Formal Verification of COLREG-Based Navigation of Maritime Autonomous Systems

Fatima Shokri-Manninen[1],
Jüri Vain[2], and Marina Waldén[1]

[1] Åbo Akademi University
{fatemeh.shokri,marina.walden}@abo.fi,
[2] Tallinn University of Technology
{juri.vain}@taltech.ee

Abstract. Along with the very actively progressing field of autonomous ground and aerial vehicles, the advent of autonomous vessels has brought up new research and technological problems originating from the specifics of marine navigation. Autonomous ships are expected to navigate safely and avoid collisions following COLREG navigation rules. Trustworthy navigation of autonomous ships presumes applying provably correct navigation algorithms and control strategies. We introduce the notion of maritime game as a special case of Stochastic Priced Timed Game and model the autonomous navigation using UPPAAL STRATEGO. Furthermore, we use the refinement technique to develop a game model in a correct-by-construction manner. The navigation strategy is verified and optimized to achieve the goal to safely reach the manoeuvre target points at a minimum cost. The approach is illustrated with a case study inspired by COLREG Rule 15.

Keywords: Verification · Refinement · Maritime autonomous systems · COLREG rules · Collisions avoidance · Navigation · Safety · Optimization · Game theory · UPPAAL STRATEGO.

1 Introduction

The demand for unmanned ships has risen aiming at reducing operation costs due to minimal crew on board and safety at sea but also promoting remote work. Autonomous ships are expected to make more and more decisions based on their current situation at sea without direct human supervision. This means that an autonomous ship should be able to detect other vessels and make appropriate adjustments to avoid collision by maintaining maritime traffic rules. However, the existence of a 'virtual captain' from the shore control centre (SCC) is still a must to perform critical or difficult operations [2] and there is a need for reconfirmation when inconsistent or corrupted commands are detected by the onboard system.

The connectivity between ships and SCC has to guarantee sufficient communication for sensor monitoring and remote control [10] when SCC intervention is needed. This connectivity also plays an important role for the safety of operations concerning collision avoidance in the remote-controlled scenarios for transforming the data and receiving information regarding the decision from SCC. Sub-second reaction time is, however, not critical regarding safe navigation in the maritime sector as it takes up to minutes for the ship to change its course in case of detection of another ship or an obstacle. In this paper the goal is to model maritime autonomous systems so that the unmanned ships learn a safe and optimal strategy for navigation pursuing collisions avoidance.

One of the most critical safety issues in the development of autonomous vehicles and self-driving cars is their poor performance under adverse weather conditions, such as rain and fog due to sensor failure [15]. However, when modelling maritime specification, we do not take into account sensor inaccuracies and possible transmission errors, since there are standard sensor redundancy design and error correction measures applied on modern vessels to ensure that ships notice each other in a timely manner. For safety assurance, a ship is able to communicate with another ship or shore via VHF radio, satellite services, etc.

For unambiguous navigation protocol, the International Maritime Organization (IMO) [11] published navigation rules to be followed by ships and other vessels at sea which are called Convention On the International Regulations (COLREG).

When developing the autonomous ship navigation system, quality assurance via tool supported model-based control synthesis and verification is of utmost importance. UPPAAL STRATEGO [9] is a branch of the UPPAAL [6] synthesis and verification tool family. It uses machine learning and model checking techniques to synthesize optimal control strategies. Hence, it is a good candidate for control synthesis tool which satisfies above mentioned needs.

In our research, we aim at adapting formal modelling with UPPAAL STRATEGO for verifying and synthesizing safe navigation of autonomous ships. As an additional contribution, we improve the autonomous ships navigation performance regarding its safety and security at the same time planning for optimal route and scheduling maneuvers according to COLREG rules.

2 Related work

There has been a variety of studies on autonomous ship navigation obeying COLREG rules. Among these fuzzy logic [17], interval programming [7], and 2D grid map [22] could be mentioned. However, the previous approaches do not deal with verification for safe navigation. Moreover, (potentially) non-deterministic behaviour of autonomous ships, communication delays, sensor failure and weather conditions are not considered in their models.

Recently, in MAXCMAS project [23], COLREG rules have been implemented in collision avoidance module (CAM) software where collision avoidance decision is generated and action taken as soon as collision risk is detected. In spite of their various simulation tools, verification methods are discussed only implicitly. Furthermore, to the best of our knowledge, our work is the first one that synthesizes a safe and optimal navigation strategy that also takes into account some of the weather conditions.

There is a fair number of publications on autonomous navigation control synthesis methods and tools that rely on various sets of assumptions - for example continuous, discrete and hybrid dynamics, as well as piece-wise linear and non-linear processes [12] [18] [16]. The main issue of the controller synthesis is the scalability of synthesis methods in case of complex control objects. Hierarchical control architectures, e.g. in SCADA are addressing this issue. While low-level control typically should tackle with continuous (often nonlinear) processes the upper control layers deal with the abstract representation which typically describes hybrid or discrete dynamics. In this work, we model the vessels dynamics on a high level of abstraction using discrete state space and continuous time.

Among the tools that are oriented to timed discrete-state models and timed game based control synthesis, UPPAAL STRATEGO has proven its relevance in several case studies, where optimal strategies have been generated using Statistical Model Checking (SMC) and machine learning. Examples include, for instance, adaptive cruise control [16], railway systems [13] and autonomous driving systems [4] [5].

In [16] the authors synthesize a safe control strategy with the goal of maintaining a safe distance between vehicles where one of them is considered to be uncontrollable by the given controller. Railway control systems, are modelled as a Stochastic Priced Timed Game in [13] by using game theory, where a set of trains considered as an environment and lights, points and sections in the railway, are assumed to be controllable. In [5] the authors also model a railway signalling system with autonomously moving blocks as a Stochastic Priced Timed Game, but in addition they consider stochastic delays in the communication. A safe and optimal driving strategy for the model is synthesised in UPPAAL STRATEGO. In [4], on the other hand, SMC has been used for formal modelling uncertainty in autonomous positioning systems. The safety of the position of a tram is proved with the levels of uncertainty and possible hazards induced by onboard satellite positioning equipment.

