

Truck Platooning

Stephanie Okosa Maxim Emile Speczyk Muhammad Umer Bin Yaqoob¹

Contents

1 Motivation	2
2 Foundation	2
2.1 Requirement Diagram	3
2.2 State Machine Diagrams	3
2.2.1 Braking Scenario	3
2.2.2 Steering Scenario	4
2.2.3 Lane Change Scenario	4
3 UPPAAL IMPLEMENTATION	4
3.1 UPPAAL Braking Modelling	5
3.2 UPPAAL Steering Modelling	5
3.3 UPPAAL Lane Change Modelling	5
3.4 UPPAAL Truck Coupling and Decoupling Modelling	6
4 Machine Learning Algorithm	6
4.1 Decision Tree Classifier	7
4.2 Isolation Forest Model	7

¹ stephanie-chinenye.okosa@stud.hshl.de, muhammad-umer-bin.yaqoob@stud.hshl.de, maxim-emile.speczyk@stud.hshl.de

5	Truck Platoon Scenarios Simulation	8
5.1	Braking Simulation	8
5.2	Steering Simulation	9
5.3	Lane Change Simulation	9
6	Declaration of Originality	10

Abstract: This project presents an autonomous truck platooning system's design, implementation, and testing. This technological solution promises to revolutionise the freight industry by increasing efficiency and safety while reducing costs. The system, modelled using UML, incorporates a machine learning algorithm to select the platoon leader based on multiple parameters. It considers various scenarios like steering, lane change, braking, joining, and leaving the platoon, including potential communication failures implemented in UPPAAL. The project will culminate in a comprehensive simulation environment using Python.

1 Motivation

Autonomous vehicles have revolutionised our society, bringing with them a myriad of opportunities and challenges. Our team has embarked on a project to develop an autonomous truck platooning system that can simulate the steering, braking, lane change, coupling and decoupling of the trucks during motion. Truck platooning essentially involves several trucks equipped with advanced driver assistance systems. These trucks follow each other closely on a motorway, literally forming a convoy or 'platoon'. The importance of this arrangement lies in the many benefits it offers which include; improved road safety, reduced fuel consumption, reduced aerodynamic drag, lower CO_2 emissions and more efficient traffic flow.

2 Foundation

In the early stages of our project, each member of our team brainstormed and generated detailed scenarios for our autonomous truck platooning system. This step helped us to identify key functional requirements and gain a deep understanding of the problem from the user's perspective. We then used UML (Unified Modelling Language), to visually represent these scenarios and the system as a whole, detailing the relationships and data flow between entities. The UML designs served as blueprints for the architecture of our system, providing a solid foundation for model specification and final implementation, while facilitating efficient communication and shared understanding among team members and stakeholders.

2.1 Requirement Diagram

Initial requirements of the truck platooning system were discussed, the following requirements were iterated, and identified as critical for the system to function properly as shown in [1] below.

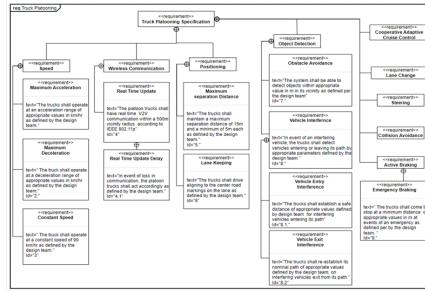


Fig. 1: Requirement Diagram

<https://github.com/MaximSpeczyk/Autonomous-Lab-A-Group-4/blob/main/requirements/truckPlatooningRequirements.png>

2.2 State Machine Diagrams

In this section we show the different scenarios states to enable us model the the timing behaviour in UPPAAL.

2.2.1 Braking Scenario

The autonomous truck platooning project focuses on synchronised braking mechanisms across the fleet. When the master truck initiates braking, every other truck must follow suit to maintain safe following distances and avoid collisions. This is achieved through efficient communication paths between the lead and following trucks as shown in [2]below.

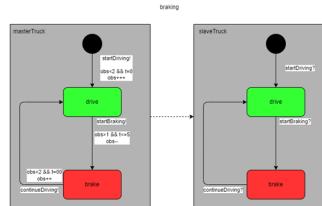


Fig. 2: Braking Scenario

<https://github.com/MaximSpeczyk/Autonomous-Lab-A-Group-4/blob/main/state%20Machine/braking.png>

2.2.2 Steering Scenario

This scenario in [3] ensures that the subordinate, or 'slave', trucks follows the lead or 'master' truck within the platoon at real time by, implementing a communication protocol wherein the leading, or 'master', truck transmits directional commands to the follower trucks. This facilitates seamless and synchronized transitions between directional states, e.g., initiating rightward or leftward shifts. The behaviour of this brake control system is illustrated by a state machine diagram in our system design.

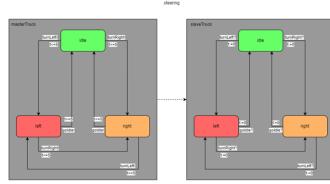


Fig. 3: Steering Scenario

<https://github.com/MaximSpeczyk/Autonomous-Lab-A-Group-4/blob/main/state%20Machine/steering.png>

2.2.3 Lane Change Scenario

Lane change coordination is key within the truck platoon fleet. The aim is to ensure that when the master truck makes a lane change, the slave trucks systematically reproduce the action to maintain the integrity of the platoon. These lane changes are managed by a highly responsive communication system between the trucks as shown in [4] below.

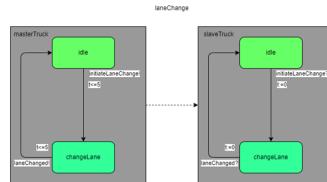


Fig. 4: Lane Change Scenario

<https://github.com/MaximSpeczyk/Autonomous-Lab-A-Group-4/blob/main/state%20Machine/laneChange.png>

3 UPPAAL IMPLEMENTATION

Here data, signal, events that are required for interactive communication between the trucks are identified, appropriate protocol for each scenario using timed automata specified, and modelled using UPPAAL. All scenarios described in this section model a platoon system where one truck acts as the 'truckMaster' and the two slave trucks, 'truckSlave' follow.

