scientific reports

OPEN

Simple algorithm for judging equivalence of differential-algebraic equation systems

Shota Kato✉, Chunpu Zhang & Manabu Kano

Mathematical formulas play a prominent role in science, technology, engineering, and mathematics (STEM) documents; understanding STEM documents usually requires knowing the difference between equation groups containing multiple equations. When two equation groups can be transformed into the same form, we call the equation groups equivalent. Existing tools cannot judge the equivalence of two equation groups; thus, we develop an algorithm to judge such an equivalence using a computer algebra system. The proposed algorithm first eliminates variables appearing only in either equation group. It then checks the equivalence of the equations one by one: the equations with identical algebraic solutions for the same variable are judged equivalent. If each equation in one equation group is equivalent to an equation in the other, the equation groups are judged equivalent; otherwise, non-equivalent. We generated 50 pairs of equation groups for evaluation. The proposed method accurately judged the equivalence of all pairs. This method is expected to facilitate comprehension of a large amount of mathematical information in STEM documents. Furthermore, this is a necessary step for machines to understand equations, including process models.

The volume of scientific literature has been increasing exponentially, and this trend continues with an average doubling period of 15 years[1] and is expected to continue. When writing a report on a particular topic, such as a review of previous studies, it is necessary to survey the increasing amount of literature. The key to understanding multiple documents and organizing the information is to recognize the difference among the documents, which requires much toil. Automatically judging the equivalence of the information would be helpful for efficiently processing a large number of documents.

Equations representing the relationships between variables play a central role in understanding documents of science, technology, engineering, and mathematics (STEM). Multiple equations are often used as a single entity to describe the relationship between variables. It is, therefore, crucial to recognize the difference between equation groups consisting of two or more equations when understanding STEM documents. A physical model is a typical example of an equation group. For example, suppose a researcher wants to build a physical model. Before building the model, the researcher surveys previous studies and identifies the differences among the multiple models in the previous studies, which is an arduous task. Several studies have dealt with a text in the chemical engineering field using natural language processing techniques[2–4], but no studies have aimed to reduce this kind of effort.

Since a variable is sometimes expressed by different symbols among documents, we have to extract variable definitions[5–7] and unify the variable symbols' representations[8] before judging the equivalence of equation groups. Such a method can be developed independently of the equivalence judgment; thus, we assume that different symbols do not represent the same variable in this study.

In order to judge the equivalence of equation groups, computers must grasp the meanings of mathematical formulas. Converting natural language into vectors is one of the methods for computers to handle the meanings of natural language, and recent studies utilize neural network models, such as Word2Vec[9], Transformer[10], and Bidirectional Encoder Representations from Transformers (BERT)[11]. Similarly, several studies represent mathematical formulas with neural network models[12–14]. Mansouri et al.[12] defined a similarity between two formulas based on their appearances, but similar-looking formulas do not necessarily perform the same calculation. For example, the similarity between $a + b = 0$ and $a - b = 0$ is higher than that between $a + b = 0$ and $a = -b$

Department of Systems Science, Kyoto University, Yoshida-honmachi, Sakyo-ku, Kyoto 606-8501, Japan. ✉email: shota@human.sys.i.kyoto-u.ac.jp

based on the models by Mansouri et al.[12]. The existing neural network models, which focus on the appearance of formulas, do not work for the equivalence judgment of equation groups.

Another approach for handling the meanings of mathematical formulas in computers is encoding the formulas with special markup languages such as Content Mathematical Markup Language (MathML)[15] and OMDoc[16]. However, such markup is rarely used to publish mathematical knowledge[17]. The commonly used methods for notating mathematical expressions are LaTeX for papers and Presentation MathML[15] for the Web.

The most effective way for computers to comprehend formulas' meanings is to use computer algebra systems (CASs). Formula transformations, for example, from LaTeX to the format usable in CASs, are well-studied in the CASs literature[18,19]. Further, CASs, such as Mathematica and Maple, have LaTeX input support.