In our work, we introduce the notion of a maritime game for control synthesis that is based on navigation specification of the ship where weather conditions are integrated. We model the navigation problem as a special case of Stochastic Priced Timed Game with a goal of collisions avoidance between two ships. Furthermore, we use the refinement technique [3] for a stepwise development of the model for avoiding complexity and ambiguity in the modelling.

3 Case study and navigation specification

3.1 Overview of the Case Study

When modelling navigation manoeuvres of autonomous ships, we focus on standard situations, addressed in COLREG. As an example, let us consider a scenario where two ships have intersecting courses as depicted in Figure 1.

In this example, in spite of the existence of remote monitoring from the SCC, we assume also that ships have autonomous navigation capability. According to Rule 15 of COLREG [19]; when two power driven vessels have intersecting courses with the risk of collision, the vessel which has the other on her own starboard (right) side shall keep out of the way and avoid crossing ahead of the other vessel. In this case the vessel giving way should adjust its speed and/or course to pass behind the approaching vessel. The adjustment will therefore be made to the starboard side. In the case depicted in Figure 1, shipB should give way while shipA maintains its direction and speed.

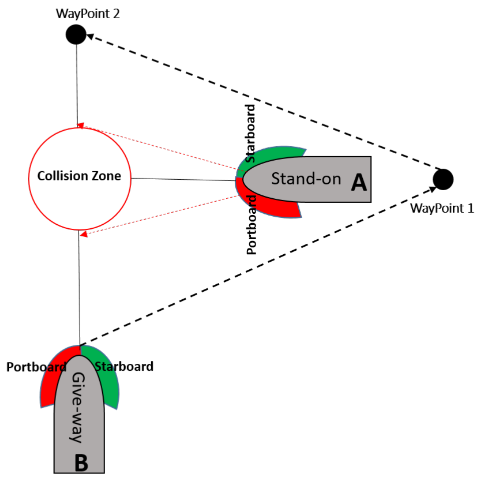

Fig. 1. Autonomous Navigation of Ships

The navigation control of shipB has a choice to slow down instead of altering its path to pass shipA. By doing this, the expected arrival time might not be as late as when following a redirected route. However, if for some reason shipA is slowing down, then the controller should navigate shipB safely to another route through a sequence of waypoints [1] (see Figure 1).

3.2 Ship waypoints (WP) and path plan

For safe navigation of vessels, we consider a set of waypoints along the route which define the routing subgoals and that the ship has to traverse during the maneuver. When a vessel plans the voyage from the current position to its next waypoint position, a course change may occur. In case of rerouting to a waypoint, a new heading should be calculated. Figure 2 shows the heading relationship between the ship and waypoint.

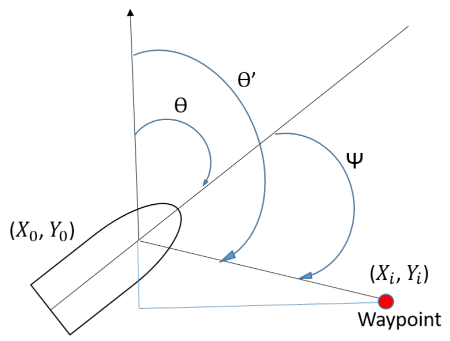

Fig. 2. Heading relation between ship and waypoint

Assume that (X_0, Y_0) is the initial position of the ship and (X_i, Y_i) are the coordinates of the targeted waypoint, the bearing angle ψ of the waypoint from the ship is calculated as follows [1]:

$$\theta' = a\tan 2 \frac{(Y_i - Y_0)}{(X_i - X_0)} \tag{1}$$

$$\psi = \theta' - \theta \tag{2}$$

where θ is the heading of the ship, θ' is the encountering angle of the waypoint from the vertical axis. Here, if the value of ψ becomes negative, then 2π (360) is added to make it positive.

To calculate the position of the ship after altering the course based on the new heading of the ship (ψ), the following calculations should be performed:

$$X = X_0 + V\cos(\psi) \qquad Y = Y_0 + V\sin(\psi) \tag{3}$$

where (X_0, Y_0) is the initial position of the ship, (X, Y) are the coordinates of the next position of the ship and V is the speed of the ship.

In our scenario with two ships assuming that (X_A, Y_A) are the coordinates of shipA and (X_B, Y_B) are the coordinates of shipB, the distance between the two ships is calculated as Euclidean distance. This could require an update of the position of either one or both ships following Equations 1, 2 and 3. After the update, the distance between the ships should be re-calculated to evaluate whether the risk of collision still remains.

3.3 Influence of wind on the ship navigation

When the ship moves in the presence of wind in addition to navigating along the true course (heading), it will also drift as a consequence of wind which is called leeway [24]. Thus, leeway (α) is the angle between the heading (TC) and the drift track (CG). Figure 3 shows the leeway angle with the presence of wind.

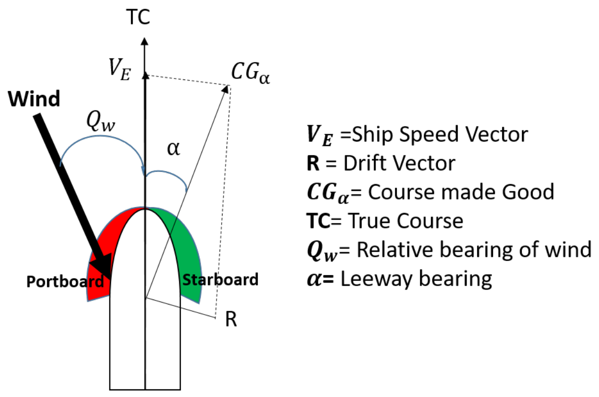

Fig. 3. Leeway angle (α)

If the wind pressure comes from portboard, it deviates the angle of the ship heading to the right then α is positive '+'. In case of the pressure from starboard the α is negative '-'. The leeway is calculated as follows [24].

$$\alpha = K(\frac{V_W}{V_E})^2 \sin Q_w \tag{4}$$

where, V_W is the speed of the wind and V_E is the speed of the ship. In this formula Q_w is the relative bearing of the wind whereas K is the leeway coefficient. After calculating the drift angle based on the wind conditions, the heading of the ship should be corrected periodically.