3.1 UPPAAL Braking Modelling

Here, the master truck leads the platoon, broadcasting driving instructions via three primary channels; 'continueDriving', 'Brake', and 'continueBraking'. Depending on these signals, the slave trucks respond by maintaining speed, applying brakes, or continuing to brake. This synchronization enables the platoon to operate harmoniously and safely.

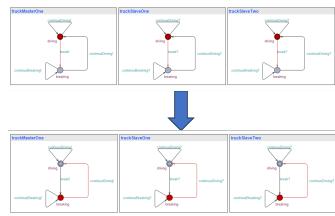


Fig. 5: UPPAAL Braking
<https://github.com/MaximSpeczyk/Autonomouse-Lab-A-Group-4/blob/main/scenarios/braking.xml>

3.2 UPPAAL Steering Modelling

The steering behaviour of both trucks is defined by three states: idle, turnLeft and turnRight. The master truck's state transitions are determined by a global clock and its own synchronisation events, while the slave truck mimics the master's steering actions. The model thus demonstrates a coordinated steering system in a truck platoon, ensuring that the slave truck accurately replicates the directions of the master.

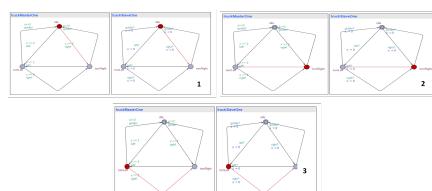


Fig. 6: UPPAAL Steering
<https://github.com/MaximSpeczyk/Autonomouse-Lab-A-Group-4/blob/main/scenarios/steering.xml>

3.3 UPPAAL Lane Change Modelling

Lane change is initiated by the lead truck after a set time, prompting the following trucks to begin the same procedure. A 'Watchdog' entity monitors the operation, ready to raise an alarm if the lane change takes longer than expected. Upon completion, the lead truck signals the end of the lane change, upon which all trucks resume cruising. This system enables safe and synchronized lane changes within a truck platoon.

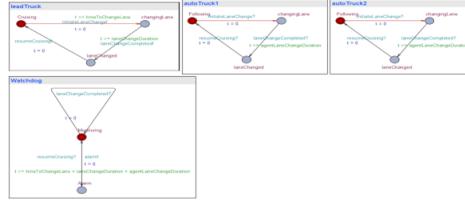


Fig. 7: UPPAAL Lane Change

<https://github.com/MaximSpeczyk/Autonomouse-Lab-A-Group-4/blob/main/scenarios/laneChanging.xml>

3.4 UPPAAL Truck Coupling and Decoupling Modelling

In this scenario, each truck initially operates independently until it identifies a suitable environment for platooning, such as a motorway. When it detects a potential platoon, the truck sends a request to join, which is evaluated by the existing platoon, taking into account factors such as the size of the platoon, the compatibility of the new truck, and the potential benefits and risks. If the request is approved, the truck joins the platoon, adjusting its speed and positioning to form a coordinated and efficient transport unit. Critical to this operation is the constant communication between the trucks, which is monitored by a watchdog unit that ensures consistent data exchange and monitors for potential anomalies. If the watchdog detects a communication disruption within a truck that exceeds a certain threshold, the affected truck is removed from the platoon until the problem is resolved. In the event of a major malfunction affecting the entire platoon, the platoon disbands and each truck returns to independent operation, demonstrating the resilience and adaptability of the system.

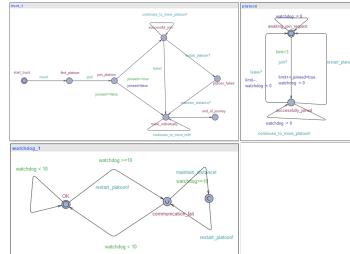


Fig. 8: UPPAAL Coupling and Decoupling

<https://github.com/MaximSpeczyk/Autonomouse-Lab-A-Group-4/blob/main/scenarios/couplingAndDecoupling.xml>

4 Machine Learning Algorithm

To determine the lead vehicle of the platoon for simulating the various scenarios, machine learning algorithm is applied on data information of the trucks with predefined features to

train our model to autonomously identify the 'truckMaster' for each platoon. Two different methods are applied in order to determine the algorithm with a better prediction accuracy.

4.1 Decision Tree Classifier

Here, the decision tree classifier is used for classification of the 'truckMaster' from the 'truckSlave' in the platoon using a set of predefined features as shown in [9]. However the prediction score is 50% therefore, we try the isolation forest model to improve the prediction accuracy of the model.

	distance_route	max_route_match	fuel_consumption	body_characteristics	equipment_sensors	leader_vehicle
0	808	100	245	3769	91	0
1	899	42	308	2732	82	1
2	899	43	237	3876	72	0
3	823	35	303	2934	73	0
4	897	50	253	3649	57	0

```

# Define features and target
x = data.drop(['leader_vehicle'], axis=1)
y = data['leader_vehicle']

# Split the data into training and testing sets
x_train, x_test, y_train, y_test = train_test_split(x, y, test_size=0.3, random_state=1)

# Create a DecisionTreeClassifier and fit it to the training data
clf = DecisionTreeClassifier(criterion='entropy', max_depth=3)
clf = clf.fit(x_train, y_train)

# Predict the test set results
y_pred = clf.predict(x_test)

# Print the accuracy of the model
print("Accuracy:", metrics.accuracy_score(y_test, y_pred))

# Plot the decision tree
plt.figure(figsize=(15,15))
clf = DecisionTreeClassifier(max_depth=3).fit(x, y)
dot_data = export_graphviz(clf, filled=True, fontsize=10)
plt.show()

```

Fig. 9: Decision Tree Classifier

<https://github.com/MaximSpeczyk/Autonomouse-Lab-A-Group-4/blob/main/Leader%20vehicle%20selection/decisionTreeClassifier/DecisionTreeClassifier.py>

4.2 Isolation Forest Model

The isolation forest model is based on decision trees and is used in identifying anomalies in a dataset by employing binary trees. The idea is to be able to spot the distinct trucks which would differ from the other trucks to be the 'truckMaster'.