CASs can solve equations and judge the equivalence of two equations by comparing their solutions for one variable. Similarly, if two equation groups are solvable for one variable, CASs can judge their equivalence by comparing their solutions. However, when one of two equation groups to be compared is not solvable for one variable, CASs alone cannot correctly judge their equivalence. Physical models are commonly represented by combinations of differential equations and algebraic equations, called differential-algebraic equation (DAE) systems. Therefore, it is essential to address this issue for machines to compare equation groups contained in documents related to chemical engineering.

In this study, we propose a method for solving this problem. The proposed method uses a computer algebra system to eliminate variables contained only in either equation group and judge whether each equation in one equation group is equivalent to an equation in the other. We generate 50 equivalent and non-equivalent pairs of equation groups and evaluate the performance of our proposed method.

Methods

Equivalence judgment methods.

Figure 1 schematically presents our proposed method for judging the equivalence of two equation groups. The algorithm 1) eliminates variables, 2) checks whether two equation groups have the same set of variables, and 3) judges the equivalence between each equation in one equation group and each equation in the other equation group. If two equation groups share the same set of variables after variable elimination and each equation in one equation group is equivalent to an equation in the other, the equation groups are judged equivalent; otherwise, they are judged non-equivalent.

Our algorithm utilizes a CAS to (1) solve equations, (2) substitute formulas into variables to eliminate the variables, and (3) judge whether two formulas are equivalent.

In this study, we assume that two equivalent equations are solvable for any variable, and each number of solutions for the variable is one, that is, the solution for the variable is unique.

Equivalence judgment of equations. Two equivalent equations have to satisfy the following requirements:

- The equations have the same set of variables.
- The solutions of the equations for any variable are the same.

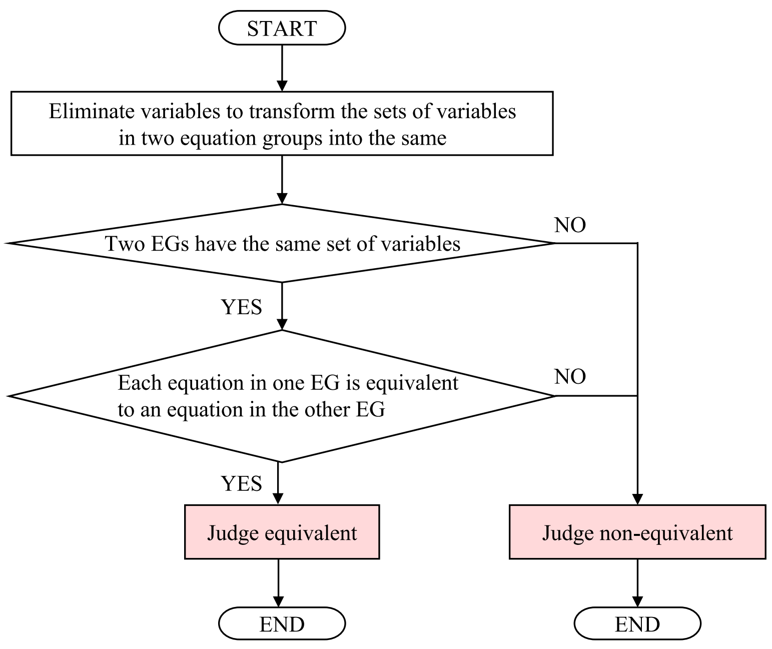
START
Eliminate variables to transform the sets of variables in two equation groups into the same
Two EGs have the same set of variables
NO
YES
Each equation in one EG is equivalent to an equation in the other EG
NO
YES
Judge equivalent
Judge non-equivalent
END
END

Figure 1. A schematic illustration of the algorithm for judging the equivalence of two equation groups. EG represents an equation group.