4 Reinforcement learning and game theory

To analyze the navigation options of autonomous ships, and more importantly, to verify the navigation decisions, especially in combination with the other ships we use game theory as a base formalism for modelling and for optimizing the games.

Reinforcement learning (RL) is commonly used for solving Markov-decision processes, where an agent interacts with the world and collects rewards [21]. RL is a powerful method for learning an optimal strategy for the agent from trial and error. UPPAAL STRATEGO [9] uses RL and model checking by combining the techniques to synthesize optimal control strategies. Any of the moves (modeled as transitions) which the agent chooses in its current state, incurs a cost, e.g. fuel consumption (fc). In Figure 1, the first time derivative of fc is denoted fc'. In this setting, the priced timed game [8] defines an infinite-state Markov (reward) decision process with the goal of finding a strategy that will minimize the cost to reach the goal state.

Reinforcement learning can be considered in the context of a game, often for one player (agent) or as a two-player game in which the other player is typically

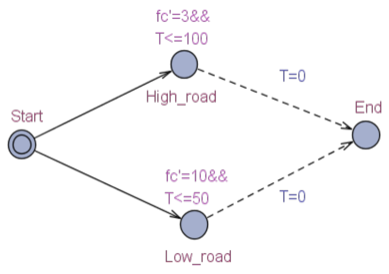

Fig. 4. An example of a timed game automaton in the UPPAAL STRATEGO

the environment. The agent playing against the environment tries to learn a policy that maximizes the agent's reward in the environment.

Our ship navigation control model is based on a game G = (S, E), consisting of a set S of states and a set E of directed edges between those states, the state transitions. Since there are two players in the considered game, where the agent (player) plays against the environment (Agent *vs.* Env), we also have two types of transitions, controllable and uncontrollable. As can be seen from the example in Figure 4, the driver as the playing agent can choose one of the two alternative roads which are represented as controllable transitions (indicated by the solid line). However, the actual travel-time depends on the intensity of traffic (environment) on a particular day, which is reflected as the upper time-bounds in the invariants of locations *High_road* and *Low_road*. Since the outgoing edges from these locations are uncontrollable (indicated by the dashed line), leaving these locations may occur latest at these time-bounds. Such a two-player (possibly antagonistic) game is represented by a tuple G= (S, $\rightarrow$, $\dashrightarrow$, s_0, Bad, $Goal$) where [13],

- $\rightarrow \subseteq S \times S$: set of controllable transitions (Player).
- $\dashrightarrow \subseteq S \times S$: set of uncontrollable transitions (Environment).
- $s_0 \in S$: initial state.
- $Bad \subseteq S$: set of states where player loses the game.
- $Goal \subseteq S$: set of states where player wins the game.
  We assume that sets Bad and $Goal$ do not intersect.

In game theory, a run is a finite or infinite sequence of states r= (s_0, s_1, s_2, . . .). In case of a finite sequence of states, the player reaches its terminal state that can be a goal state that is a winning state, or a bad state that corresponds to the winning state of the adversary. We call a run safe, if there is no bad state ($s \in Bad$) in the run.

The player's strategy is a complete algorithm for playing the game, telling what move a player should do for every possible situation throughout the game. In a state transition system setting a player decides depending on its current state ($s \in S$) and strategy σ which transition to execute next. Formally, a strategy for the player is a mapping,

$$\sigma : S \to S \text{ such that } \forall r \in \rho, r.s \in S : (r.s, \sigma(r.s)) \in E,$$

where ρ is a set of runs, such that from any state s of a run r in ρ, strategy σ chooses the next state reachable by an outgoing edge of s.

A strategy is called safe if in the run, any of the outgoing transitions in the state ($s \in S$) does not lead to bad states:

$$\sigma_{safe} = \{(s_i, s_j) \mid s_i, s_j \in S \wedge (s_i, s_j) \in E \wedge s_j \notin Bad\}.$$

A strategy is feasible if it is a safe strategy and reaches the goal state ($s \in Goal$) in the run:

$$\sigma_{feasible} = \{(s_i, s_j) \mid (s_i, s_j) \in \sigma_{safe} \wedge (\exists s : s \in S \wedge s = s_j \wedge s \in Goal)\}.$$

Similarly we define a feasible run. A run r is feasible if it is finite of length $|r|$, safe and reaches a goal state ($s \in Goal$):

$$\begin{aligned} r_{feasible} = \{r \mid (\forall s_i : s_i \in r, i \in [1, |r|-1] : s_i \notin Bad) \wedge \\ (\exists s_j : s_j \in r \wedge j = |r| \wedge s_j \in Goal)\}. \end{aligned}$$

A run is called optimal, if it is feasible and reaches the goal state ($s \in Goal$) in the run with a minimum cost:

$$r_{optimal} = \{r \mid r \in r_{feasible} \wedge cost(r) \le min(ran(cost))\}.$$

where cost function $cost : \rho \to \mathbb{R}$ assigns a real-valued number to each run.

