COLREG-Compliant Machine Learning for Safe and Legal Autonomous Maritime Navigation

Alfie Anthony Treloar*, Dany Varghese†, Shubhi Verma†, Alireza Tamaddoni-Nezhad†, Alan Hunter*

*Department of Mechanical Engineering, University of Bath, Bath, UK
{a.o.anthony.treloar,a.j.hunter}@bath.ac.uk

†School of Computer Science and Electronic Engineering, University of Surrey, Guildford, UK
{dany.varghese,s.verma,a.tamaddoni-nezhad}@surrey.ac.uk

***Abstract*—This paper presents preliminary work on integrating symbolic learning and reasoning into autonomous maritime systems using inductive logic programming (ILP). A key challenge in operationalising ILP is bridging the gap between continuous sensing and actuation data and discrete symbolic logic. We propose a framework that enables autonomous vessels to query maritime rules (COLREGs) and learn from human oversight. Using the ILP system PyGol, we demonstrate the learning of COLREG Rule 13 for overtaking situations from discretised bearing data, and further explore the learning of an exception to Rule 15 for crossing situations through examples inspired by case law. These results show the potential for interpretable, legally compliant decision-making and lay the groundwork for learning more complex rules in dynamic maritime environments.**

***Index Terms*—Maritime law; autonomy; machine learning; inductive logic programming**

I. Introduction

Autonomous maritime vessels offer the potential for increased efficiency, safety, and sustainability. However, enabling these vessels to make automated decisions in complex and uncertain scenarios requires robust and explainable artificial intelligence (AI). A critical aspect of this challenge is ensuring compliance with the International Regulations for Preventing Collisions at Sea (COLREGs) [1], [2], which are essential for human safety, public acceptance, and legal and insurance frameworks. The COLREGs are intentionally written with a degree of vagueness to accommodate the complexity of real-world situations and to allow for human interpretation in courts of law. While this flexibility is manageable for human operators who can articulate their reasoning, it presents a significant challenge for AI systems [3], [4], which must not only interpret these rules but also explain their decision-making processes in a transparent and understandable manner. There have been various approaches to implementing decision making for COLREGs-compliant autonomous control systems [5], from fuzzy logic systems [6]–[8] to the more recent reinforcement learning based methods [9]–[11].

Recent advancements in statistical AI have led to impressive performance in narrow perceptive tasks, particularly through the use of neural networks. However, these models are often considered "black boxes" due to their reliance on vast numbers of numerical weights and connections, which obscure the reasoning behind their outputs. For autonomous systems to gain widespread acceptance, transparency and explainability are crucial [12]. While neural models can offer limited insights, such as confidence scores, and can be employed for decision-making through techniques like reinforcement learning, the overall transparency and explainability of learned neural policies remain limited.

Traditional symbolic AI, including expert systems based on formal logic, offered clear reasoning pathways but lacked the perceptive capabilities of modern neural networks. Neuro-symbolic AI seeks to combine the strengths of both approaches: the perceptive power of neural models with the explainability of symbolic reasoning [13]. This hybrid approach holds promise for developing autonomous systems that are both effective and interpretable. Two key research challenges in this domain are: 1) developing efficient algorithms capable of learning symbolic logic programs from examples, and 2) integrating these symbolic reasoning components with neural perceptive models and real-world systems. The latter challenge is the primary focus of this paper.

A major hurdle in operationalising symbolic AI in real-world systems is bridging the gap between continuous numerical data, used by sensors and actuators, and the discrete symbolic logic required for reasoning. Autonomous systems interact with the world through continuous physical quantities, whereas symbolic logic operates on atomic predicates. This discrepancy makes it difficult to express world models, decisions, and predictions in symbolic logic.

This paper presents preliminary work on implementing a conceptual framework for autonomous maritime vessels using inductive logic programming (ILP) [14], a technique that enables the learning of symbolic rules from examples. The framework, illustrated in Figure 1, integrates a modern autonomy system equipped with various sensors that perceive the physical environment. Neural perceptive models are employed to interpret sensor data, such as detecting and estimating the relative pose of nearby vessels. A key component of the system is its interface with a symbolic knowledge base, which allows it to query the COLREGs for guidance on safe and legally compliant decision-making. Crucially, the framework also incorporates human oversight, enabling operators to override

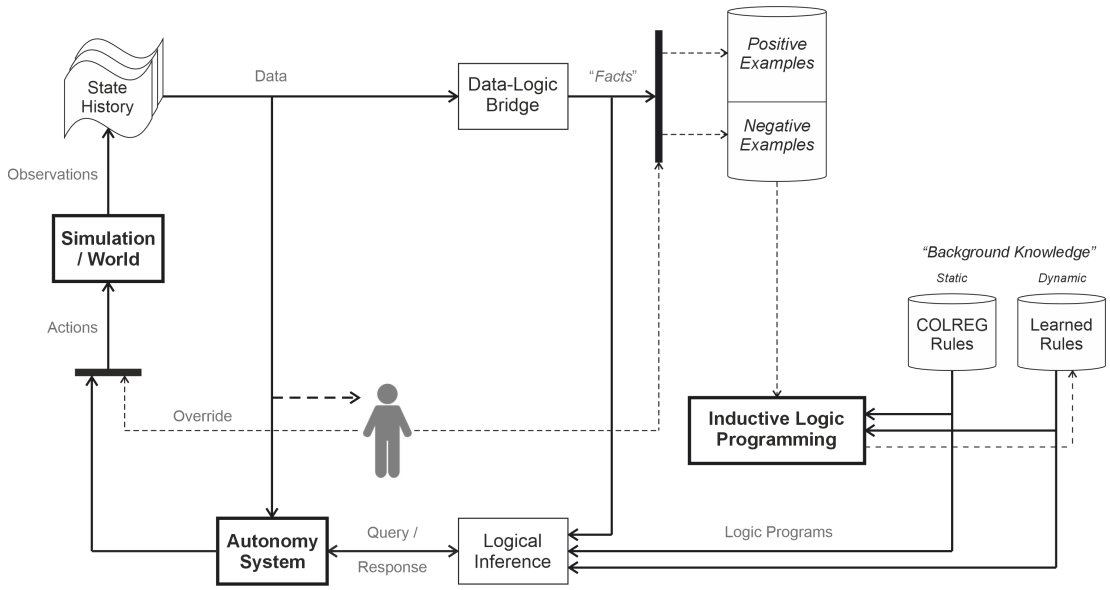


Fig. 1: Conceptual framework for an AI system for maritime autonomy that can satisfy the COLREGs and learn from human intervention through inductive logic programming.

autonomous decisions in novel or ambiguous scenarios. These human interventions are then used as learning opportunities, feeding into the ILP system to refine and expand its rule set over time.