	Relative Position	Speed	Acceleration	Eng	Sensor State	Vehicle Characteristics	Behavior	Time Duration
0	16.42026	86.59919	0.07608	3.35222	ObjectDetected	Truck	Constant	546.14027
1	16.43398	86.59919	2.371729	22.16324	ObjectDetected	Truck	Constant	546.135498
2	0.03482	75.95367	-9.77892	44.09593	ObjectDetected	Truck	Enter	239.756485
3	52.49013	24.57039	6.02302	38.28575	ObjectDetected	Truck	Enter	239.800149
4	05.85484	03.95648	0.25091	34.54108	NoObjectDetected	Truck	Constant	203.540764

```

# Load the dataset from the csv file
data = pd.read_csv('truck_platoon_dataset.csv')

# Remove the 'Vehicle Characteristics' column
x = data.drop(['Vehicle Characteristics'], axis=1)
y = data['Vehicle Characteristics']

# Encode the categorical features using LabelEncoder
label_encoder = LabelEncoder()
for column in x:
    if x[column].dtype == object:
        x[column] = label_encoder.fit_transform(x[column])

# Split the data into training and test datasets
x_train, x_test, y_train, y_test = train_test_split(x, y, test_size=0.3, random_state=42)

```

Fig. 10: Isolation Forest Model [1]

<https://github.com/MaximSpeczyk/Autonomouse-Lab-A-Group-4/tree/main/Leader%20vehicle%20selection/isolationForestModel>

```

# Train the isolation forest model
model = IsolationForest(n_estimators=100, random_state=42)
model.fit(X_train)

# Predict the anomaly scores for the test dataset
anomaly_scores = model.decision_function(X_test)

# Identify the lead truck in the test dataset as the instance with the lowest anomaly score
lead_truck_index = np.argmin(anomaly_scores)
lead_truck = X_test[lead_truck_index]
print("Lead truck in test dataset!")
print(lead_truck)

# Calculate the anomaly score (lower score indicates a higher likelihood of being an anomaly)
# Note: g is used to classify anomalies
threshold = -anomaly_scores[lead_truck_index] # You can adjust this threshold as per your requirements

# Classify samples as anomalies or not based on the threshold
predictions = (anomaly_scores < threshold).astype(int)

# Calculate the accuracy score (percentage of correct classifications)
accuracy = (predictions == y_test).mean()
print("Accuracy: " + str(accuracy))

# Define the objective function for optimization
def accuracy_objective(threshold):
    predictions = (anomaly_scores < threshold).astype(int)
    accuracy = (predictions == y_test).mean()
    return -accuracy # Minimize the negative accuracy to maximize accuracy

# Create an optimizer to find the threshold that achieves accuracy
result = minimize_scalar(accuracy_objective, bounds=(-100, anomaly_scores), method='bounded')
best_accuracy = -result.fun

# Classify samples as anomalies or not based on the best threshold
predictions = (anomaly_scores < best_threshold).astype(int)

# Calculate the accuracy score (percentage of correct classifications)
accuracy = (predictions == y_test).mean()
print("Best Accuracy: " + str(accuracy))
print("Best Threshold: " + str(best_threshold))

```

Fig. 11: Isolation Forest Model [2]

<https://github.com/MaximSpeczyk/Autonomouse-Lab-A-Group-4/tree/main/Leader%20vehicle%20selection/isolationForestModel>

With the random selection of data for fitting the model, the prediction accuracy is a 1.0. However, limitations lie in the overfitting of the test samples, as the model is not trained to consider all possible features, the model fitting does not have any dependency on the feature set as it is trained randomly i.e., no related data to find the 'truckMaster', and the 'Gap' feature data is incorrect as the value is expected to be constant in order to meet the 'fuel efficiency' requirement. In order to mitigate some of these constraints and optimize prediction, we can use pass optimization on the dataset, have some adjustments made to the feature parameters, as well as fitting the model with the feature that have correlation for identifying which is the 'truckMaster' of the platoon.

5 Truck Platoon Scenarios Simulation

Python is used as a simulation environment for implementing the various modelled scenarios in UPPAAL. The Python code provided uses the asyncio library to simulate a platoon of vehicles with a leader and several followers.

5.1 Braking Simulation

Each vehicle is represented by a class, with the leader switching between cruising and braking states based on specified time intervals, and the followers proceed. The simulation runs to completion with all vehicles operating simultaneously as shown in [12] below.

Fig. 12: Braking Simulation

<https://github.com/MaximSpeczyk/Autonomouse-Lab-A-Group-4/blob/main/Python%20simulations/Braking.py>

5.2 Steering Simulation

The leader truck cyclically changes its state (idle, turn left, turn right) every second and stores these states in a queue. The follower mimics the leader's states, but with a delay of 2 seconds. The script uses the `asyncio` library to allow the leader and follower to operate simultaneously in a non-blocking manner, providing a basic model of a delayed vehicle platooning scenario.