Algorithm 1 Equation equivalence judgment

```
Input: Two equations, $e_\mathrm{A}$ and $e_\mathrm{B}$, and their sets of variables, $V_{e_\mathrm{A}}$ and $V_{e_\mathrm{B}}$
Output: Judgment result
if $V_{e_\mathrm{A}} \neq V_{e_\mathrm{B}}$ then
    return false
else
    for $i = 1$ to $|V_{e_\mathrm{A}}|$ do
        $v_{e_\mathrm{A},i} \leftarrow i$th variable in $V_{e_\mathrm{A}}$
        if $e_\mathrm{A}$ and $e_\mathrm{B}$ are solvable for $v_{e_\mathrm{A},i}$ and each number of solutions for $v_{e_\mathrm{A},i}$ is one then
            $s_\mathrm{A} \leftarrow$ solution of $e_\mathrm{A}$ for $v_{e_\mathrm{A},i}$
            $s_\mathrm{B} \leftarrow$ solution of $e_\mathrm{B}$ for $v_{e_\mathrm{A},i}$
            if $s_\mathrm{A} = s_\mathrm{B}$ then
                return true
            else
                return false
            end if
        end if
    end for
end if
return false
```

We propose Algorithm 1 to sequentially check the above points and judge the equivalence of two equations, e_A and e_B. In Algorithm 1, the sets of variables of e_A and e_B, V_{e_A} and V_{e_B}, are compared at first (Lines 1 and 2). The algorithm then checks whether e_A and e_B are solvable for any variable, and each number of solutions for the variable is one (Line 6). Here, 'solvable' means that a variable can be explicitly expressed by the other variables. If the solutions of e_A and e_B, s_A and s_B, are the same, the two equations are judged equivalent (Lines 7–10). The judgment was conducted by converting s_A and s_B to simple forms by conversion rules implemented in CAS, such as Sympy's simplify[20] and Mathematica's Simplify[21], and determining if the obtained two formulas are the same. The equations are judged non-equivalent if any variable's solutions are different (Lines 11 and 12), either equation cannot be solved, or the number of solutions is more than one (Line 17).

Variable elimination in equation group. Before judging the equivalence of two equation groups, we eliminate variables in each equation group so that the equation groups have the same set of variables. For example, equation groups $\{x + y = 0\}$ and $\{x = t, y = -t\}$ are easily judged to be equivalent when the variable t is removed from the latter equation group. A variable is eliminated by solving an equation for the variables and substituting its solution to the other equations with the variable as shown in Algorithm 2.

Algorithm 2 Variable elimination in an equation group

```
Input: An equation group $E$ and a variable to be eliminated $v$
Output: $E^*$
$E_v \leftarrow$ equations including $v$ in $E$
for $i = 1$ to $|E_v|$ do
    $e_i \leftarrow i$th equation in $E_v$
    if $e_i$ is solvable for $v$ and has only one solution for $v$ then
        $s \leftarrow$ solution of $e_i$ for $v$
        $E'_v \leftarrow E_v \setminus \{e_i\}$
        eliminate $v$ in $E'_v$ by substituting $s$ into $v$
        $E^* \leftarrow (E \setminus E_v) \cup E'_v$
        return $E^*$
    end if
end for
```

To eliminate a variable v in an equation group E, the algorithm first obtains the equations including v in E, E_v (Line 1). If the ith equation in E_v, e_i, is solvable for v and has only one solution for v, the solution of e_i for v, s, is computed (Lines 3–5). Then, s is substituted into v in all equations in E_v except e_i, and the set of the substituted equations E'_v is obtained (Lines 6 and 7). Finally, the set of equations E^* that does not include v is derived by replacing E_v in E with E'_v (Line 8).

Equivalence judgment of equation groups. We judge not only the equivalence between two equations but also the equivalence between two equation groups consisting of multiple equations. Equivalent two equation groups after variable elimination need to satisfy the following two conditions:

- The two equation groups share the same set of variables.
- Each equation in one equation group is equivalent to an equation in the other equation group.

Based on these conditions, we propose Algorithm 3 for equivalence judgment of two equation groups E_{A} and E_{B}, whose sets of variables are $V_{E_{\mathrm{A}}}$ and $V_{E_{\mathrm{B}}}$, respectively.

