decrement the node with flow context (compared to a metric of 5 in Fig. 17.).

Section 3.3 attempts to predict inputs that are more complicated mathematical permutations of equations present in our ontology. Unsurprisingly, our results are not ideal. This is because our algorithm is not designed for a combination of first-principles equations. It has also yet to generalize to all variations of a mathematical expression, such as the equivalence of $\frac{d}{dt}C_A = r_A$. The outputs were computed purely using the scoring function without constraint satisfaction because otherwise, no constraints would be satisfied and the output would be empty. Of interest, however, is that the form of the outputs are rather similar to the input. One side is a derivative, and the other is a linear combination of terms. This suggests that our algorithm places more emphasis on how terms relate to each other and which term goes where, rather than the relationships of variables inside a term. This seems to be consistent with the design of our algorithm since it does not account for parent-child relationships between nodes of an equation tree. Intuitively, such relationships capture how each specific variable mathematically relates to another, a heuristic which our algorithm does not consider.

## 5. Conclusion

This project successfully accomplishes our first step towards explanation generation of AI model forms. Our method is able to accurately predict the first-principles representation of an input equation given that it is in our ontology and that its variables are sufficiently contextualized. Our experimental results also offer valuable insights into our constraint satisfaction and similarity metric algorithms, in particular the behavior of the algorithm with missing contexts and its tendency to weigh the form of an equation over individual variable relationships when predicting.

Future work would extend our method to generating explanations for equations that are combinations of multiple first-principles equations, such as the inputs proposed in section 3.3.. The goal is to tag parts of AI generated model forms with first-principles definitions, thereby providing insight on the relationship between such model forms and first-principles knowledge. Our current method also has much room for improvement. The first-principles knowledge base of the ontology can be greatly expanded, and new heuristics can be added to the prediction algorithm.

## 6. References

[1] Venkatasubramanian, V. (2019). The promise of artificial intelligence in chemical engineering: Is it here, finally. AIChE J, 65(2), 466-478.

[2] Chakraborty, A., Sivaram, A., & Venkatasubramanian, V. (2021). Ai-Darwin: A first principles-based model Discovery Engine using machine learning. *Computers & Chemical Engineering*, *154*, 107470. https://doi.org/10.1016/j.compchemeng.2021.107470

[3] Fogler, H. S. (2018). *Essentials of Chemical Reaction Engineering*. Prentice Hall.