Fig. 13: Steering Simulation
<https://github.com/MaximSpeczyk/Autonomous-Lab-A-Group-4/blob/main/Python%20simulations/steering.py>

5.3 Lane Change Simulation

Here, the leader changes lanes after a certain time, and the followers proceeds after a second delay. The state of each vehicle is monitored and updated using an asynchronous event loop from the `asyncio` library. After a successful lane change, the vehicles return to their initial state of following the leader, who returns to cruising mode as shown in [14] below.

```

1 import abc
2
3 class Vehicle(abc.ABC):
4     def __init__(self, name):
5         self.name = name
6         self.state = "resting"
7
8     @abc.abstractmethod
9     def leader(self):
10         pass
11
12     @abc.abstractmethod
13     def follow(self, leader):
14         pass
15
16     @abc.abstractmethod
17     def turn_left(self):
18         pass
19
20     @abc.abstractmethod
21     def turn_right(self):
22         pass
23
24     @abc.abstractmethod
25     def go_straight(self):
26         pass
27
28     @abc.abstractmethod
29     def stop(self):
30         pass
31
32     @abc.abstractmethod
33     def sleep(self):
34         pass
35
36     @abc.abstractmethod
37     def wake_up(self):
38         pass
39
40     @abc.abstractmethod
41     def change_lane(self):
42         pass
43
44     @abc.abstractmethod
45     def get_state(self):
46         pass
47
48     @abc.abstractmethod
49     def set_state(self, state):
50         pass
51
52     @abc.abstractmethod
53     def __str__(self):
54         pass
55
56     @abc.abstractmethod
57     def __eq__(self, other):
58         pass
59
60     @abc.abstractmethod
61     def __ne__(self, other):
62         pass
63
64     @abc.abstractmethod
65     def __gt__(self, other):
66         pass
67
68     @abc.abstractmethod
69     def __lt__(self, other):
70         pass
71
72     @abc.abstractmethod
73     def __ge__(self, other):
74         pass
75
76     @abc.abstractmethod
77     def __le__(self, other):
78         pass
79
80     @abc.abstractmethod
81     def __hash__(self):
82         pass
83
84     @abc.abstractmethod
85     def __repr__(self):
86         pass
87
88     @abc.abstractmethod
89     def __format__(self, format):
90         pass
91
92     @abc.abstractmethod
93     def __bool__(self):
94         pass
95
96     @abc.abstractmethod
97     def __nonzero__(self):
98         pass
99
100    @abc.abstractmethod
101    def __len__(self):
102        pass
103
104    @abc.abstractmethod
105    def __iter__(self):
106        pass
107
108    @abc.abstractmethod
109    def __next__(self):
110        pass
111
112    @abc.abstractmethod
113    def __radd__(self, other):
114        pass
115
116    @abc.abstractmethod
117    def __rsub__(self, other):
118        pass
119
120    @abc.abstractmethod
121    def __rmul__(self, other):
122        pass
123
124    @abc.abstractmethod
125    def __rtruediv__(self, other):
126        pass
127
128    @abc.abstractmethod
129    def __rmod__(self, other):
130        pass
131
132    @abc.abstractmethod
133    def __rpow__(self, other):
134        pass
135
136    @abc.abstractmethod
137    def __rdiv__(self, other):
138        pass
139
140    @abc.abstractmethod
141    def __rtruediv__(self, other):
142        pass
143
144    @abc.abstractmethod
145    def __rmod__(self, other):
146        pass
147
148    @abc.abstractmethod
149    def __rpow__(self, other):
150        pass
151
152    @abc.abstractmethod
153    def __rdiv__(self, other):
154        pass
155
156    @abc.abstractmethod
157    def __rtruediv__(self, other):
158        pass
159
160    @abc.abstractmethod
161    def __rmod__(self, other):
162        pass
163
164    @abc.abstractmethod
165    def __rpow__(self, other):
166        pass
167
168    @abc.abstractmethod
169    def __rdiv__(self, other):
170        pass
171
172    @abc.abstractmethod
173    def __rtruediv__(self, other):
174        pass
175
176    @abc.abstractmethod
177    def __rmod__(self, other):
178        pass
179
180    @abc.abstractmethod
181    def __rpow__(self, other):
182        pass
183
184    @abc.abstractmethod
185    def __rdiv__(self, other):
186        pass
187
188    @abc.abstractmethod
189    def __rtruediv__(self, other):
190        pass
191
192    @abc.abstractmethod
193    def __rmod__(self, other):
194        pass
195
196    @abc.abstractmethod
197    def __rpow__(self, other):
198        pass
199
200    @abc.abstractmethod
201    def __rdiv__(self, other):
202        pass
203
204    @abc.abstractmethod
205    def __rtruediv__(self, other):
206        pass
207
208    @abc.abstractmethod
209    def __rmod__(self, other):
210        pass
211
212    @abc.abstractmethod
213    def __rpow__(self, other):
214        pass
215
216    @abc.abstractmethod
217    def __rdiv__(self, other):
218        pass
219
220    @abc.abstractmethod
221    def __rtruediv__(self, other):
222        pass
223
224    @abc.abstractmethod
225    def __rmod__(self, other):
226        pass
227
228    @abc.abstractmethod
229    def __rpow__(self, other):
230        pass
231
232    @abc.abstractmethod
233    def __rdiv__(self, other):
234        pass
235
236    @abc.abstractmethod
237    def __rtruediv__(self, other):
238        pass
239
240    @abc.abstractmethod
241    def __rmod__(self, other):
242        pass
243
244    @abc.abstractmethod
245    def __rpow__(self, other):
246        pass
247
248    @abc.abstractmethod
249    def __rdiv__(self, other):
250        pass
251
252    @abc.abstractmethod
253    def __rtruediv__(self, other):
254        pass
255
256    @abc.abstractmethod
257    def __rmod__(self, other):
258        pass
259