Algorithm 3 Equation group equivalence judgment

```
Input: Two equation groups, $E_A$ and $E_B$, and their sets of variables, $V_{E_A}$ and $V_{E_B}$
Output: Judgment result
1: $V_{bs} \leftarrow V_{E_A} \cap V_{E_B}$
2: $V_{ws,E_A} \leftarrow$ set of within-shared variables of $E_A$
3: $V_{ws,E_B} \leftarrow$ set of within-shared variables of $E_B$
4: $V^*_{E_A} \leftarrow V_{ws,E_A} \setminus V_{bs}$
5: $V^*_{E_B} \leftarrow V_{ws,E_B} \setminus V_{bs}$
6: if $(V_{E_A} \setminus V_{E_B}) \neq V^*_{E_A}$ or $(V_{E_B} \setminus V_{E_A}) \neq V^*_{E_B}$ then
7:     return false
8: else
9:     while $V_{E_A} \neq V_{E_B}$ do
10:        if $V^*_{E_A} \neq \varnothing$ then
11:            $v^*_{E_A} \leftarrow$ one of the variables in $V^*_{E_A}$
12:            update $E_A$ by eliminating $v^*_{E_A}$ following Algorithm 2
13:        else if $V^*_{E_B} \neq \varnothing$ then
14:            $v^*_{E_B} \leftarrow$ one of the variables in $V^*_{E_B}$
15:            update $E_B$ by eliminating $v^*_{E_B}$ following Algorithm 2
16:        end if
17:        update $V_{E_A}$, $V_{E_B}$, $V_{bs}$, $V_{ws,E_A}$, $V_{ws,E_B}$, $V^*_{E_A}$, and $V^*_{E_B}$
18:    end while
19: end if
20: if $\forall e_A \in E_A, \exists e_B \in E_B, e_A = e_B$ and $|E_A| = |E_B|$ then
21:     return true
22: else
23:     return false
24: end if
```

At first, the algorithm compares $V_{E_{\mathrm{A}}}$ and $V_{E_{\mathrm{B}}}$ to transform them into the same (Lines 1–19). Here, we define between-shared variables V_{bs} as the variables shared between E_{A} and E_{B}, and within-shared variables $V_{\mathrm{ws,E_i}}$ as the variables shared between the equations within an equation group E_i (Lines 1–3). $V_{E_{\mathrm{A}}}$ and $V_{E_{\mathrm{B}}}$ are transformed into the same by eliminating the variables appearing only in either equation group. Besides, variables that can be eliminated in an equation group are included in its within-shared variables. Hence, the variables to be eliminated in E_{A} and E_{B}, which are denoted by $V^*_{E_{\mathrm{A}}}$ and $V^*_{E_{\mathrm{B}}}$, are the set difference of $V_{\mathrm{ws,E_A}}$ and V_{bs} and that of $V_{\mathrm{ws,E_B}}$ and V_{bs}, respectively (Lines 4 and 5). When the variables appearing only in either equation group are not equal to the variables to be eliminated, $V_{E_{\mathrm{A}}}$ and $V_{E_{\mathrm{B}}}$ cannot be transformed into the same. In such a case, E_{A} and E_{B} are judged non-equivalent (Lines 6 and 7). Otherwise, the variables in $V^*_{E_{\mathrm{A}}}$ and $V^*_{E_{\mathrm{B}}}$ are eliminated one by one until the two equation groups share the same set of variables (Lines 8–18).

After the variable elimination, the algorithm checks whether each equation in E_{A} is equivalent to an equation in E_{B} and each number of the equations is the same. If all the equations in E_{A} and E_{B} have a one-to-one relationship, the two equation groups are judged to be equivalent; otherwise, non-equivalent (Lines 20–24).

Figure 2 shows an example of two equivalent equation groups and their within-group shared variables and between-group shared variables. Our proposed algorithm eliminates the variable k to transform the sets of the variables in these equation groups into the same, checks whether each equation in E_{A} is equivalent to an equation in E_{B}, and judges they are equivalent.

Experimental settings. We created 50 equivalent and non-equivalent pairs of equations and equation groups based on physical models in the textbook about process control[22] to evaluate the proposed method. Tables 1 and 2 present 7 cases, and the entire cases can be found as Supplementary Information. The equations used in the experiments are DAEs consisting of four arithmetic operations, elementary functions, and derivatives.