A winning strategy is optimal, if it is a safe and feasible strategy and there is a run ending up in goal state ($Goal$) with a minimum cost:

$$\sigma_{optimal} = \{(s_i, s_j) \mid (s_i, s_j) \in \sigma_{feasible} \wedge (\exists r : r \in r_{optimal} \wedge s_i \in r \wedge s_j \in r)\}.$$

5 Maritime Game

To formally model the navigation problem, we formalise it in the game tuple G_M=(S, $\to$, $\dashrightarrow$, s_0, Bad, $Goal$) where,

- *S:* is a set of states of the ships. To grant the decidability of G_M, we consider S as a finite set.
  In principle, S consists of as many partitions as there are ships involved in the game. For the two ships in the paper we consider $S = S_A \times S_B$, where S_A denotes the state component of shipA and S_B of shipB. Both S_A and S_B are composed of the same set of variables and the state vector of a ship has the following structure:

$$< P, WP, H, HS, Vel, WW, WS >$$

  where the state variables and their domains are defined as follows:

- $P \in \{(x, y) | x \in X \wedge y \in Y\}$: Position of the ship in the 2D coordinate system. The position will be updated periodically based on the heading and speed of the ship.
- $WP \in \{(x_{wp0}, y_{wp0}), (x_{wp1}, y_{wp1}), \ldots, (x_{wpn}, y_{wpn})\}$: Ordered set of waypoints of the ship in the coordinate system. The next waypoint for a ship may change in case the ship changes course.
- $H \in [0, 360]$: Heading of the ship in the degree interval. Heading of the vessel will be updated in case of rerouting.
- $HS \in \{initial, deviated\}$: Heading Status of the ship. Before changing course, the value of the heading status is *initial*. As soon as the heading becomes updated due to navigation to the waypoint, it will be assigned to *deviated*. If the heading of the ship is *deviated*, the ship needs to do a rerouting to go back to the initial path and original heading.
- $Vel \in [0, 10]$: Velocity of the vessel in the normalized integer interval. During navigation at sea, vessels can accelerate, decelerate or continue its voyage with the same speed. The discrete interval $[0, 10]$ is a reasonable approximation for the speed, since we mainly consider big cargo ships with a cruising speed of about 12 knots. Moreover, the approximated speed has always been rounded up to the worst case.

In addition to the above state variables, we introduce new variables used in the model refinement.

- ◦ (Refinement) $WW \in BOOL$: Windy Weather condition in boolean expressions. If Windy Weather has the value true, it means that weather is windy. It is introduced in the refinement step, we introduce environmental condition which affects the vessel course. We consider that the heading of the ship is drifted due to the wind pressure.
- ◦ (Refinement) $WS \in V_W \times Q_w$: Wind Specification of vessels. For calculation of drift angle by Equation 4 when the wind pressure is present, we need to know the wind speed (V_W) and angle of the wind (Q_w) that comes to the vessel.

Thus, at each time instant the state variables of the ships' state vectors acquire one value from their domain as determined by the transition relations, either $\rightarrow$ or $\dashrightarrow$. We use the same navigation specifications for both ships in the game, with the difference that shipA does not contain the controllable navigation variables WP and HS that shipB has or weather features WW and WS that affect the vessel. This is because we are interested in capturing the behaviour of shipB as an agent under different circumstances.

– $\rightarrow$, $\dashrightarrow$*:* The transition relations are defined as $\rightarrow \subseteq S \times G(V) \times Act \times S$ and $\dashrightarrow \subseteq (S \times G(V) \times Pr \times Act \times S)$ where,
  - $G(V)$: is the set of constraints in guards and V denotes the set of integer and boolean variables.
  - *Act*: is a sequence of assignment actions with integer and boolean expressions. According to player preference, one of the enabled transitions will be chosen in the current state. We define four functions that define the effect of player actions. The functions for player transitions are as follows:

  - ∘ *Update_Heading*: $(P, WP) \rightarrow H$ is a function that calculates a angle for the ship's heading angle H from the ship's current position P and its next waypoint WP.
  - ∘ *Update_Speed*: $(Vel, [-2, 2]) \rightarrow Vel$ is a function that assigns a new velocity to the vessel based on the current velocity Vel and a value in the interval $[-2, 2]$. This interval indicates the acceleration/deceleration of 0-2 speed units.
  - ∘ *Update_Position* : $(H, Vel, P) \rightarrow P$ is a function that gives a new position for the ship given its previous position P, heading H and velocity Vel.
  - ∘ (Refinement) *Leeway_angle*: $(WS, Vel) \rightarrow H$ is a function that assigns a calculated drift angle for the ships's heading H, depending on the wind specification WS and velocity Vel.
- • Pr: denotes the set of integer-valued probabilities.

In this two-player game, we model shipB with its controller as a player. As a consequence, all transition for this player are considered to be controllable. However, we assume shipA as an Environment for shipB with the same functionality except that shipA has some stochastic transitions with probability (Pr) in addition to normal transitions.

– $s_0 \in s_{B0} \times s_{A0}$: is initialized with random speed, initial position and heading of ships. For analyzing concrete incidents these initial values could be based on values from existing datasets.

– *Bad:* is the state where two ships get to collision zone (see Figure 1).

– *Goal:* is the state that is reached when two ships have passed each other within a safe distance.

6 Model development in UPPAAL STRATEGO

We model the navigation problem[1] as a Stochastic Priced Timed Game using the tool UPPAAL STRATEGO where the controller of shipB should dynamically plan its maneuver, while the opponent (shipA) moving according to its preset trajectory forces shipB to change its route. In this game, we define the fuel consumption (fc) as a the price to be minimized under the safe strategy. The change in velocity of the ship is directly related to fc, so that the consumption of fuel increases if the ship slows down and speeds up again rather than changes the route, causing the price to increase.

The goal is that the ships move to their target positions in a safe way (without the risk of a collision) while at the same time optimizing the fuel consumption.

To avoid ambiguity, we use refinement [3], which enables the system to be created in a stepwise manner gradually adding details into the model and proving that each refinement step preserves the correctness of the previous steps. Models can be refined either via *superposition refinement* [14] [20], where new features are added to the automaton, or by *data refinement*, where abstract features are replaced by more concrete ones.

[1] The game model is found in: https://github.com/fshokri/Game-model

The current refinement process of the model consists of one abstract model with one refinement step using both superposition and data refinement. The abstract model presents the general view of safe navigation with given waypoints. In the refinement step, we introduce weather conditions windy or clear for the ship navigation. We assume that strong wind from shipB starboard increases fuel consumption when turning right. The two step timed automata model introduced here should be seen as a modeling step towards an implementation.