II. Symbolic Logic: Background Knowledge

In ILP, background knowledge comprises logical facts and programs that support the learning of new hypotheses from provided examples. In the context of maritime autonomy, this background knowledge needs to encompass:

- Geometrical relationships and rates of change, e.g.,
  - Distance, course, bearing and heading, speed, closing speed, bearing rate, etc.
- Waterway and environmental conditions, e.g.,
  - Water depth, shipping channels, sea state, etc.
- Procedures and duties governed by maritime law;
- Principles of good seamanship.

Such information can be sourced from textbooks, handbooks, and other authoritative references. To apply ILP effectively, this knowledge must be encoded using symbolic logic.

1) Geometry: For autonomous agents, expressing geometric relationships with other entities and between those entities is essential for situational awareness. These parameters are crucial for assessing risk, predicting future events, such as the closest point of approach (CPA), and ensuring compliance with maritime law and good seamanship.

We begin by encoding concepts related to static bearings. Since symbolic logic requires information to be expressed as symbolic predicates, it is beneficial to discretise continuous numerical quantities to constrain the search space. Accordingly, we divide relative bearings into 32 sectors [15], each spanning 11.25 deg, as illustrated in Figure 2(b). These sectors are indexed from $\texttt{S} = 0$ (covering $354.375\,\mathrm{deg}$ to $5.625\,\mathrm{deg}$) to $\texttt{S} = 31$ ($343.125\,\mathrm{deg}$ to $354.375\,\mathrm{deg}$), and represented as unique predicates `s0` to `s31`, as given in Figure 3(a). The predicate

```
sector(X,Y,S)
```

expresses that vessel `Y` is in the integer sector `S` relative to vessel `X`. We also define broader angular regions corresponding to important directional zones (e.g., "on the starboard bow" and "astern"), as illustrated in Figure 2(a) and expressed as predicates in Figure 3(b). For example, the predicate `starboard(X,Y)` expresses that the vessel `Y` is on the starboard side of vessel `X` and `bow(X,Y)` expresses that vessel `Y` is on bow of vessel `X`.

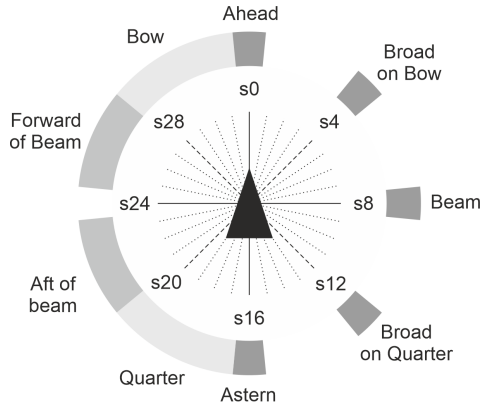
Quantising distances and rates of change is more challenging due to lack of a universally accepted system. Moreover, representing the evolution of these parameters within dynamic scenarios and linking them to agent actions adds further complexity. In the meantime, we have defined a predicate

```
risk_collision(X,Y)
```

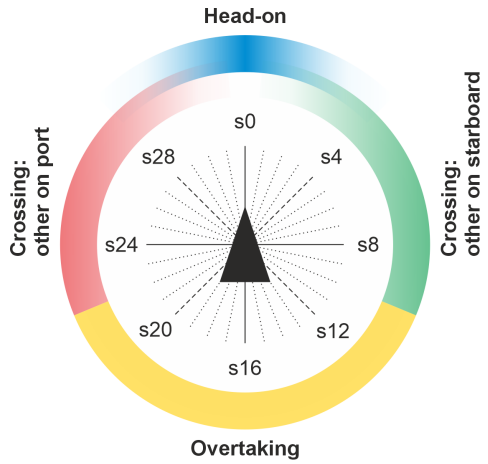
that expresses the risk of collision between two vessels, `X` and `Y`. However, its implementation is non-trivial. Notably, a proposal to use only CPA as a risk metric was rejected during the 1972 convention [16] indicating the need for more nuanced, learnable rules. These challenges will be addressed in future work.

2) Collision Regulations (COLREGs): We have encoded logic for the COLREGs under nominal conditions:

- Encounter rules
  - **Rule 13**: Overtaking situations;
  - **Rule 14**: Head-on situations;
  - **Rule 15**: Crossing situations.
- Duty rules



(a)



(b)

Fig. 2: (a) Discretisation of relative bearings into 32 sectors, following with the nautical point system and their correspondence with zones of nautical importance. (b) Regions of applicability for COLREG rules 13 (overtaking), 14 (head-on), and 15 (crossing). Note the lack of specificity for rule 14.

- **Rule 16**: Give-way vessel;
- **Rule 17**: Stand-on vessel.