260    @abc.abstractmethod
261    def __rpow__(self, other):
262        pass
263
264    @abc.abstractmethod
265    def __rdiv__(self, other):
266        pass
267
268    @abc.abstractmethod
269    def __rtruediv__(self, other):
270        pass
271
272    @abc.abstractmethod
273    def __rmod__(self, other):
274        pass
275
276    @abc.abstractmethod
277    def __rpow__(self, other):
278        pass
279
280    @abc.abstractmethod
281    def __rdiv__(self, other):
282        pass
283
284    @abc.abstractmethod
285    def __rtruediv__(self, other):
286        pass
287
288    @abc.abstractmethod
289    def __rmod__(self, other):
290        pass
291
292    @abc.abstractmethod
293    def __rpow__(self, other):
294        pass
295
296    @abc.abstractmethod
297    def __rdiv__(self, other):
298        pass
299
300    @abc.abstractmethod
301    def __rtruediv__(self, other):
302        pass
303
304    @abc.abstractmethod
305    def __rmod__(self, other):
306        pass
307
308    @abc.abstractmethod
309    def __rpow__(self, other):
310        pass
311
312    @abc.abstractmethod
313    def __rdiv__(self, other):
314        pass
315
316    @abc.abstractmethod
317    def __rtruediv__(self, other):
318        pass
319
320    @abc.abstractmethod
321    def __rmod__(self, other):
322        pass
323
324    @abc.abstractmethod
325    def __rpow__(self, other):
326        pass
327
328    @abc.abstractmethod
329    def __rdiv__(self, other):
330        pass
331
332    @abc.abstractmethod
333    def __rtruediv__(self, other):
334        pass
335
336    @abc.abstractmethod
337    def __rmod__(self, other):
338        pass
339
340    @abc.abstractmethod
341    def __rpow__(self, other):
342        pass
343
344    @abc.abstractmethod
345    def __rdiv__(self, other):
346        pass
347
348    @abc.abstractmethod
349    def __rtruediv__(self, other):
350        pass
351
352    @abc.abstractmethod
353    def __rmod__(self, other):
354        pass
355
356    @abc.abstractmethod
357    def __rpow__(self, other):
358        pass
359
360    @abc.abstractmethod
361    def __rdiv__(self, other):
362        pass
363
364    @abc.abstractmethod
365    def __rtruediv__(self, other):
366        pass
367
368    @abc.abstractmethod
369    def __rmod__(self, other):
370        pass
371
372    @abc.abstractmethod
373    def __rpow__(self, other):
374        pass
375
376    @abc.abstractmethod
377    def __rdiv__(self, other):
378        pass
379
380    @abc.abstractmethod
381    def __rtruediv__(self, other):
382        pass
383
384    @abc.abstractmethod
385    def __rmod__(self, other):
386        pass
387
388    @abc.abstractmethod
389    def __rpow__(self, other):
390        pass
391
392    @abc.abstractmethod
393    def __rdiv__(self, other):
394        pass
395
396    @abc.abstractmethod
397    def __rtruediv__(self, other):
398        pass
399
400    @abc.abstractmethod
401    def __rmod__(self, other):
402        pass
403
404    @abc.abstractmethod
405    def __rpow__(self, other):
406        pass
407
408    @abc.abstractmethod
409    def __rdiv__(self, other):
410        pass
411
412    @abc.abstractmethod
413    def __rtruediv__(self, other):
414        pass
415
416    @abc.abstractmethod
417    def __rmod__(self, other):
418        pass
419
420    @abc.abstractmethod
421    def __rpow__(self, other):
422        pass
423
424    @abc.abstractmethod
425    def __rdiv__(self, other):
426        pass
427
428    @abc.abstractmethod
429    def __rtruediv__(self, other):
430        pass
431
432    @abc.abstractmethod
433    def __rmod__(self, other):
434        pass
435
436    @abc.abstractmethod
437    def __rpow__(self, other):
438        pass
439
440    @abc.abstractmethod
441    def __rdiv__(self, other):
442        pass
443
444    @abc.abstractmethod
445    def __rtruediv__(self, other):
446        pass
447
448    @abc.abstractmethod
449    def __rmod__(self, other):
450        pass
451
452    @abc.abstractmethod
453    def __rpow__(self, other):
454        pass
455
456    @abc.abstractmethod
457    def __rdiv__(self, other):
458        pass
459
460    @abc.abstractmethod
461    def __rtruediv__(self, other):
462        pass
463
464    @abc.abstractmethod
465    def __rmod__(self, other):
466        pass
467
468    @abc.abstractmethod
469    def __rpow__(self, other):
470        pass
471
472    @abc.abstractmethod
473    def __rdiv__(self, other):
474        pass
475
476    @abc.abstractmethod
477    def __rtruediv__(self, other):
478        pass
479
480    @abc.abstractmethod
481    def __rmod__(self, other):
482        pass
483
484    @abc.abstractmethod
485    def __rpow__(self, other):
486        pass
487
488    @abc.abstractmethod
489    def __rdiv__(self, other):
490        pass
491
492    @abc.abstractmethod
493    def __rtruediv__(self, other):
494        pass
495
496    @abc.abstractmethod
497    def __rmod__(self, other):
498        pass
499
500    @abc.abstractmethod
501    def __rpow__(self, other):
502        pass
503
504    @abc.abstractmethod
505    def __rdiv__(self, other):
506        pass
507
508    @abc.abstractmethod
509    def __rtruediv__(self, other):
510        pass
511
512    @abc.abstractmethod
513    def __rmod__(self, other):
514        pass
515
516    @abc.abstractmethod
517    def __rpow__(self, other):
518        pass
519
520    @abc.abstractmethod
521    def __rdiv__(self, other):
522        pass
523
524    @abc.abstractmethod
525    def __rtruediv__(self, other):
526        pass
527
528    @abc.abstractmethod
529    def __rmod__(self, other):
530        pass
531
532    @abc.abstractmethod
533    def __rpow__(self, other):
534        pass
535