6.1 An Abstract Model Of Autonomous Ships

For synchronizing the state transitions of the two ships, shipB and shipA the scheduler template is created with two channels for each ship; *ShipBgo*!, *ShipAgo*! as well as *Update_B*! and *Update_A*! (see Figure 5). The first channel enables each ship to move while the second one updates their positions after moving action. We define two functions in this automaton namely *initialize*() and *UpdateDistance*(). The former initializes two ships with initial headings and positions in the coordinate system. The latter calculates the distance between two ships after movement. If the distance becomes smaller than the defined safe distance, it means that the ship collides with another ship and the game is over. For restarting the game, we add one transition from state *End* to *Init*. In this template, we also add exponential rates where the process may be delayed an arbitrary long time in the worst case. The user-supplied rate (here 5) has been chosen as an expert estimate for unbounded delays but it can be tuned according to a particular situation.

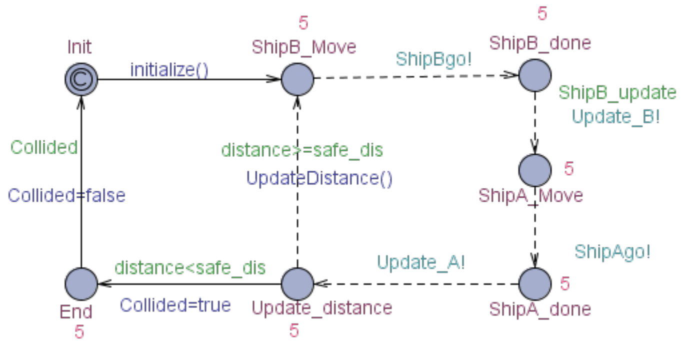

Fig. 5. The Scheduler in UPPAAL STRATEGO

In the abstract model the behaviour of shipB is depicted in Figure 6 where the model is divided into ship (upper) and controller (lower) automata templates. In the shipB template, we model the different states of a ship that are reachable from the initial state (*Moving*), while in the controller we only consider events for giving permissions to take actions. Since shipB is the player in this game, all transitions for shipB and its controller are considered to be controllable.

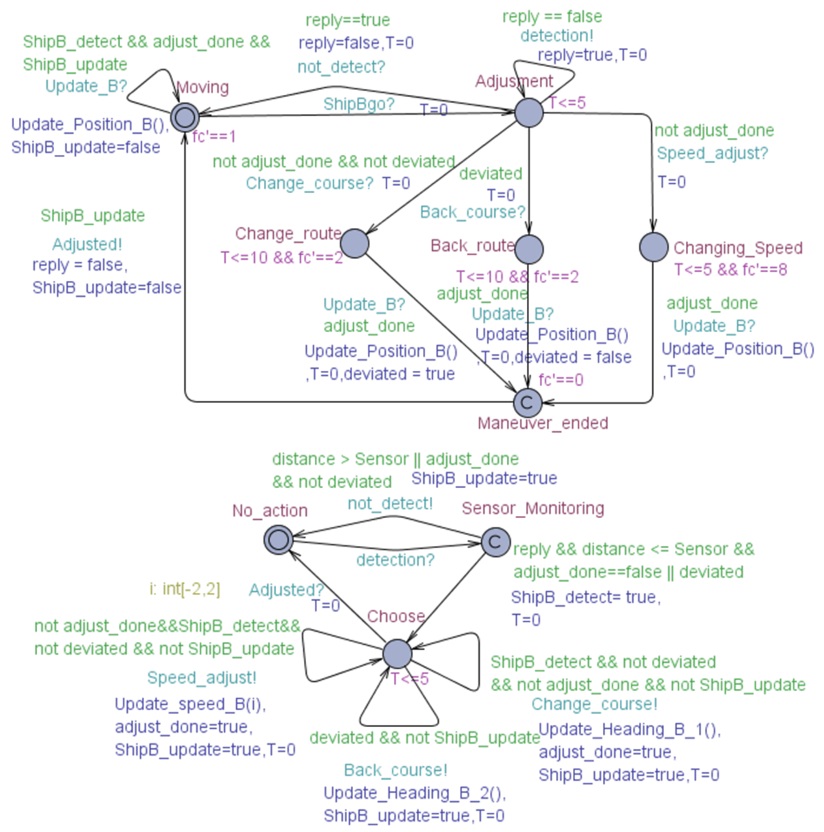

Fig. 6. The Abstract Model of ShipB (upper) and ShipB Controller (lower)

In Figure 6, shipB starts its move in state *Moving* if it gets permission via *ShipBgo*?. If shipB already did adjustment after detection, it periodically updates its position via the self-loop edge of state *Moving* that is synchronized with the scheduler via channel *Update_B*? (see Figure 5). In state *Adjustment* shipB sends the request to controller about detection (*detection*!) (see Figure 6). If the controller detects a ship by its sensor ($distance <= sensor$) then it will non-deterministically change speed or change course. In case of rerouting, the new heading will be calculated by functions *Update_Heading_B*()_1 and *Update_Heading_B*()_2 in the controller, and the new position of the ship will be updated in shipB by *Update_Position_B*(). Note that the implementation of the two functions is the same, the only difference is that the function *Update_Heading_B*()_1 uses *waypoint*1 for calculating the new heading, while *waypoint*2 is used for the function *Update_Heading_B*()_2. Since the UPPAAL tool has limited support for simulation of double values, we convert the integer values to ones of double type by multiplying them by 1.0 (see Figure 7) in the arc tangent formula in the function *Update_Heading_B*()_1.

When shipB moves to the maneuver waypoint, it deviates from its original path and the heading status becomes *deviated*. If the heading of the ship is *deviated*, the ship needs to go back to its original path ($waypoint2$) by moving to state $Back\_route$ after the collision risk has been removed.