These are given in Figure 3(c). The predicate

```
applies(X,Y,R)
```

expresses that the rule `R` applies between vessels `X` and `Y`, where `R` is from a set of rules:

```
R ∈ { rule13_overtaking,
      rule14_headon,
      rule15_crossing,
      rule16_giveway,
      rule17_standon,
      ... }.
```

However, many exceptions to these rules remain unencoded and will be learned during the course of this work. Some exceptions are not explicitly stated in the COLREGs but are established through case law. As these exceptions are

```
s0(S)  :- S=0.  % 354.375 -   5.625 deg
s1(S)  :- S=1.  %   5.625 -  16.875 deg
s2(S)  :- S=2.  %  16.875 -  28.125 deg
s3(S)  :- S=3.  %  28.125 -  39.375 deg
s4(S)  :- S=4.  %  39.375 -  50.625 deg
s5(S)  :- S=5.  %  50.625 -  61.875 deg
s6(S)  :- S=6.  %  61.875 -  73.125 deg
s7(S)  :- S=7.  %  73.125 -  84.375 deg
s8(S)  :- S=8.  %  84.375 -  95.625 deg
          .
          .
          .
s29(S) :- S=29. % 320.625 - 331.875 deg
s30(S) :- S=30. % 331.875 - 343.125 deg
s31(S) :- S=31. % 343.125 - 354.375 deg
```

(a)

```
starboard(X,Y) :- sector(X,Y,S), S>=1, S=<15.
port(X,Y)      :- sector(X,Y,S), S>=17, S=<31.
forward(X,Y)   :- sector(X,Y,S), S>=0, S=<7.
forward(X,Y)   :- sector(X,Y,S), S>=25, S=<31.
aft(X,Y)       :- sector(X,Y,S), S>=9, S=<23.

bow(X,Y)           :- sector(X,Y,S), S>=1, S=<4.
bow(X,Y)           :- sector(X,Y,S), S>=28, S=<31.
forward_beam(X,Y)  :- sector(X,Y,S), S>=5, S=<7.
forward_beam(X,Y)  :- sector(X,Y,S), S>=25, S=<27.
aft_beam(X,Y)      :- sector(X,Y,S), S>=9, S=<11.
aft_beam(X,Y)      :- sector(X,Y,S), S>=21, S=<23.
quarter(X,Y)       :- sector(X,Y,S), S>=12, S=<15.
quarter(X,Y)       :- sector(X,Y,S), S>=17, S=<20.

ahead(X,Y)         :- sector(X,Y,S), S=0.
astern(X,Y)        :- sector(X,Y,S), S=16.
beam(X,Y)          :- sector(X,Y,S), S=8.
beam(X,Y)          :- sector(X,Y,S), S=24.
broad(X,Y)         :- sector(X,Y,S), S=4.
broad(X,Y)         :- sector(X,Y,S), S=12.
broad(X,Y)         :- sector(X,Y,S), S=20.
broad(X,Y)         :- sector(X,Y,S), S=28.
```

(b)

```
% Encounter Rules: 13, 14, and 15

applies(X,Y,rule13_overtaking) :-
    risk_collision(Y,X), sector(Y,X,S), S>=10, S=<22.

applies(X,Y,rule14_headon) :-
    risk_collision(X,Y),
    sector(X,Y,S1), S1=0, sector(Y,X,S2), S2=0.

applies(X,Y,rule15_crossing) :-
    risk_collision(X,Y),
    not(applies(X,Y,rule13_overtaking)),
    not(applies(X,Y,rule14_headon)).

% Duty Rules: 13, 14, and 15

applies(X,Y,rule16_giveway) :-
    applies(X,Y,rule13_overtaking), aft(X,Y).
applies(X,Y,rule16_giveway) :-
    applies(X,Y,rule14_crossing).
applies(X,Y,rule16_giveway) :-
    applies(X,Y,rule15_crossing), starboard(X,Y).

applies(X,Y,rule17_standon) :-
    applies(X,Y,rule13_overtaking), aft(Y,X).
applies(X,Y,rule17_standon) :-
    applies(X,Y,rule15_crossing), port(X,Y).
```

(c)

Fig. 3: Background knowledge relating to (a) discretised bearings, (b) important angular zones; and (c) COLREG Rules 13-17.

encountered and assimilated into the background knowledge it is expected that they will enhance the robustness and safety of the autonomous decision-making over time.

III. Symbolic Machine Learning: Inductive Logic Programming

Inductive logic programming (ILP) is a form of machine learning that learns a set of logical rules from data using first order logic [17]. This generates a hypothesis that aims to explain positive examples without contradicting negative ones, using the provided background knowledge. One of the methods by which it does this is known as inverse entailment [18] and works by inverting the traditional deductive reasoning process.

Inverse entailment uses background knowledge and examples to induce a hypothesis from the examples. Given an example, it constructs a bottom clause, which is the most specific clause that logically entails the example and background knowledge. The process is then to generalise from this constraint to identify a hypothesis that is a logical subset of the bottom clause, which also covers all the positive examples and remains consistent with the negative examples. This process enables a more efficient search of the space of all hypotheses and allows the method to learn complex rules from a relatively limited number of examples, known as few-shot learning.

PyGol[1] [19] is an ILP system based on the concept of meta inverse entailment (MIE) [20], which combines top-down [21] and bottom-up [18] approaches to learning. The hypothesis space is bounded by a bottom clause and a meta theory, both of which can be generated automatically from the background knowledge, making PyGol distinct from most other ILP systems. In traditional ILP frameworks, such as Metagol [21] or Popper [22], users must explicitly define language biases, including modes, meta-rules, or types. In contrast, PyGol eliminates this requirement by inferring them automatically from background knowledge.

Like other similar ILP systems, PyGol supports numerical learning, noise tolerance, recursion, and predicate invention. However, its key advantage lies in its ability to handle noise more effectively. One of the most challenging aspects in top-down systems such as Metagol or Popper is dealing with numerical values/constants in hypotheses. These systems typically address this challenge using meta-rules or other specialised numerical reasoning techniques. Numerical reasoning enables deriving conclusions from numerical data, with examples including Lazy Evaluation [23] in Aleph [24], NumSynth [25] in Popper [22], and NumLog [26], which uses a probability-distribution-based method. In PyGol, this process is more straightforward, as it leverages the bottom clause and a binning approach [27] to learn numerical predicates. This design makes PyGol easier to use in domains that involve mixed symbolic and numerical reasoning.