536    @abc.abstractmethod
537    def __rdiv__(self, other):
538        pass
539
540    @abc.abstractmethod
541    def __rtruediv__(self, other):
542        pass
543
544    @abc.abstractmethod
545    def __rmod__(self, other):
546        pass
547
548    @abc.abstractmethod
549    def __rpow__(self, other):
550        pass
551
552    @abc.abstractmethod
553    def __rdiv__(self, other):
554        pass
555
556    @abc.abstractmethod
557    def __rtruediv__(self, other):
558        pass
559
560    @abc.abstractmethod
561    def __rmod__(self, other):
562        pass
563
564    @abc.abstractmethod
565    def __rpow__(self, other):
566        pass
567
568    @abc.abstractmethod
569    def __rdiv__(self, other):
570        pass
571
572    @abc.abstractmethod
573    def __rtruediv__(self, other):
574        pass
575
576    @abc.abstractmethod
577    def __rmod__(self, other):
578        pass
579
580    @abc.abstractmethod
581    def __rpow__(self, other):
582        pass
583
584    @abc.abstractmethod
585    def __rdiv__(self, other):
586        pass
587
588    @abc.abstractmethod
589    def __rtruediv__(self, other):
590        pass
591
592    @abc.abstractmethod
593    def __rmod__(self, other):
594        pass
595
596    @abc.abstractmethod
597    def __rpow__(self, other):
598        pass
599
599    @abc.abstractmethod
600    def __rdiv__(self, other):
601        pass
602
603    @abc.abstractmethod
604    def __rtruediv__(self, other):
605        pass
606
607    @abc.abstractmethod
608    def __rmod__(self, other):
609        pass
610
611    @abc.abstractmethod
612    def __rpow__(self, other):
613        pass
614
615    @abc.abstractmethod
616    def __rdiv__(self, other):
617        pass
618
619    @abc.abstractmethod
620    def __rtruediv__(self, other):
621        pass
622
623    @abc.abstractmethod
624    def __rmod__(self, other):
625        pass
626
627    @abc.abstractmethod
628    def __rpow__(self, other):
629        pass
630
631    @abc.abstractmethod
632    def __rdiv__(self, other):
633        pass
634
635    @abc.abstractmethod
636    def __rtruediv__(self, other):
637        pass
638
639    @abc.abstractmethod
640    def __rmod__(self, other):
641        pass
642
643    @abc.abstractmethod
644    def __rpow__(self, other):
645        pass
646
647    @abc.abstractmethod
648    def __rdiv__(self, other):
649        pass
650
651    @abc.abstractmethod
652    def __rtruediv__(self, other):
653        pass
654
655    @abc.abstractmethod
656    def __rmod__(self, other):
657        pass
658
659    @abc.abstractmethod
660    def __rpow__(self, other):
661        pass
662
663    @abc.abstractmethod
664    def __rdiv__(self, other):
665        pass
666
667    @abc.abstractmethod
668    def __rtruediv__(self, other):
669        pass
670
671    @abc.abstractmethod
672    def __rmod__(self, other):
673        pass
674
675    @abc.abstractmethod
676    def __rpow__(self, other):
677        pass
678
679    @abc.abstractmethod
680    def __rdiv__(self, other):
681        pass
682
683    @abc.abstractmethod
684    def __rtruediv__(self, other):
685        pass
686
687    @abc.abstractmethod
688    def __rmod__(self, other):
689        pass
690
691    @abc.abstractmethod
692    def __rpow__(self, other):
693        pass
694
695    @abc.abstractmethod
696    def __rdiv__(self, other):
697        pass
698
699    @abc.abstractmethod
700    def __rtruediv__(self, other):
701        pass
702
703    @abc.abstractmethod
704    def __rmod__(self, other):
705        pass
706
707    @abc.abstractmethod
708    def __rpow__(self, other):
709        pass
710
711    @abc.abstractmethod
712    def __rdiv__(self, other):
713        pass
714
715    @abc.abstractmethod
716    def __rtruediv__(self, other):
717        pass
718
719    @abc.abstractmethod
720    def __rmod__(self, other):
721        pass
722
723    @abc.abstractmethod
724    def __rpow__(self, other):
725        pass
726
727    @abc.abstractmethod
728    def __rdiv__(self, other):
729        pass
730
731    @abc.abstractmethod
732    def __rtruediv__(self, other):
733        pass
734
735    @abc.abstractmethod
736    def __rmod__(self, other):
737        pass
738
739    @abc.abstractmethod
740    def __rpow__(self, other):
741        pass
742
743    @abc.abstractmethod
744    def __rdiv__(self, other):
745        pass
746
747    @abc.abstractmethod
748    def __rtruediv__(self, other):
749        pass
750
751    @abc.abstractmethod
752    def __rmod__(self, other):
753        pass
754
755    @abc.abstractmethod
756    def __rpow__(self, other):
757        pass
758
759    @abc.abstractmethod
760    def __rdiv__(self, other):
761        pass
762
763    @abc.abstractmethod
764    def __rtruediv__(self, other):
765        pass
766
767    @abc.abstractmethod
768    def __rmod__(self, other):
769        pass
770
771    @abc.abstractmethod
772    def __rpow__(self, other):
773        pass
774
775    @abc.abstractmethod
776    def __rdiv__(self, other):
777        pass
778
779    @abc.abstractmethod
780    def __rtruediv__(self, other):
781        pass
782
783    @abc.abstractmethod
784    def __rmod__(self, other):
785        pass
786
787    @abc.abstractmethod
788    def __rpow__(self, other):
789        pass
790
791    @abc.abstractmethod
792    def __rdiv__(self, other):
793        pass
794
795    @abc.abstractmethod
796    def __rtruediv__(self, other):
797        pass
798
799    @abc.abstractmethod
800    def __rmod__(self, other):
801        pass
802
803    @abc.abstractmethod
804    def __rpow__(self, other):
805        pass
806
807    @abc.abstractmethod
808    def __rdiv__(self, other):
809        pass
810