As can be seen from Figure 6, we add the continuous variable (fc') as a hybrid clock in the invariant of the states having non zero duration, i.e. states $Change\_route$, $Back\_route$ and $Changing\_Speed$, to show how much fuel the ship consumes for adjustment. As the value for the states $Change\_route$ and $Back\_route$ are smaller than state $Changing\_Speed$, they will consume less fuel. Function $Update\_Heading\_B$ uses Equation 1 and 2 to calculate the new heading whereas we use Equation 3 in function $Update\_Position\_B()$. Figure 7 shows the implementation of function $Update\_Heading\_B_1$ according to the Equation 1 and 2. While Figure 8 shows the updating of the position of the ship according to Equation 3 (left) and calculating the Euclidean distance between two ships (right).

```
// Changing route needs to update the heading
void Update_Heading_B()_1{
    double degree
    int deg, newHeading;
    degree= atan2(Wpy*1.0-Pos_y_B*1.0, Wpx*1.0-Pos_x_B*1.0);
   deg = fint(degree);
    newHeading= deg-headingB;
    if (newHeading >= 0)
      headingB = newHeading;
    else
      headingB = newHeading + 360;}
```

Fig. 7. C function for updating the heading of the ship

```
void Update_Position_B(){
 int x, y;
 x=fint(round(cos(headingB)));
 y=fint(round(sin(headingB)));
 Pos_x_B= x* SpeedB +Pos_x_B;
 Pos_y_B= y * SpeedB +Pos_y_B;}
```

```
// calculating distance between two ships
void UpdateDistance{
double diffx, diffy;
diffx = pow( Pos_x_A - Pos_x_B, 2);
 diffy = pow(Pos_y_A -Pos_y_B, 2);
  distance = fint(sqrt( diffx + diffy}
```

Fig. 8. C function for updating the position (left) and the distance (right) of the ship

We model shipA as an environment for shipB. For this reason, all transitions in shipA automaton are uncontrollable. We follow the same structure as above for the shipA template (see Figure 9), except that in shipA both the ship template and its controller template are integrated to one. Moreover, we consider a stochastic behaviour for shipA. According to COLREG, shipA should maintain its direction and speed. For this reason, shipA keeps moving straight on with the probability *weight* $= 8$ to indicate that this should happen with a high probability. The parameters for these probability distributions are defined from common practice.

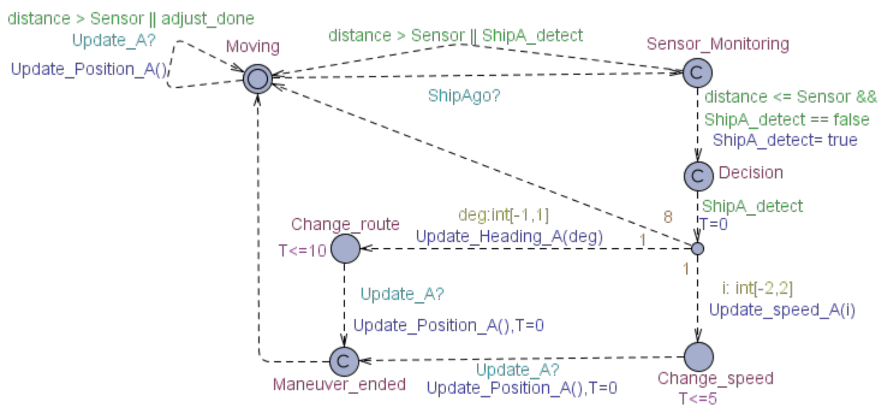

Fig. 9. The Abstract Model Of ShipA in UPPAAL STRATEGO

6.2 Introducing weather conditions

In the refinement step, we consider the impact of weather conditions on shipB navigation using the boolean variable *windy* which is non-deterministically assigned a value true or false in the scheduler template. For shipB, we add two new states *Change_route_drift* and *Back_route_drift* where the rerouting includes the drifting due to windy weather (see Figures 10). We assume that the fuel consumption for these states ($fc' == 5$) is greater than in clear weather ($fc' == 2$) as strong wind from shipB starboard increases fuel consumption when turning right. Changing course even in the windy weather is a better choice than changing speed due to less fuel consumption. State *Changing_Speed* remains unchanged in windy weather, since wind cannot considerably change the speed of the ship [24].

The correction of the ship heading under windy circumstances is performed by functions $Update\_HeadB\_wind1()$ and $Update\_HeadB\_wind2()$ in the controller by updating the heading of the ship from the new calculated drift angle by function $leeway\_angle()$ (Equation 4). The difference between updating heading functions in windy weather ($Update\_HeadB\_wind1()$, $Update\_HeadB\_wind2()$) and without wind ($Update\_Heading\_B()\_1$, $Update\_Heading\_B()\_2$) (see Figure 7) is only that leeway angle is added to the new heading of ship.

$$newHeading = deg - headingB + leeway;$$

For the controller template, we add two new transitions in the *Choose* state regarding windy weather where drift angle will be considered in case of changing course.