[1]Available: https://github.com/danyvarghese/PyGol

IV. Results and Discussion

Two machine learning examples were explored in this preliminary study. The first involves a straightforward task with a known solution: identifying the conditions under which COLREG Rule 13 applies in nominal overtaking situations. Although the rule is well-defined in nominal situations, this example serves as a benchmark to validate the learning process. The same methodology can be extended to uncover exceptions to rule applicability, such as the more ambiguous head-on encounters governed by Rule 14. It can also be used to prioritise conflicting rules in multi-vessel scenarios, and to address complex situations involving specific environmental or operational conditions, and ultimately to guide the actions of autonomous agents. The second example focuses on learning rule prioritisation using examples and decisions derived from maritime case law. Specifically, it explores exceptions to Rule 15, using scenarios inspired by the Thomas Everett v Esso Chittagong case.

A. COLREG Rule 13 (Overtaking)

The geometric conditions for the applicability of Rule 13 are clearly defined in the COLREGs. They state that a vessel is being *overtaken* when the other is more than $22.5\,\mathrm{deg}$ abaft of the beam. This corresponds to sectors 10 through 22 in our discretised bearing model and makes it straightforward to manually encode the logic for this rule, as shown in Figure 3(c) and reproduced in Figure 6(b). Here, we demonstrate that PyGol ILP can approximate this rule by learning from a small set of examples, thereby validating the learning framework and establishing a foundation for more complex rule discovery.

Five examples were provided to PyGol. Two of these were positive with a risk of collision and with the agent vessel within the overtaking arc of $112.5\,\mathrm{deg}$ to $247.5\,\mathrm{deg}$:

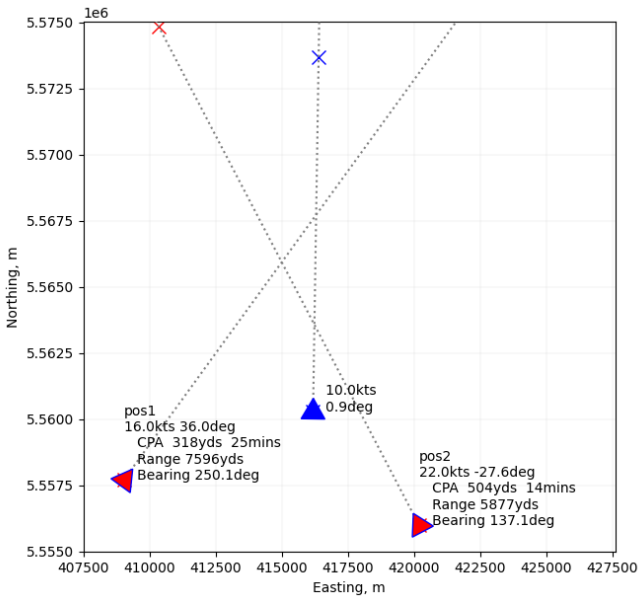
- **Positive Example 1**: agent is on the port quarter of the other vessel, *just inside* the overtaking arc (sector 22)
- **Positive Example 2**: agent is broad on the starboard quarter, *well inside* the overtaking arc (sector 12)

Three of these were negative:

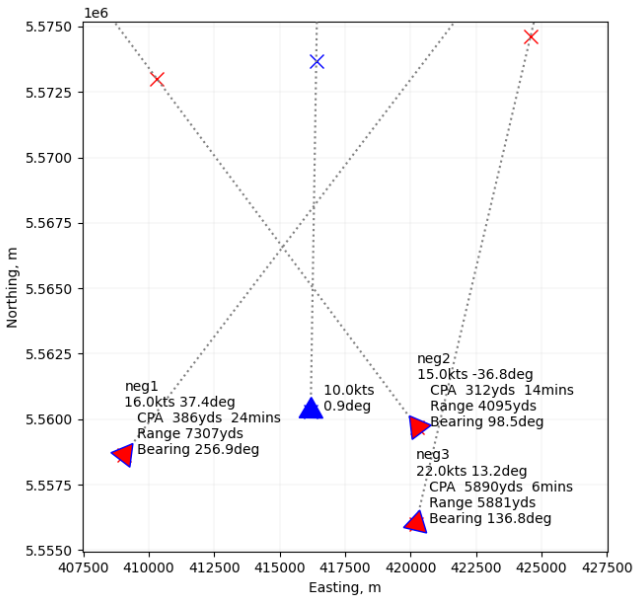
- **Negative Example 1**: agent is on the port quarter of the other vessel, *just outside* the overtaking arc (sector 23)
- **Negative Example 2**: agent is on the startboard quarter, *just outside* the overtaking arc (sector 9)
- **Negative Example 3**: agent is broad on the starboard quarter but there is no risk of collision

The geometry for these examples is illustrated in Figure 4(a,b) and encoded in symbolic logic in Figure 4(c,d).

PyGol begins by generating a set of bottom clauses, followed by a hypothesis set from which it selects the optimal solution. The bottom clauses are logic programs that capture all predicates that are true for each example, effectively overfitting the data. However, they serve a critical role by bounding the space of possible solutions. The bottom clauses used for learning Rule 13 are shown in Figure 5(a). To accommodate numerical arguments, the bottom clauses are extended with three additional predicates: `range(A,B,C)` for evaluating



(a) Positive examples



(b) Negative examples

```
% Other is on the port quarter,
% just inside the region for Rule 13
risk_collision(agent_pos1,other_pos1).
sector(other_pos1,agent_pos1,22).
applies(agent_pos1,other_pos1,rule13_overtaking).

% Other is broad on the starboard quarter
risk_collision(agent_pos2,other_pos2).
sector(other_pos2,agent_pos2,12).
applies(agent_pos2,other_pos2,rule13_overtaking).
```