811    @abc.abstractmethod
812    def __rtruediv__(self, other):
813        pass
814
815    @abc.abstractmethod
816    def __rmod__(self, other):
817        pass
818
819    @abc.abstractmethod
820    def __rpow__(self, other):
821        pass
822
823    @abc.abstractmethod
824    def __rdiv__(self, other):
825        pass
826
827    @abc.abstractmethod
828    def __rtruediv__(self, other):
829        pass
830
831    @abc.abstractmethod
832    def __rmod__(self, other):
833        pass
834
835    @abc.abstractmethod
836    def __rpow__(self, other):
837        pass
838
839    @abc.abstractmethod
840    def __rdiv__(self, other):
841        pass
842
843    @abc.abstractmethod
844    def __rtruediv__(self, other):
845        pass
846
847    @abc.abstractmethod
848    def __rmod__(self, other):
849        pass
850
851    @abc.abstractmethod
852    def __rpow__(self, other):
853        pass
854
855    @abc.abstractmethod
856    def __rdiv__(self, other):
857        pass
858
859    @abc.abstractmethod
860    def __rtruediv__(self, other):
861        pass
862
863    @abc.abstractmethod
864    def __rmod__(self, other):
865        pass
866
867    @abc.abstractmethod
868    def __rpow__(self, other):
869        pass
870
871    @abc.abstractmethod
872    def __rdiv__(self, other):
873        pass
874
875    @abc.abstractmethod
876    def __rtruediv__(self, other):
877        pass
878
879    @abc.abstractmethod
880    def __rmod__(self, other):
881        pass
882
883    @abc.abstractmethod
884    def __rpow__(self, other):
885        pass
886
887    @abc.abstractmethod
888    def __rdiv__(self, other):
889        pass
890
891    @abc.abstractmethod
892    def __rtruediv__(self, other):
893        pass
894
895    @abc.abstractmethod
896    def __rmod__(self, other):
897        pass
898
899    @abc.abstractmethod
900    def __rpow__(self, other):
901        pass
902
903    @abc.abstractmethod
904    def __rdiv__(self, other):
905        pass
906
907    @abc.abstractmethod
908    def __rtruediv__(self, other):
909        pass
910
911    @abc.abstractmethod
912    def __rmod__(self, other):
913        pass
914
915    @abc.abstractmethod
916    def __rpow__(self, other):
917        pass
918
919    @abc.abstractmethod
920    def __rdiv__(self, other):
921        pass
922
923    @abc.abstractmethod
924    def __rtruediv__(self, other):
925        pass
926
927    @abc.abstractmethod
928    def __rmod__(self, other):
929        pass
930
931    @abc.abstractmethod
932    def __rpow__(self, other):
933        pass
934
935    @abc.abstractmethod
936    def __rdiv__(self, other):
937        pass
938
939    @abc.abstractmethod
940    def __rtruediv__(self, other):
941        pass
942
943    @abc.abstractmethod
944    def __rmod__(self, other):
945        pass
946
947    @abc.abstractmethod
948    def __rpow__(self, other):
949        pass
950
951    @abc.abstractmethod
952    def __rdiv__(self, other):
953        pass
954
955    @abc.abstractmethod
956    def __rtruediv__(self, other):
957        pass
958
959    @abc.abstractmethod
960    def __rmod__(self, other):
961        pass
962
963    @abc.abstractmethod
964    def __rpow__(self, other):
965        pass
966
967    @abc.abstractmethod
968    def __rdiv__(self, other):
969        pass
970
971    @abc.abstractmethod
972    def __rtruediv__(self, other):
973        pass
974
975    @abc.abstractmethod
976    def __rmod__(self, other):
977        pass
978
979    @abc.abstractmethod
980    def __rpow__(self, other):
981        pass
982
983    @abc.abstractmethod
984    def __rdiv__(self, other):
985        pass
986
987    @abc.abstractmethod
988    def __rtruediv__(self, other):
989        pass
990
991    @abc.abstractmethod
992    def __rmod__(self, other):
993        pass
994
995    @abc.abstractmethod
996    def __rpow__(self, other):
997        pass
998
999    @abc.abstractmethod
1000    def __rdiv__(self, other):
1001        pass
1002
1003    @abc.abstractmethod
1004    def __rtruediv__(self, other):
1005        pass
1006
1007    @abc.abstractmethod
1008    def __rmod__(self, other):
1009        pass
1010
1011    @abc.abstractmethod
1012    def __rpow__(self, other):
1013        pass
1014
1015    @abc.abstractmethod
1016    def __rdiv__(self, other):
1017        pass
1018
1019    @abc.abstractmethod
1020    def __rtruediv__(self, other):
1021        pass
1022
1023    @abc.abstractmethod
1024    def __rmod__(self, other):
1025        pass
1026
1027    @abc.abstractmethod
1028    def __rpow__(self, other):
1029        pass
1030
1031    @abc.abstractmethod
1032    def __rdiv__(self, other):
1033        pass
1034
1035    @abc.abstractmethod
1036    def __rtruediv__(self, other):
1037        pass
1038
1039    @abc.abstractmethod
1040    def __rmod__(self, other):
1041        pass
1042
1043    @abc.abstractmethod
1044    def __rpow__(self, other):
1045        pass
1046
1047    @abc.abstractmethod
1048    def __rdiv__(self, other):
1049        pass
1050
1051    @abc.abstractmethod
1052    def __rtruediv__(self, other):
1053        pass
1054
1055    @abc.abstractmethod
1056    def __rmod__(self, other):
1057        pass
1058
1059    @abc.abstractmethod
1060    def __rpow__(self, other):
1061        pass
1062
1063    @abc.abstractmethod
1064    def __rdiv__(self, other):
1065        pass
1066
1067    @abc.abstractmethod
1068    def __rtruediv__(self, other):
1069        pass
1070
1071    @abc.abstractmethod
1072    def __rmod__(self, other):
1073        pass
1074
1075    @abc.abstractmethod
1076    def __rpow__(self, other):
1077        pass
1078
1079    @abc.abstractmethod
1080    def __rdiv__(self, other):
1081        pass
1082