We implemented our proposed algorithm in Python using a Python-based CAS, Sympy[23]. We prepared TeX-formatted equation groups and parsed them using the 'parse_latex' function. Although the 'parse_latex' function is incomplete, we confirmed that all equations were converted correctly to Sympy expressions.

Results and discussion

The proposed method correctly judged the equivalence of all 50 pairs. This section describes how the proposed algorithm realized the correct judgment in each case.

Cases of equation equivalence judgment. In Case 1, the sets of variables of the two equations were different; thus, Algorithm 1 returned false (Lines 1 and 2).

$$E_{\mathrm{A}} \quad \begin{aligned} -r_{\mathrm{A}} &= k_0 \exp\left(-E/(RT)\right) C_{\mathrm{A}} \\ V\frac{\mathrm{d}C_{\mathrm{A}}}{\mathrm{d}t} &= q(C_0 - C_{\mathrm{A}}) + r_{\mathrm{A}}V \\ V\rho C\frac{\mathrm{d}T}{\mathrm{d}t} &= wC(T_{\mathrm{i}} - T) + H_{\mathrm{r}}Vr_{\mathrm{A}} + UA(T_{\mathrm{c}} - T) \end{aligned} \qquad V_{\mathrm{ws},E_{\mathrm{A}}} = \{r_{\mathrm{A}}, T, C_{\mathrm{A}}, V, t\}$$

$$E_{\mathrm{B}} \quad \begin{aligned} -r_{\mathrm{A}} &= kC_{\mathrm{A}} \\ k &= k_0 \exp\left(-E/(RT)\right) \\ V\frac{\mathrm{d}C_{\mathrm{A}}}{\mathrm{d}t} &= q(C_0 - C_{\mathrm{A}}) + r_{\mathrm{A}}V \\ V\rho C\frac{\mathrm{d}T}{\mathrm{d}t} &= wC(T_{\mathrm{i}} - T) + H_{\mathrm{r}}Vr_{\mathrm{A}} + UA(T_{\mathrm{c}} - T) \end{aligned} \qquad V_{\mathrm{ws},E_{\mathrm{B}}} = \{r_{\mathrm{A}}, k, T, C_{\mathrm{A}}, V, t\}$$

$$\begin{aligned} V_{\mathrm{bs}} = \{&r_{\mathrm{A}}, k_0, E, R, T, C_{\mathrm{A}}, V, t, q, C_0, \\ &\rho, C, w, T_{\mathrm{i}}, H_{\mathrm{r}}, U, T_{\mathrm{c}}\} \end{aligned}$$

Figure 2. An example two equivalent equation groups E_{A} and E_{B}. $V_{\mathrm{ws,E_A}}$ and $V_{\mathrm{ws,E_B}}$ are the within-shared variables of E_{A} and E_{B}, and V_{bs} is the between-shared variables of the two equation groups.

Case	Equation		Equivalent
1	E_{A}	$\rho\frac{\mathrm{dV}}{\mathrm{dt}} = w_1 + w_2 - w$	No
	E_{B}	$\rho\frac{\mathrm{dVx}}{\mathrm{dt}} = w_1x_1 + w_2x_2 - wx$	
2	E_{A}	$\rho\frac{\mathrm{dV}}{\mathrm{dt}} = w_1 + w_2 - w$	Yes
	E_{B}	$w_1 + w_2 - w - \rho\frac{\mathrm{dV}}{\mathrm{dt}} = 0$	
3	E_{A}	$\rho\frac{\mathrm{dV}}{\mathrm{dt}} = w_1 + w_2 - w$	No
	E_{B}	$\rho\frac{\mathrm{dV}}{\mathrm{dt}} = w_1 - w_2 - w$	

Table 1. Pairs of equations used in experiments.

The two equations in Case 2 had the same set of variables and the same solutions for one variable, for example, w_1; thereby, Algorithm 1 returned true (Lines 9 and 10).

In Case 3 where the two equations had the same set of variables but the solutions for one variable were different, Algorithm 1 returned false (Lines 11 and 12).