(c) Positive examples

```
% Other is on the port quarter,
% just outside the region for Rule 13
risk_collision(agent_neg1,other_neg1).
sector(other_neg1,agent_neg1,23).
applies(agent_neg1,other_neg1,rule13_overtaking).

% Other is on the starboard quarter,
% just outside the region for Rule 13
risk_collision(agent_neg2,other_neg2).
sector(other_neg2,agent_neg2,9).
applies(agent_neg2,other_neg2,rule13_overtaking).

% Other is broad on the starboard quarter
% without risk of collision
sector(other_neg3,agent_neg3,12).
applies(agent_neg3,other_neg3,rule13_overtaking).
```

(d) Negative examples

Fig. 4: Examples for learning COLREG Rule 13 (overtaking). The left and right columns show (a,c) two positive examples and (b,d) three negative examples, respectively. The top row (a,b) visualises these examples in our basic maritime autonomy simulator. The blue triangle represents the agent vessel and the red triangles are other vessels. The crosses show waypoints in each vessels' navigation plan before taking action to comply with the COLREGs. The bottom row (c,d) expresses the examples in symbolic logic.

whether a value `A` lies within a specified range ($B \leq A \leq C$), and `leq(A,B)` for checking if it is less than or equal to another ($A \leq B$), and `geq(A,B)` for checking if it is greater than or equal to another ($A \geq B$). Using this extended set, PyGol generates a hypothesis set comprising candidate logic programs that are consistent across all positive examples, along with a corresponding set for negative examples. These are illustrated in Figure 5(a). The optimal hypothesis is selected based on minimal size while maximising coverage of positive examples and minimising coverage of negative ones. The selected hypothesis is shown in Figure 6(a).

The logic program learned by PyGol closely matches the manually encoded version in Figure 6(b), with a minor discrepancy. The learned boundary for the starboard side applicability of the rule is $S = 12$, whereas the correct manually encoded boundary is $S = 10$. This difference is consistent with an interpolation between the positive and negative examples that were provided across the boundary. Additional training examples would allow further refinement.

B. Exception to Rule 15 (Crossing): Thomaseverett v Esso Chittagong

Our second example considers a scenario from our companion paper [28], which covers a wider breadth of COLREG rules and focuses on resolution of legal ambiguity. It aims to learn an exception to COLREG Rule 15 for crossing situations involving a vessel constrained by its draught, which involves COLREG Rule 18 for privileged vessels.

```
% Positive Example 1
applies(A,B,rule13_overtaking) :-
    risk_collision(A,B), sector(B,A,22), port(B,A), aft(B,A), aft_beam(B,A)
% Positive Example 2
applies(A,B,rule13_overtaking) :-
    risk_collision(A,B), sector(B,A,12), starboard(B,A), aft(B,A), broad(B,A),
    quarter(B,A)
% Negative Example 1
applies(A,B,rule13_overtaking) :-
    risk_collision(A,B), sector(B,A,23), port(B,A), aft(B,A), aft_beam(B,A)
% Negative Example 2
applies(A,B,rule13_overtaking) :-
    risk_collision(A,B), sector(B,A,23), port(B,A), aft(B,A), aft_beam(B,A)
% Negative Example 3
applies(A,B,rule13_overtaking) :-
    sector(B,A,12), starboard(B,A), aft(B,A), broad(B,A), quarter(B,A)
```