1083    @abc.abstractmethod
1084    def __rtruediv__(self, other):
1085        pass
1086
1087    @abc.abstractmethod
1088    def __rmod__(self, other):
1089        pass
1090
1091    @abc.abstractmethod
1092    def __rpow__(self, other):
1093        pass
1094
1095    @abc.abstractmethod
1096    def __rdiv__(self, other):
1097        pass
1098
1099    @abc.abstractmethod
1100    def __rtruediv__(self, other):
1101        pass
1102
1103    @abc.abstractmethod
1104    def __rmod__(self, other):
1105        pass
1106
1107    @abc.abstractmethod
1108    def __rpow__(self, other):
1109        pass
1110
1111    @abc.abstractmethod
1112    def __rdiv__(self, other):
1113        pass
1114
1115    @abc.abstractmethod
1116    def __rtruediv__(self, other):
1117        pass
1118
1119    @abc.abstractmethod
1120    def __rmod__(self, other):
1121        pass
1122
1123    @abc.abstractmethod
1124    def __rpow__(self, other):
1125        pass
1126
1127    @abc.abstractmethod
1128    def __rdiv__(self, other):
1129        pass
1130
1131    @abc.abstractmethod
1132    def __rtruediv__(self, other):
1133        pass
1134
1135    @abc.abstractmethod
1136    def __rmod__(self, other):
1137        pass
1138
1139    @abc.abstractmethod
1140    def __rpow__(self, other):
1141        pass
1142
1143    @abc.abstractmethod
1144    def __rdiv__(self, other):
1145        pass
1146
1147    @abc.abstractmethod
1148    def __rtruediv__(self, other):
1149        pass
1150
1151    @abc.abstractmethod
1152    def __rmod__(self, other):
1153        pass
1154
1155    @abc.abstractmethod
1156    def __rpow__(self, other):
1157        pass
1158
1159    @abc.abstractmethod
1160    def __rdiv__(self, other):
1161        pass
1162
1163    @abc.abstractmethod
1164    def __rtruediv__(self, other):
1165        pass
1166
1167    @abc.abstractmethod
1168    def __rmod__(self, other):
1169        pass
1170
1171    @abc.abstractmethod
1172    def __rpow__(self, other):
1173        pass
1174
1175    @abc.abstractmethod
1176    def __rdiv__(self, other):
1177        pass
1178
1179    @abc.abstractmethod
1180    def __rtruediv__(self, other):
1181        pass
1182
1183    @abc.abstractmethod
1184    def __rmod__(self, other):
1185        pass
1186
1187    @abc.abstractmethod
1188    def __rpow__(self, other):
1189        pass
1190
1191    @abc.abstractmethod
1192    def __rdiv__(self, other):
1193        pass
1194
1195    @abc.abstractmethod
1196    def __rtruediv__(self, other):
1197        pass
1198
1199    @abc.abstractmethod
1200    def __rmod__(self, other):
1201        pass
1202
1203    @abc.abstractmethod
1204    def __rpow__(self, other):
1205        pass
1206
1207    @abc.abstractmethod
1208    def __rdiv__(self, other):
1209        pass
1210
1211    @abc.abstractmethod
1212    def __rtruediv__(self, other):
1213        pass
1214
1215    @abc.abstractmethod
1216    def __rmod__(self, other):
1217        pass
1218
1219    @abc.abstractmethod
1220    def __rpow__(self, other):
1221        pass
1222
1223    @abc.abstractmethod
1224    def __rdiv__(self, other):
1225        pass
1226
1227    @abc.abstractmethod
1228    def __rtruediv__(self, other):
1229        pass
1230
1231    @abc.abstractmethod
1232    def __rmod__(self, other):
1233        pass
1234
1235    @abc.abstractmethod
1236    def __rpow__(self, other):
1237        pass
1238
1239    @abc.abstractmethod
1240    def __rdiv__(self, other):
1241        pass
1242
1243    @abc.abstractmethod
1244    def __rtruediv__(self, other):
1245        pass
1246
1247    @abc.abstractmethod
1248    def __rmod__(self, other):
1249        pass
1250
1251    @abc.abstractmethod
1252    def __rpow__(self, other):
1253        pass
1254
1255    @abc.abstractmethod
1256    def __rdiv__(self, other):
1257        pass
1258
1259    @abc.abstractmethod
1260    def __rtruediv__(self, other):
1261        pass
1262
1263    @abc.abstractmethod
1264    def __rmod__(self, other):
1265        pass
1266
1267    @abc.abstractmethod
1268    def __rpow__(self, other):
1269        pass
1270
1271    @abc.abstractmethod
1272    def __rdiv__(self, other):
1273        pass
1274
1275    @abc.abstractmethod
1276    def __rtruediv__(self, other):
1277        pass
1278
1279    @abc.abstractmethod
1280    def __rmod__(self, other):
1281        pass
1282
1283    @abc.abstractmethod
1284    def __rpow__(self, other):
1285        pass
1286
1287    @abc.abstractmethod
1288    def __rdiv__(self, other):
1289        pass
1290
1291    @abc.abstractmethod
1292    def __rtruediv__(self, other):
1293        pass
1294
1295    @abc.abstractmethod
1296    def __rmod__(self, other):
1297        pass
1298
1299    @abc.abstractmethod
1300    def __rpow__(self, other):
1301        pass
1302
1303    @abc.abstractmethod
1304    def __rdiv__(self, other):
1305        pass
1306
1307    @abc.abstractmethod
1308    def __rtruediv__(self, other):
1309        pass
1310
1311    @abc.abstractmethod
1312    def __rmod__(self, other):
1313        pass
1314
1315    @abc.abstractmethod
1316    def __rpow__(self, other):
1317        pass
1318
1319    @abc.abstractmethod
1320    def __rdiv__(self, other):
1321        pass
1322
1323    @abc.abstractmethod
1324    def __rtruediv__(self, other):
1325        pass
1326
1327    @abc.abstractmethod
1328    def __rmod__(self, other):
1329        pass
1330
1331    @abc.abstractmethod
1332    def __rpow__(self, other):
1333        pass
1334
1335    @abc.abstractmethod
1336    def __rdiv__(self, other):
1337        pass
1338
1339    @abc.abstractmethod
1340    def __rtruediv__(self, other):
1341        pass
1342
1343    @abc.abstractmethod

```

Date & Place - Stephanie Okosa