This article can be cited as K. Challita and H. Farhat, Monitoring the Relative Positions of Cars in a Two-Dimensional Space, International Journal of Artificial Intelligence, vol. 17, no. 1, pp. 83-101, 2019.

Monitoring the Relative Positions of Cars in a Two-Dimensional Space

Khalil Challita and Hikmat Farhat

Department of Computer Science
Notre Dame University - Louaize
kchallita@ndu.edu.lb
hfarhat@ndu.edu.lb

ABSTRACT

We propose in this paper a qualitative model that allows us to monitor cars in real-time. Our model uses two distinct and complementary formalisms for capturing the relative positions of a large set of cars that are moving over time. The algorithm we propose and implement enables us to answer some high level questions, such as determining if two cars are at a safe distance from each other or if a collision between them is inevitable. The main aim of this paper is to elaborate a model where a set of cars can move safely on a highway.

Keywords: GIS, qualitative reasoning, RCC8, PLTL, CSP.

2012 Computing Classification System: Computing methodologies—Artificial Intelligence—Knowledge representation and reasoning—Temporal reasoning—Spatial and physical reasoning;500.

1 Introduction

Since its inception, qualitative spatial reasoning has found many applications in several important areas such as geographic information systems (Sweeney, 1999; Goodchild, 2009; Yang, Kafatos, Wong, Wolf and Yang, 2010; Bonham-Carter, 2014), natural language processing (Biddle, Luke, Magowan and White, 2013; Arnold and Tilton, 2015), robot navigation (Trautman, Jiaxin, Murray and Krause, 2013; Kümmerle, Ruhnke, Steder, Stachniss and Burgard, 2015), cars automation (Sule, Gupta and Desai, 2015; Weyer, Fink and Adelt, 2015), and robots and navigation (Fernandez-Gauna, Lopez-Guede, Zulueta, Echegoyen and Grana, 2011; Zato, Villarrubia, Sanchez, Bajo and Corchado, 2013). In this paper we turn our attention to designing and implementing an algorithm that allows us to compare the relative positions of cars that evolve over time. The following papers were of help for us when designing and building our model (Voda, 1998; Preitl, Precup, Fodor and Bede, 2006; Tomescu, Preitl, Precup and Tar, 2007; Precup, Preitl, Petriu, Tar, Tomescu and Pozna, 2009; Precup, David, Petriu, Preitl and Radac, 2011; Ginter and Pieper, 2011; Bakefayat and Tabrizi, 2016; Challita, 2009). We would like to be able to answer simple questions such as: *Is car x too close to car y*? *Is car x going to hit another car in the future*? For that, we use two very well known qualitative formalisms to design a spatio-temporal model for our problem: Propositional linear temporal

logic (PLTL) and the Region Connection Calculus (RCC). These formalisms are described in Sections 3 and 4.
Note that other researchers have already combined these formalisms to study objects in the space (e.g. (Wolter and Zakharyaschev, 2000; Challita, 2012; Challita, 2017)). In this case we combine PLTL to $RCC8$, which is a very well known fragment of RCC. Our algorithm captures the relative positions of cars in a two-dimensional space and allows us to monitor their movements. We assume in our model that all the cars have the same size and that they evolve on a three-lanes highway. Since a car is not allowed to go backward, at each time step it is free to move in one of the three basic directions: North, East, and West; or it can simply stand still. The last action could represent an immobile car due to heavy traffic ahead, a roadblock, or a breakdown.
The purpose of our work is to elaborate a model where a set of cars can move safely on a highway (i.e. by avoiding accidents).
This paper is divided as follows. We start by defining our problem in Section 2. Next, we give an overview of the RCC8 formalism in Section 3 and explain how to interpret its basic relations in our model. In Section 4, we introduce propositional linear temporal logic (PLTL) and justify its use to capture the movement of cars over time. In Section 5, we propose two algorithms that simulate the evolution of a set of cars in a two-dimensional space. Before concluding, we implement and discuss our models in Section 6.

2 Problem definition

As we already mentioned in the Introduction, we are interested in monitoring a set of cars that evolve in a two-dimensional plane, where at each time step a car is characterized by its position and speed. Starting with a simple model, we add to it more constraints in order to allow the cars to evolve in a safe environment with zero accidents.
In order to achieve our aim, we should be able to answer simple questions such as: (1) *"Are cars x and y going to collide?"*, (2) *"Are cars x and y very close to each other?"*, (3) *"Is it safe for car x to change lane?"*, (4) *"Can car x safely increase its speed?"*.
Integrating our model to a real-life automated monitoring system should help us take the appropriate action based on a given situation. For example, we may be able to avoid case 1 (i.e. having an accident) by reducing the speed of one of the cars or by changing its direction. In case 2, a warning should be issued to the following car to reduce its speed or change its direction. Case 3 is easy to deal with as long as the car does not change lane when it increases its speed, provided that there is no other car at a close distance in front of it.
Note that reasoning with cars in a two-dimensional space is enough since it captures realistically what happens in a real-life scenario. Our model can be extended naturally to the case of a three-dimensional space where, for example, we have several layers of bridges with several cars evolving at different levels. In such case we should consider different sets of cars separately, based on the planes they belong to and where we assume that cars that belong to two different sets (or planes) do not interfere together.
For example, if we have a highway with cars $x_1, \ldots, x_n$ and a bridge above it with cars $y_1, \ldots, y_m$,