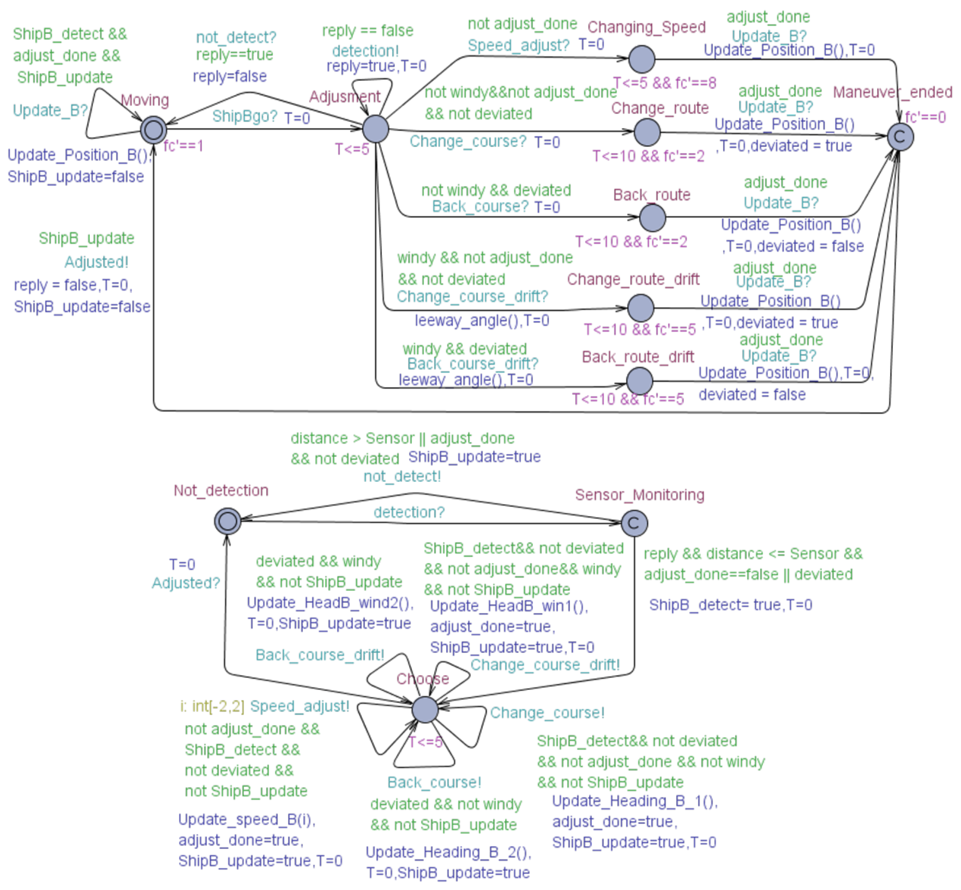

Fig. 10. The Refined Model Of ShipB and Controller in UPPAAL STRATEGO

The weather conditions are superimposed as a new feature to the abstract model by strengthening the guards in the model and adding assignments to the variables of the new feature. The variables of the heading are data refined to take the leeway into accout, but otherwise the behaviour of the model remains the same.

6.3 Verification and Validation

UPPAAL STRATEGO provides an extended query language, where strategies can be verified and optimized (by reinforcement learning) for stochastic priced timed games. The constructed strategies can be used as constraints in performing SMC of a game (like Query 3 and 4 in Table 1). Table 1 presents verifed Queries 1 to 4.

The main objective of the maritime game model is to synthesise a safe strategy such that shipB never collide with shipA. Furthermore, ShipB should choose the action in case of collision avoidance that consumes less fuel in the windy weather or without wind just under the safe circumstance. To satisfy requirements, a safe and optimal strategy needs to be synthesised.

Table 1. UPPAAL STRATEGO Queries

Id	Query	Result
1	strategy Safe= control: A[] not Scheduler_refined.End	Satisfied
2	strategy OptSafe = minE(fc)[<=200]: <> ShipB_refined.Maneuver_ended under Safe	Satisfied
3	Pr[<=200](<> ShipB_refined.Change_route) under OptSafe >= Pr[<=200](<> ShipB_refined.Changing_Speed) under OptSafe	Satisfied
4	Pr[<=200](<> windy && ShipB_refined.Change_route_drift) under OptSafe >= Pr[<=200](<> windy && ShipB_refined.Changing_Speed) under OptSafe	Satisfied

Query 1 defines the Safe strategy where two ships never collide by checking if for every possible path, the state *End* of the component *Scheduler* is never visited. This particular state is reached only when two ships collide and the game is over.

Query 2 synthesizes the optimal strategy with a goal of minimizing the value of the hybrid clock fc within 200 time units under the safe strategy. Note that this hybrid clock is used to measure the fuel consumption of shipB. The synthesised strategy is, thus, both safe and it strives for an optimal fuel consumption for shipB.

Query 3 presents the comparison of the selection of states *Change_route* and *Changing_Speed* by shipB in case of adjustment after detecting the other ship within 200 time units under the previously computed Opt (near-optimal) strategy. The probability of state *Change_route* to get selected are greater than *Changing_Speed*. This is because *Change_route* has a lower fc' rate compared to other locations. UPPAAL STRATEGO executes this query for 62 runs and estimates the probability to be true (value=1) with confidence 0.95.

Query 4 presents the same comparison as Query 3 with the difference that weather conditions are taken into account. It states that the likelihood for shipB to opt for action $Change\_route\_drift$ in windy weather within 200 time units under the safe and optimal strategy is higher than selection of *Changing_Speed*.

UPPAAL STRATEGO executes this query for 582 runs and estimates the probability for it to be true with confidence 0.95.

Note that Queries 1 to 3 are proved both in the abstract and the refined models. Query 4 is provable only in the refinement model, because of the new variable $windy$ and the new state $Change\_route\_drift$ introduced in the refinement.

7 Conclusions and Future Work

The novelty of this paper is introducing the notion of maritime game as a special case of Stochastic Priced Timed Game and constructing the respective model of the autonomous navigation using UPPAAL STRATEGO. The practical usability of our approach (maritime game) is to develop the theory of autonomous ships safe navigation and for that purpose to analyze the navigation problem in a rigorous state-based model setting. We use the refinement technique to develop a game model in a correct-by-construction manner. The stepwise refinement approach helps to avoid ambiguity in the modelling to verify the satisfiability of the safety requirements of the model.

In this paper, the approach for the strategy synthesis of safe navigation has been presented as a stochastic two players priced game with the goal of collision avoidance. Taking into account several practically important side constraints such as wind, currents, navigation mistakes by the vessel of the adversary, and involvement of other obstacles (nautical signs, small boats) complicates the synthesis task and presumes the validation of the approach under extra constraints not studied in standard game-theoretic setting yet. Though limited with two ships navigating in offshore scenarios, our work is the first attempt to synthesize a safe and optimal navigation strategy that also takes into account weather conditions. The navigation problem is exemplified based on navigation specification and COLREG Rule 15. Further developing the Maritime theory to capture multi-vessel navigation situations in traffic-intensive harbour zones and integration of winter navigation remain as future work.

Acknowledgments

This study has been partly supported by the project ESC funded by the Academy of Finland (grant No.308980) and the Estonian Ministry of Education and Research institutional research grant no. IUT33-13. We would like to thank Marius Mikuionis for his assistance with UPPAAL STRATEGO. We also thank the anonymous reviewers for their inputs and suggestions.