Algorithm 1 fails to accurately judge the equivalence of equations in the following two cases: 1) when the number of the solutions for one variable is more than one and 2) when either equation cannot be solved. The first case occurs when all variables in an equation appear in a second or higher-order form. Since such equations have been rarely seen in describing physical models, they would not be a problem in practice. The second case appears when an equation consists only of partial derivatives of variables, where it is impossible to solve for a single variable without information other than the equation. Developing a method to deal with such cases is a subject for future work.

Cases of equation group equivalence judgment. In Case 4, the sets of variables of the two equation groups V_{A} and V_{B} were different. Algorithm 3 first derived the between-shared variables and within-shared variables of the two equation groups as follows (Lines 1–3):

$$V_{\mathrm{bs}} = \{r_{\mathrm{A}}, k_0, E, R, T, C_{\mathrm{A}}, V, t, q, C_0, \rho, C, w, T_{\mathrm{i}}, H_{\mathrm{r}}, U, T_{\mathrm{c}}\}, \tag{1}$$

$$V_{\mathrm{ws,E_A}} = \{r_{\mathrm{A}}, T, C_{\mathrm{A}}, V, t\}, \tag{2}$$

$$V_{\mathrm{ws,E_B}} = \{r_{\mathrm{A}}, k, T, C_{\mathrm{A}}, V, t\}. \tag{3}$$

Then, the variable appearing only in E_{B}, $V^*_{E_{\mathrm{B}}} = \{k\}$ was obtained (Line 5), and k in E_{B} was eliminated by substituting $k = k_0 \exp\left(-E/(RT)\right)$ into $-r_{\mathrm{A}} = kC_{\mathrm{A}}$ (Lines 13–15). After this variable elimination, V_{A} and V_{B} became the same, and all equations in E_{A} were equivalent to those in E_{B}; thus, Algorithm 3 returned true (Lines 20 and 21).

In Case 5, E_{A} and E_{B} had different sets of variables, and variable elimination was required in both equation groups. As the same method in Case 4, Algorithm 3 eliminated the variables in two equation groups (Lines 9–18) and returned true (Lines 20 and 21).

Case	Equation group		Equivalent
4	E_A	$-r_A = k_0 \exp(-E/(RT))C_A$ $V\frac{dC_A}{dt} = q(C_0 - C_A) + r_A V$ $V\rho C\frac{dT}{dt} = wC(T_i - T) + H_r V r_A + UA(T_c - T)$	Yes
	E_B	$-r_A = kC_A$ $k = k_0 \exp(-E/(RT))$ $V\frac{dC_A}{dt} = q(C_0 - C_A) + r_A V$ $V\rho C\frac{dT}{dt} = wC(T_i - T) + H_r V r_A + UA(T_c - T)$	
5	E_A	$A\frac{dh}{dt} = q_i - C_v\sqrt{h}$ $C_v = C_0\sqrt{g/g_c}$	Yes
	E_B	$P = P_a + \rho gh/g_c$ $q = C_0\sqrt{(P - P_a)/\rho}$ $A\frac{dh}{dt} = q_i - q$	
6	E_A	$-r_A = k_0 \exp(-E/(RT))C_A$ $V\frac{dC_A}{dt} = q(C_0 - C_A) + r_A V$ $V\rho C\frac{dT}{dt} = wC(T_i - T) + H_r V r_A + UA(T_c - T)$	No
	E_B	$A\frac{dh}{dt} = q_i - C_v\sqrt{h}$ $C_v = C_0\sqrt{g/g_c}$	
7	E_A	$-r_A = kC_A$ $k = k_0 \exp(-E/(RT))$ $V\frac{dC_A}{dt} = q(C_0 - C_A) + r_A V$ $V\rho C\frac{dT}{dt} = wC(T_i - T) + H_r V r_A + UA(T_c - T)$	No
	E_B	$-r_A = kC_A^2$ $k = k_0 \exp(-E/(RT))$ $V\frac{dC_A}{dt} = q(C_0 - C_A) + r_A V$ $V\rho C\frac{dT}{dt} = wC(T_i - T) + H_r V r_A + UA(T_c - T)$	

Table 2. Pairs of equation groups used in experiments.

Case 6 had two equation groups with different variables. However, the variables appearing only in either equation group were not equal to the variable to be eliminated: for example, k_0 appearing only in E_A could not be eliminated. Hence, E_A and E_B were judged non-equivalent (Lines 6 and 7).