(a) Bottom clauses

```
% Positive Example 1
applies(A,B,rule_13) :- aft_beam(B,A), sector(B,A,C), risk_collision(A,B)
applies(A,B,rule_13) :- risk_collision(A,B), sector(B,A,C), range(C,12,22)
applies(A,B,rule_13) :- port(B,A), sector(B,A,C), risk_collision(A,B)
applies(A,B,rule_13) :- aft_beam(B,A), port(B,A), risk_collision(A,B)
applies(A,B,rule_13) :- aft(B,A), sector(B,A,C), risk_collision(A,B)
applies(A,B,rule_13) :- aft(B,A), aft_beam(B,A), risk_collision(A,B)
applies(A,B,rule_13) :- aft_beam(B,A), sector(B,A,C), range(C,12,22)
applies(A,B,rule_13) :- aft(B,A), port(B,A), risk_collision(A,B)
applies(A,B,rule_13) :- aft_beam(B,A), sector(B,A,C), port(B,A)
applies(A,B,rule_13) :- port(B,A), sector(B,A,C), range(C,12,22)
applies(A,B,rule_13) :- aft(B,A), sector(B,A,C), aft_beam(B,A)
applies(A,B,rule_13) :- aft(B,A), sector(B,A,C), range(C,12,22)
applies(A,B,rule_13) :- aft(B,A), sector(B,A,C), port(B,A)
applies(A,B,rule_13) :- aft(B,A), aft_beam(B,A), port(B,A)
% Positive Example 2
applies(A,B,rule_13) :- risk_collision(A,B), sector(B,A,C), starboard(B,A)
applies(A,B,rule_13) :- risk_collision(A,B), quarter(B,A), starboard(B,A)
applies(A,B,rule_13) :- risk_collision(A,B), sector(B,A,C), quarter(B,A)
applies(A,B,rule_13) :- risk_collision(A,B), broad(B,A), starboard(B,A)
applies(A,B,rule_13) :- risk_collision(A,B), sector(B,A,C), broad(B,A)
applies(A,B,rule_13) :- aft(B,A), risk_collision(A,B), starboard(B,A)
applies(A,B,rule_13) :- risk_collision(A,B), broad(B,A), quarter(B,A)
applies(A,B,rule_13) :- aft(B,A), sector(B,A,C), risk_collision(A,B)
applies(A,B,rule_13) :- starboard(B,A), sector(B,A,C), range(C,12,22)
applies(A,B,rule_13) :- quarter(B,A), sector(B,A,C), starboard(B,A)
applies(A,B,rule_13) :- aft(B,A), risk_collision(A,B), quarter(B,A)
applies(A,B,rule_13) :- quarter(B,A), sector(B,A,C), range(C,12,22)
applies(A,B,rule_13) :- broad(B,A), sector(B,A,C), starboard(B,A)
applies(A,B,rule_13) :- aft(B,A), risk_collision(A,B), broad(B,A)
applies(A,B,rule_13) :- broad(B,A), quarter(B,A), starboard(B,A)
applies(A,B,rule_13) :- broad(B,A), sector(B,A,C), range(C,12,22)
applies(A,B,rule_13) :- aft(B,A), sector(B,A,C), starboard(B,A)
applies(A,B,rule_13) :- broad(B,A), sector(B,A,C), quarter(B,A)
applies(A,B,rule_13) :- aft(B,A), quarter(B,A), starboard(B,A)
applies(A,B,rule_13) :- aft(B,A), sector(B,A,C), range(C,12,22)
applies(A,B,rule_13) :- aft(B,A), sector(B,A,C), quarter(B,A)
applies(A,B,rule_13) :- aft(B,A), broad(B,A), starboard(B,A)
applies(A,B,rule_13) :- broad(B,A), sector(B,A,C), range(C,12,22)
applies(A,B,rule_13) :- aft(B,A), sector(B,A,C), broad(B,A)
applies(A,B,rule_13) :- aft(B,A), broad(B,A), quarter(B,A)
% Negative Example 1
applies(A,B,rule_13) :- aft_beam(B,A), sector(B,A,C), risk_collision(A,B)
applies(A,B,rule_13) :- port(B,A), sector(B,A,C), risk_collision(A,B)
applies(A,B,rule_13) :- aft_beam(B,A), port(B,A), risk_collision(A,B)
applies(A,B,rule_13) :- aft(B,A), sector(B,A,C), risk_collision(A,B)
applies(A,B,rule_13) :- aft(B,A), aft_beam(B,A), risk_collision(A,B)
applies(A,B,rule_13) :- aft(B,A), port(B,A), risk_collision(A,B)
applies(A,B,rule_13) :- aft_beam(B,A), sector(B,A,C), port(B,A)
applies(A,B,rule_13) :- aft(B,A), sector(B,A,C), aft_beam(B,A)
applies(A,B,rule_13) :- aft(B,A), sector(B,A,C), port(B,A)
applies(A,B,rule_13) :- aft(B,A), aft_beam(B,A), port(B,A)
% Negative Example 2
applies(A,B,rule_13) :- aft_beam(B,A), sector(B,A,C), risk_collision(A,B)
applies(A,B,rule_13) :- port(B,A), sector(B,A,C), risk_collision(A,B)
applies(A,B,rule_13) :- aft_beam(B,A), port(B,A), risk_collision(A,B)
applies(A,B,rule_13) :- aft(B,A), sector(B,A,C), risk_collision(A,B)
applies(A,B,rule_13) :- aft(B,A), aft_beam(B,A), risk_collision(A,B)
applies(A,B,rule_13) :- aft(B,A), port(B,A), risk_collision(A,B)
applies(A,B,rule_13) :- aft_beam(B,A), sector(B,A,C), port(B,A)
applies(A,B,rule_13) :- aft(B,A), sector(B,A,C), aft_beam(B,A)
applies(A,B,rule_13) :- aft(B,A), sector(B,A,C), port(B,A)
applies(A,B,rule_13) :- aft(B,A), aft_beam(B,A), port(B,A)
Negative Example 3
applies(A,B,rule_13) :- quarter(B,A), sector(B,A,C), starboard(B,A)
applies(A,B,rule_13) :- starboard(B,A), sector(B,A,C), range(C,12,22)
applies(A,B,rule_13) :- broad(B,A), sector(B,A,C), starboard(B,A)
applies(A,B,rule_13) :- quarter(B,A), sector(B,A,C), range(C,12,22)
applies(A,B,rule_13) :- broad(B,A), quarter(B,A), starboard(B,A)
applies(A,B,rule_13) :- aft(B,A), sector(B,A,C), starboard(B,A)
applies(A,B,rule_13) :- broad(B,A), sector(B,A,C), quarter(B,A)
applies(A,B,rule_13) :- aft(B,A), sector(B,A,C), aft_beam(B,A)
applies(A,B,rule_13) :- aft(B,A), sector(B,A,C), port(B,A)
applies(A,B,rule_13) :- aft(B,A), aft_beam(B,A), port(B,A)
```

(b) Hypothesis space

Fig. 5: Intermediate steps during the ILP learning procedure. The learned logic program is indicated in blue.

```
applies(A,B,rule13_overtaking) :-
   risk_collision(A,B), sector(B,A,C),
   range(C,12,22)
```

(a) Learned using ILP

```
applies(X,Y,rule13_overtaking) :-
    risk_collision(Y,X), sector(Y,X,S),
    S>=10, S=<22.
```

(b) Manually encoded

Fig. 6: Logic programs for Rule 13 (overtaking): (a) learned using ILP from five examples (two positive and three negative) versus (b) manually encoded. There is a small discrepancy of a different decision boundary, indicated in green versus red for the correct and incorrect value.

A new predicate is introduced as the target of our machine learning for this example:

```
priority(X,Y,R1,R2,R).
```

For a vessel pair `X` and `Y`, it evaluates the priority of one rule `R1` over another `R2` for the purpose of resolving ambiguity, yielding a priority rule `R`.

Assessing the special status of a constrained draught is non-trivial (like the risk of collision) and needs to be addressed in future work. However, in the meantime, we have defined an appropriate predicate

```
waterway(X,constrained_draught)
```

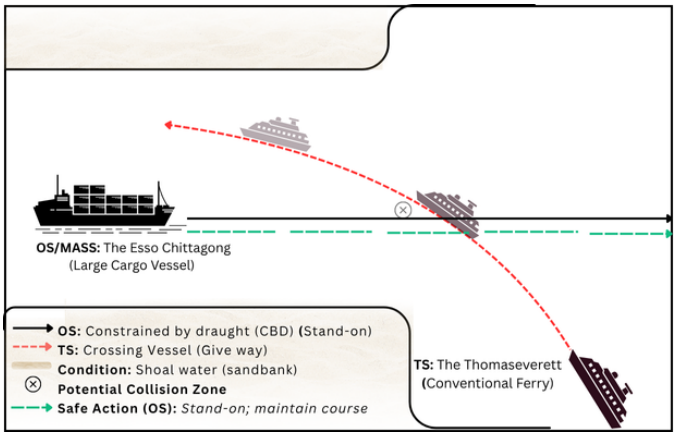
that expresses this. We also introduced `rule18_privilege` to the set of rules.