then we should reason with the two different groups of cars (i.e. x_i and y_j) separately. Such a case is illustrated in Figure 1 where the cars on the road are represented by discs and the cars on the bridge (dotted lines) are represented by squares.

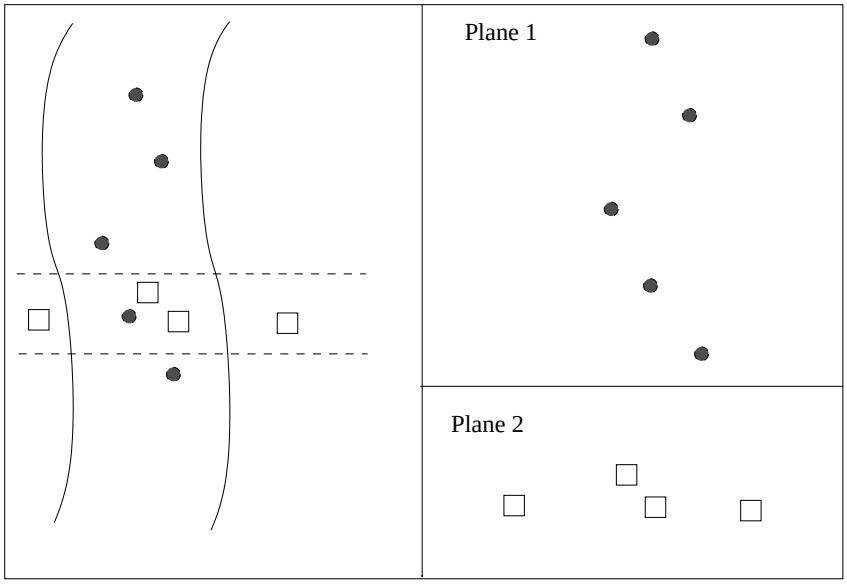

Figure 1: Cars on the left are represented in two distinct planes on the right.

Indeed, projecting both sets of cars on a single plane may yield some errors: a car y_k may be passing on the bridge exactly above the car x_l at the time of the projection. In this case our conclusion that x_l *collides with* y_k would be erroneous.
In our model, we assume that all the cars have the same size. We represent them by 1-unit discs or squares, as shown in Figure 2. In an unrestricted model, a car is free to move in any of the four basic directions (i.e. north, east, south, west). Each car is characterized by its position on a two-dimensional plane and its speed, the latter being bounded by an upper and a lower value. At each timestep, the speed of a car determines the number of squares it crosses. Two cars collide together whenever their paths overlap.
It is natural to choose a discrete model for the timesteps. For example, we may assume that the initial car configuration starts at t_0 and that the following states are captured at $t_1, t_2, \ldots, t_i, \ldots, t_n$. The time between timesteps t_i and t_{i+1} may correspond to a fraction of a second in a real-life case.
In this paper, we consider a realistic model where the cars evolve on a three-lanes highway; which could be naturally extended to the case of a n-lanes highway. Each car is determined by its speed, its position, and its temporal relations with respect to surrounding cars. At each timestep, and whenever possible, a car can:

- Move to the left or to the right (to change lane).
- Increase or decrease its speed by at most two units.

We set the speed of a car to be a value between 0 and 3. It determines the number of squares a car can cross. Note that in order to avoid collisions, the action to be taken by a car affects

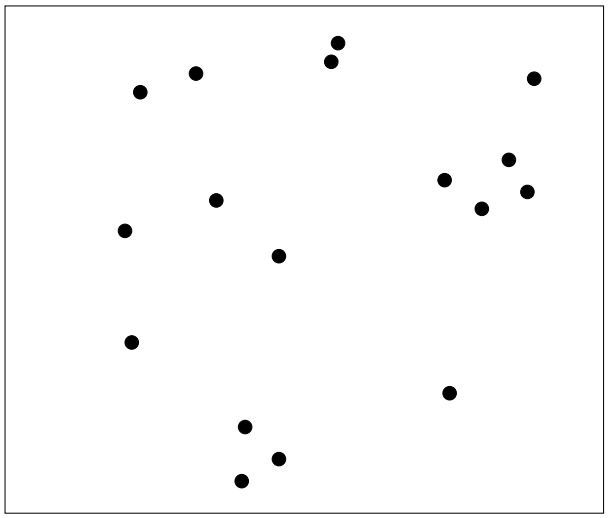

Figure 2: Cars represented by discs of diameter 1.