In Case 7, the equation groups had different variables, and the sets of variables became the same after variable elimination. The first equation in E_A was not equivalent to any equation in E_B; thereby Algorithm 3 returned false (Lines 22 and 23).

Algorithm 3 sometimes fails to precisely judge the equivalence when an equation group does not lead to a single form after variable elimination. Assuming that we have an equation group as follows and k_1 needs to be eliminated.

$$\begin{aligned} k_1 &= k_{10}\exp(-E_1/(RT)), \\ k_2 &= k_{20}\exp(-E_2/(RT)), \\ \frac{dx_1}{dt} &= -k_1x_1, \\ \frac{dx_2}{dt} &= -k_1x_1 - k_2x_2. \end{aligned} \tag{4}$$

The number of the solutions for k_1 is three, and the equation groups after eliminating k_1 depend on the solution used for the substitution. Although such examples will need to be addressed in the future, the proposed method is useful for judging the equivalence of many types of physical models as shown in Supplementary Information. (Table S1)

Conclusion

We proposed a simple rule-based method for equation group equivalence judgment. The proposed method eliminates variables that appear only in either equation group and checks whether all equations in the two equation groups have a one-to-one relationship. The method was implemented in Python and a Python-based CAS, Sympy, and 50 equivalent and non-equivalent pairs of equations and equation groups were used for experiments. The results have shown that the proposed method can accurately judge whether two equation groups are equivalent.

The proposed method still has some limitations. The method has two assumptions to be removed: (1) any equation is solvable for one variable, and (2) the number of solutions for each equation is one. Furthermore, the equivalence judgment algorithm for equation groups (Algorithm 3) cannot correctly judge the equivalence of some equation groups that do not lead to a single form after variable elimination as explained in Section 3.2. Our future work will address these limitations by extending the method in this paper. Furthermore, we plan to expand the scope to more types of calculations, such as summation symbol Σ, integral symbol $\int$, vectors, and matrices.

Data availability

The datasets used in the current study are available at https://github.com/humansys-lab/dae-equiv-judge.

Code availability

The code used in the current study is available at https://github.com/humansys-lab/dae-equiv-judge.