References

1. Yaseen Adnan Ahmed and Kazuhiko Hasegawa. Fuzzy reasoned waypoint controller for automatic ship guidance. *IFAC-PapersOnLine*, 49(23):604–609, 2016.
2. Sauli Ahvenjärvi. The human element and autonomous ships. *TransNav: International Journal on Marine Navigation and Safety of Sea Transportation*, 10, 2016.

3. Ralph-Johan Back and Joakim von Wright. *Refinement Calculus: A Systematic Introduction*. Springer Heidelberg, Graduate Texts in Computer Science, 1998.
4. Davide Basile, Alessandro Fantechi, Luigi Rucher, and Gianluca Mandò. Statistical model checking of hazards in an autonomous tramway positioning system. In *International Conference on Reliability, Safety, and Security of Railway Systems*, pages 41–58. Springer, 2019.
5. Davide Basile, Maurice H ter Beek, and Axel Legay. Strategy Synthesis for Autonomous Driving in a Moving Block Railway System with UPPAAL STRATEGO. In *International Conference on Formal Techniques for Distributed Objects, Components, and Systems*, pages 3–21. Springer, 2020.
6. Gerd Behrmann, Alexandre David, and Kim G Larsen. A tutorial on UPPAAL. In *Formal methods for the design of real-time systems*, pages 200–236. Springer, 2004.
7. Michael R Benjamin, Joseph A Curcio, John J Leonard, and Paul M Newman. Navigation of unmanned marine vehicles in accordance with the rules of the road. In *Proceedings 2006 IEEE International Conference on Robotics and Automation, 2006. ICRA 2006.*, pages 3581–3587. IEEE, 2006.
8. Alexandre David, Peter G Jensen, Kim Guldstrand Larsen, Axel Legay, Didier Lime, Mathias Grund Sørensen, and Jakob H Taankvist. On time with minimal expected cost! In *International Symposium on Automated Technology for Verification and Analysis*, pages 129–145. Springer, 2014.
9. Alexandre David, Peter Gjøl Jensen, Kim Guldstrand Larsen, Marius Mikučionis, and Jakob Haahr Taankvist. UPPAAL STRATEGO. In *International Conference on Tools and Algorithms for the Construction and Analysis of Systems*, pages 206–211. Springer, 2015.
10. Marko Höyhtyä, Jyrki Huusko, Markku Kiviranta, Kenneth Solberg, and Juha Rokka. Connectivity for autonomous ships: Architecture, use cases, and research challenges. In *2017 International Conference on Information and Communication Technology Convergence (ICTC)*, pages 345–350. IEEE, 2017.
11. IMO. Convention on the international regulations for preventing collisions at sea (COLREGs), 1972.
12. Yazdi Ibrahim Jenie, Erik-Jan van Kampen, and Bart Remes. Cooperative autonomous collision avoidance system for unmanned aerial vehicle. In *Advances in Aerospace Guidance, Navigation and Control*, pages 387–405. Springer, 2013.
13. Shyam Lal Karra, Kim Guldstrand Larsen, Florian Lorber, and Jiří Srba. Safe and time-optimal control for railway games. In *International Conference on Reliability, Safety, and Security of Railway Systems*, pages 106–122. Springer, 2019.
14. Shmuel Katz. A superimposition control construct for distributed systems. *ACM Transactions on Programming Languages and Systems (TOPLAS)*, 15(2):337–356, 1993.
15. Matti Kutila, Pasi Pyykönen, Hanno Holzhüter, Michèle Colomb, and Pierre Duthon. Automotive LIDAR performance verification in fog and rain. In *2018 21st International Conference on Intelligent Transportation Systems (ITSC)*, pages 1695–1701. IEEE, 2018.
16. Kim Guldstrand Larsen, Marius Mikučionis, and Jakob Haahr Taankvist. Safe and optimal adaptive cruise control. In *Correct System Design*, pages 260–277. Springer, 2015.
17. Sang-Min Lee, Kyung-Yub Kwon, and Joongseon Joh. A fuzzy logic for autonomous navigation of marine vehicles satisfying COLREG guidelines. *International Journal of Control, Automation, and Systems*, 2(2):171–181, 2004.

18. Maximilian Mühlegg, Johann C Dauer, Jörg Dittrich, and Florian Holzapfel. Adaptive trajectory controller for generic fixed-wing unmanned aircraft. In *Advances in Aerospace Guidance, Navigation and Control*, pages 443–461. Springer, 2013.
19. Lokukaluge Prasad Perera, Joao Paulo Carvalho, and Carlos Guedes Soares. Autonomous guidance and navigation based on the COLREGs rules and regulations of collision avoidance. In *Proceedings of the international workshop advanced ship design for pollution prevention*, pages 205–216, 2009.
20. Colin Snook and Marina Waldén. Refinement of Statemachines using Event-B semantics. In *International Conference of B Users*, pages 171–185. Springer, 2007.
21. Richard S Sutton and Andrew G Barto. *Reinforcement learning: An introduction.* Cambridge, MA: MIT Press, 2011.
22. Ken Teo, Kai Wei Ong, and Hoe Chee Lai. Obstacle detection, avoidance and anti collision for MEREDITH AUV. In *OCEANS 2009*, pages 1–10. IEEE, 2009.
23. Jesus Mediavilla Varas, Spyros Hirdaris, Renny Smith, Paolo Scialla, Walter Caharija, Zakirul Bhuiyan, Terry Mills, Wasif Naeem, Liang Hu, and Ian Renton. MAXCMAS project: Autonomous COLREGs compliant ship navigation. In *Proceedings of the 16th Conference on Computer Applications and Information Technology in the Maritime Industries (COMPIT)*, pages 454–464, 2017.
24. Sun Wuchen, Bu Renxiang, Liu Yong, Li Xinyu, Li Liangqi, and Fan Pengfei. Prediction of leeway and drift angle based on empirical formula. In *Proceedings of the Asia-Pacific Conference on Intelligent Medical 2018 & International Conference on Transportation and Traffic Engineering 2018*, pages 196–199, 2018.