The training examples for machine learning were inspired by the Thomaseverett v Esso Chittagong case law [29]. The scenario is illustrated in Figure 7(a). The own ship (OS) is the Esso Chittagong and the target ship (TS) is the Thomaseverett. Normally, TS would have the duty to stand on in this crossing situation, with the OS on her port side. However, the OS is in a narrow channel and constrained by her draught. This gives her privilege under COLREG Rule 18 to become the stand-on vessel instead of give-way. This is an exception to the nominal rules. In this legal case, there was a collision because TS followed Rule 17 (stand on) instead of Rule 16 (give way) and the courts found that TS was in violation of Rule 18.

The following examples for learning the exception were modeled on the Thomaseverett v Esso Chittagong scenario, with the agent in place of the Esso Chittagong:

- **Positive Example 1**: agent is *constrained* by draught and the other is broad on the starboard bow (sector 4).
- **Positive Example 2**: agent is *constrained* by draught and the other is 3 points off the starboard bow (sector 2).
- **Negative Example 1**: agent is *unconstrained* and the other is broad on the starboard bow.

These are expressed in symbolic logic in Figure 7(b). The logic program that was learned by PyGol from these examples



(a)

```
vessel(agent_pos1).
vessel(other_pos1).
sector(agent_pos1,other_pos1,4).
risk_collision(agent_pos1,other_pos1).
waterway(agent_pos1,constrained_draught).
priority(agent_pos1,other_pos1,
rule16_giveway,rule17_standon,rule16_standon).

vessel(agent_pos2).
vessel(other_pos2).
sector(agent_pos2,other_pos2,2).
risk_collision(agent_pos2,other_pos2).
waterway(agent_pos2,constrained_draught).
priority(agent_pos2,other_pos2,
rule16_giveway,rule17_standon,rule16_standon).

vessel(agent_neg3).
vessel(other_neg3).
sector(agent_neg1,other_neg1,4).
risk_collision(agent_neg1,other_neg1).
priority(agent_neg1,other_neg1,
rule16_giveway,rule17_standon,rule16_standon).
```

(b) Training examples

```
priority(A,B,rule16_giveway,rule17_standon,rule16_standon)
:-
    starboard(A,B),
    applies(A,B,rule15_crossing),
    applies(A,B,rule18_privilege)
```

(c) Learned program

Fig. 7: ILP learning outcome for the exception to COLREG Rule 15 (crossing) involving a vessel constrained by its draught: (a) scenario inspired by Thomaseverett v Esso Chittagong case law, (b) positive and negative examples, and (c) the learned logic program.

is given in Figure 7(c). It correctly identified the exception to follow Rule 17 (stand on) in the situation where Rule 18 (privilege) applies.

V. Conclusions and Future Work

This work represents a step toward integrating symbolic logic-based learning and reasoning into maritime autonomy systems. By leveraging ILP, we demonstrate the potential to encode and learn maritime law and concepts of good seamanship into interpretable and verifiable logic programs. Our initial experiments show that ILP can approximate known rules from limited examples, providing a foundation for learning more complex behaviours and exceptions over time. However, several significant challenges remain before such systems can be deployed in real-world maritime environments:

- Discretising measures of distance and rate of change in a meaningful and operationally relevant way;
- Incorporating uncertainty, particularly due to noisy or incomplete sensor data.
- Transitioning from static to dynamic scenarios, and encoding temporal evolution of scenarios into symbolic logic;
- Encoding or learning complex concepts, such as risk of collision, vessel status, and intent;
- Learning actions that will satisfy difficult situations involving ambiguity or conflicting duties;
- Handling multi-vessel interactions, including reasoning about relationships between third-party vessels;

Future work will focus on expanding the expressiveness of the background knowledge, refining the learning process with richer examples, and ultimately progressing towards integrating these capabilities into real-time autonomous systems.

Acknowledgment

The authors acknowledge the EPSRC grant on "*Human-machine learning of ambiguities to support safe, effective, and legal decision-making*" (EP/X030156/1). We would also like to thank Daniel Cyrus and James Trewern for their help with the ILP modelling and Edward Clark for his help with the maritime simulator.

References

[1] I. M. Organization, *COLREG: Convention on the International Regulations for Preventing Collisions at Sea, 1972*. IMO, 2003.

[2] S. Jinks, *Reeds Marine Deck 1: Collision Regulations Handbook*, ser. Reeds Marine Deck. Bloomsbury Publishing, 2023. [Online]. Available: https://books.google.co.uk/books?id=WuylEAAAQBAJ

[3] X.-Y. Zhou, J.-J. Huang, F.-W. Wang, Z.-L. Wu, and Z.-J. Liu, "A study of the application barriers to the use of autonomous ships posed by the good seamanship requirement of colregs," *Journal of Navigation*, vol. 73, no. 3, p. 710–725, 2020.

[4] M. Katsivela, "COLREGs and autonomous vessels: Legal and ethical concerns under canadian law," *Maritime Safety and Security law journals*, 2021.