Received: 7 December 2022; Accepted: 5 July 2023
Published online: 17 July 2023

References

1. Fortunato, S. *et al.* Science of science. *Science* **359**, eaao0185. https://doi.org/10.1126/science.aao0185 (2018).
2. Ho, D., Shkolnik, A. S., Ferraro, N. J., Rizkin, B. A. & Hartman, R. L. Using word embeddings in abstracts to accelerate metallocene catalysis polymerization research. *Comput. Chem. Eng.* **141**, 107026. https://doi.org/10.1016/j.compchemeng.2020.107026 (2020).
3. Kumar, A., Ganesh, S., Gupta, D. & Kodamana, H. A text mining framework for screening catalysts and critical process parameters from scientific literature - a study on hydrogen production from alcohol. *Chem. Eng. Res. Design* **184**, 90–102. https://doi.org/10.1016/j.cherd.2022.05.018 (2022).
4. Zaki, M., Jayadeva, & Krishnan, N. M. A. Extracting processing and testing parameters from materials science literature for improved property prediction of glasses. *Chem. Eng. Process. Process Intensif.* **180**, 108607. https://doi.org/10.1016/j.cep.2021.108607 (2022).
5. Lin, J., Wang, X., Wang, Z., Beyette, D. & Liu, J.-C. Prediction of mathematical expression declarations based on spatial, semantic, and syntactic analysis. In *Proceedings of the ACM Symposium on Document Engineering 2019*, DocEng '19, 1–10, https://doi.org/10.1145/3342558.3345399 (Berlin, Germany, 2019).
6. Jo, H., Kang, D., Head, A. & Hearst, M. A. Modeling mathematical notation semantics in academic papers. In *Findings of the Association for Computational Linguistics: EMNLP 2021*, 3102–3115, https://doi.org/10.18653/v1/2021.findings-emnlp.266 (Punta Cana, Dominican Republic, 2021).
7. Kang, D. *et al.* Document-level definition detection in scholarly documents: Existing models, error analyses, and future directions. In *Proceedings of the First Workshop on Scholarly Document Processing*, 196–206, https://doi.org/10.18653/v1/2020.sdp-1.22 (Online, 2020).
8. Kato, S., Kanegami, K. & Kano, M. ProcessBERT: A pre-trained language model for judging equivalence of variable definitions in process models*. *IFAC-PapersOnLine* **55**, 957–962. https://doi.org/10.1016/j.ifacol.2022.07.568 (2022).
9. Mikolov, T., Chen, K., Corrado, G. & Dean, J. Efficient estimation of word representations in vector space. Preprint at arxiv:1301.3781 (2013).
10. Vaswani, A. *et al.* Attention is all you need. In *Advances in neural information processing systems*, 5998–6008 (2017).
11. Devlin, J., Chang, M.-W., Lee, K. & Toutanova, K. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, 4171–4186, https://doi.org/10.18653/v1/N19-1423 (Minneapolis, Minnesota, 2019).
12. Mansouri, B. *et al.* Tangent-CFT: An embedding model for mathematical formulas. In *Proceedings of the 2019 ACM SIGIR International Conference on Theory of Information Retrieval*, ICTIR '19, 11–18, https://doi.org/10.1145/3341981.3344235 (Santa Clara, CA, USA, 2019).
13. Peng, S., Yuan, K., Gao, L. & Tang, Z. MathBERT: A pre-trained model for mathematical formula understanding. Preprint at arxiv:2105.00377 (2021).
14. Lewkowycz, A. *et al.* Solving quantitative reasoning problems with language models. Preprint at arxiv:2206.14858 (2022).
15. Ausbrooks, R. *et al.* Mathematical markup language (MathML) version 3.0 2nd edn (W3C recommendation). Tech. Rep., W3C (2014).
16. Kohlhase, M. *OMDoc – An open markup format for mathematical documents [version 1.2]: Foreword by Alan Bundy* (Springer, 2006).
17. Kamali, S. & Tompa, F. W. Improving mathematics retrieval. In *DML 2009. Towards digital mathematics library*, 37–48 (Grand Bend, Ontario, Canada, 2009).
18. Greiner-Petter, A., Schubotz, M., Cohl, H. S. & Gipp, B. Semantic preserving bijective mappings for expressions involving special functions between computer algebra systems and document preparation systems. *Aslib J. Inf. Manag.*https://doi.org/10.1108/AJIM-08-2018-0185 *(2019).*
19. Greiner-Petter, A., Schubotz, M., Aizawa, A. & Gipp, B. Making presentation math computable: Proposing a context sensitive approach for translating latex to computer algebra systems. *Math. Softw.- ICMS*https://doi.org/10.1007/978-3-030-52200-1_33 *(2020).*
20. Simplification - sympy 1.12 documentation. https://docs.sympy.org/latest/tutorials/intro-tutorial/simplification.html (2023). Accessed: 03-June-2023].
21. Simplify - Wolfram Language & System Documentation Center. https://reference.wolfram.com/language/ref/Simplify.html (2014). [Accessed: 03-June-2023].
22. Seborg, D. E., Edgar, T. F., Mellichamp, D. A. & Doyle, F. J. III. *Process dynamics and control* (John Wiley & Sons, 2010).
23. Meurer, A. *et al.* SymPy: Symbolic computing in Python. *PeerJ Comput. Sci.* **3**, e103. https://doi.org/10.7717/peerj-cs.103 (2017).

Acknowledgements
This work was supported by JSPS KAKENHI Grant Number JP21K18849 and JP23K13595.

Author contributions
All authors contributed to the study conception and design. S.K. and C.Z. performed data collection and experiments. The first draft of the manuscript was written by S.K. and all authors commented on previous versions of the manuscript. All authors read and approved the final manuscript.

Competing interests
The authors declare no competing interests.

Additional information
Supplementary Information The online version contains supplementary material available at https://doi.org/10.1038/s41598-023-38254-y.

Correspondence and requests for materials should be addressed to S.K.