[5] L. Hu, H. Hu, W. Naeem, and Z. Wang, "A review on colregs-compliant navigation of autonomous surface vehicles: From traditional to learning-based approaches," *Journal of Automation and Intelligence*, vol. 1, no. 1, p. 100003, 2022. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S294985542200003X

[6] H. Namgung, "Local route planning for collision avoidance of maritime autonomous surface ships in compliance with colregs rules," *Sustainability*, vol. 14, no. 1, 2022. [Online]. Available: https://www.mdpi.com/2071-1050/14/1/198

[7] A. Bakdi and E. Vanem, "Fullest colregs evaluation using fuzzy logic for collaborative decision-making analysis of autonomous ships in complex situations," *IEEE Transactions on Intelligent Transportation Systems*, vol. 23, no. 10, pp. 18 433–18 445, 2022.

[8] L. Perera, J. Carvalho, and C. Guedes Soares, *Autonomous Guidance and Navigation based on the COLREGs rules and regulations of collision avoidance*. Taylor & Francis Group, London, UK, 12 2009, pp. 205–216.

[9] B. Lesy, A. Anwar, and S. Mercelis, "Evaluating robustness of reinforcement learning algorithms for autonomous shipping," 2024. [Online]. Available: https://arxiv.org/abs/2411.04915

[10] C. Wang, X. Zhang, H. Gao, H. Su, K. Zheng, and W. Wang, "Efficient reinforcement learning for autonomous ship collision avoidance under learning experience reuse," in *2022 IEEE International Conference on Unmanned Systems (ICUS)*, 2022, pp. 1563–1568.

[11] X. Jiang, J. Li, Z. Huang, J. Huang, and R. Li, "Exploring the performance impact of soft constraint integration on reinforcement learning-based autonomous vessel navigation: Experimental insights," *International Journal of Naval Architecture and Ocean Engineering*, vol. 16, p. 100609, 2024. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S2092678224000281

[12] A. N. Madsen and T. E. Kim, "A state-of-the-art review of ai decision transparency for autonomous shipping," *Journal of International Maritime Safety, Environmental Affairs, and Shipping*, vol. 8, no. 1-2, p. 2336751, 2024.

[13] P. Hitzler, A. Eberhart, M. Ebrahimi, M. K. Sarker, and L. Zhou, "Neuro-symbolic approaches in artificial intelligence," *National Science Review*, vol. 9, no. 6, p. nwac035, 03 2022. [Online]. Available: https://doi.org/10.1093/nsr/nwac035

[14] S. Muggleton and L. de Raedt, "Inductive logic programming: Theory and methods," *The Journal of Logic Programming*, vol. 19-20, pp. 629–679, 1994, special Issue: Ten Years of Logic Programming. [Online]. Available: https://www.sciencedirect.com/science/article/pii/0743106694900353

[15] I. Dear and P. Kemp, *The Oxford Companion to Ships and the Sea*, ser. Oxford Companion To... Oxford University Press, 2006. [Online]. Available: https://books.google.co.uk/books?id=HPTwQAAACAAJ

[16] A. Cockcroft and J. Lameijer, *Guide to the Collision Avoidance Rules*. Butterworth-Heinemann, 2003. [Online]. Available: https://books.google.co.uk/books?id=dgK3XYYhjeIC

[17] A. Cropper and S. Dumancic, "Inductive logic programming at 30: a new introduction," *CoRR*, vol. abs/2008.07912, 2020. [Online]. Available: https://arxiv.org/abs/2008.07912

[18] S. Muggleton, "Inverse entailment and progol," vol. 13, no. 3–4, p. 245–286, Dec. 1995. [Online]. Available: https://doi.org/10.1007/BF03037227

[19] D. Varghese, D. Barroso-Bergada, D. A. Bohan, and A. Tamaddoni-Nezhad, "Efficient abductive learning of microbial interactions using meta inverse entailment," in *International Conference on Inductive Logic Programming*. Springer, 2022, pp. 127–141. [Online]. Available: https://link.springer.com/chapter/10.1007/978-3-031-55630-2_10

[20] D. Varghese, "Explainable and efficient machine learning using meta inverse entailment," Ph.D. dissertation, University of Surrey, 2024.

[21] S. H. Muggleton, D. Lin, N. Pahlavi, and A. Tamaddoni-Nezhad, "Meta-interpretive learning: application to grammatical inference," *Mach. Learn.*, vol. 94, no. 1, pp. 25–49, 2014. [Online]. Available: https://doi.org/10.1007/s10994-013-5358-3

[22] A. Cropper and R. Morel, "Learning programs by learning from failures," *Machine Learning*, vol. 110, pp. 801–856, 2021.

[23] A. Srinivasan and R. Camacho, "Experiments in numerical reasoning with inductive logic programming," *Journal of Logic Programming*, 1997.

[24] A. Srinivasan, "The ALEPH manual," *Machine Learning at the Computing Laboratory, Oxford University*, 2001. [Online]. Available: https://www.cs.ox.ac.uk/activities/programinduction/Aleph/aleph.html

[25] C. Hocquette and A. Cropper, "Relational program synthesis with numerical reasoning," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 37, no. 5, 2023, pp. 6425–6433.

[26] D. Cyrus, D. Varghese, R. Bauer, and A. Tamaddoni-Nezhad, "Numerical-symbolic learning from biomedical data," in *2025 IEEE Conference on Artificial Intelligence (CAI)*. IEEE, 2025, pp. 1338–1343.

[27] D. Varghese, R. Bauer, and A. Tamaddoni-Nezhad, "Few-shot learning of diagnostic rules for neurodegenerative diseases using inductive logic programming," in *International Conference on Inductive Logic Programming*. Springer, 2023, pp. 109–123. [Online]. Available: https://link.springer.com/chapter/10.1007/978-3-031-49299-0_8

[28] S. Verma, D. Varghese, A. A. Treloar, A. Tamaddoni-Nezhad, and A. Hunter, "Resolving Legal Ambiguities for Safe MASS Navigation: A Socio-Technical Approach Using Human-Like Computing," *IEEE OCEANS 2025, Great Lakes*, 2025, (submitted).

[29] R. A. Cahill, *Collisions and Their Causes*, 3rd ed. The Nautical Institute, London, UK, 2002.