(and is also affected by) its surrounding ones. For example, assume that at time t_i a car is on the middle lane with a speed of 1. If at time t_{i+1} it moves west and increases its speed by 1, then it shall move to the left lane and move forward two squares. To achieve this manoeuvre safely, the car must check that the left squares it is going to use remain free at time t_{i+1}. In other words, it must make sure that no car that is closely behind it on the left lane is moving fast and may use those same squares and, similarly, no car ahead of it on the left lane is slowing down. This is because we make the following assumptions at any time t_i:

1. For any two cars x and y:
    - If x is in front of y then x moves first.
    - If x and y are on the same level, then the leftmost one moves first.
2. A collision occurs between two cars if their paths overlap at some timestep t_i.

We are aware of the fact that these two points do not reflect what actually happens in real life since all the cars move concurrently. But we adopt them given the fact that our model is discrete rather than continuous.
Since the cars are expected to move over time, we must endow our model with a spatial and temporal formalisms to track their evolution in a two-dimensional plane. The model we present not only provide a safe environment for cars, but also allows us to answer a wide variety of questions such as: (1) *Since when car x is behind car y?"*, (2) *Are cars x and y close to each other?"*, (3) *Is it safe for car x to change lane?"*

3 Spatial representation of a set of cars

We use the well-known $RCC8$ formalism to compare the relative positions of cars in a two-dimensional space. Recall that this formalism is one of the subsets of RCC, the original Region Connection Calculus introduced by Randell, Cui and Cohn (Randell, Cui and Cohn, 1992; Cui,

Cohn and Randell, 1993). It was first studied by Bennett (Bennett, 1994) and then analyzed by Renz and Nebel (Renz and Nebel, 1997), as well as by many other researchers (Li and Wang, 2006; Challita, 2012; Amaneddine and Challita, 2016).
$RCC8$ has eight basic relations that are jointly exhaustive and pairwise disjoint. An example of a spatial representation in the plane of these relations is given in Figure 3.

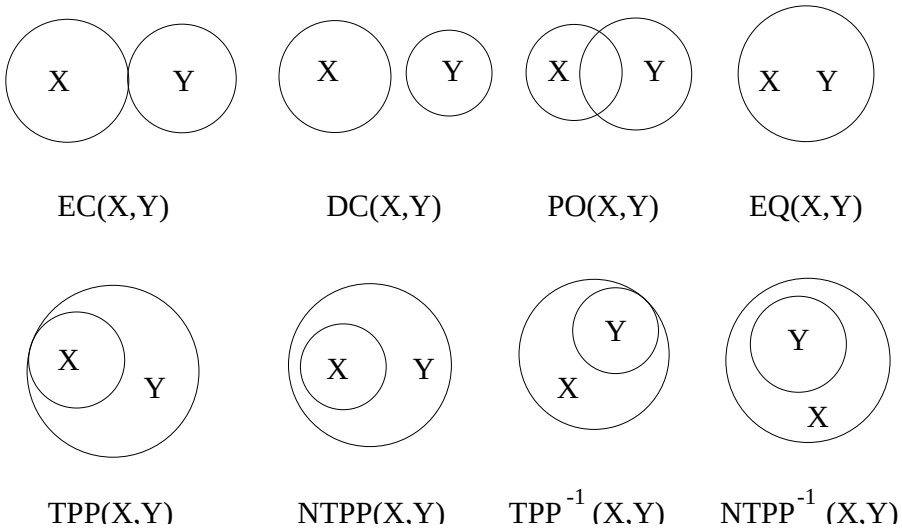

Figure 3: Bi-dimensional representation of the relations of $RCC8$.

The basic relations are EC, DC, PO, EQ, TPP, $NTPP$, TPP^{-1}, $NTPP^{-1}$. Their respective and informal meanings for two spatial regions are: "externally connected", "disconnected", "partial overlap", "equal", "tangential proper part", "non-tangential proper part", "tangentially contains" and "strictly contains".
The RCC formalism allows us to reason about non-empty spatial regions. Given two spatial variables x and y, $C(x, y)$ has the following interpretation: the topological closures of x and y have at least one common point. Given the relation C, one can define other RCC relations. We next give the basic definitions relevant to our case.

$$DC(x, y) \stackrel{def}{\equiv} \neg C(x, y) \tag{3.1}$$

$$EQ(x, y) \stackrel{def}{\equiv} P(x, y) \wedge P(y, x) \tag{3.2}$$

$$PO(x, y) \stackrel{def}{\equiv} O(x, y) \wedge \neg P(x, y) \wedge \neg P(y, x) \tag{3.3}$$

$$EC(x, y) \stackrel{def}{\equiv} \neg O(x, y) \wedge C(x, y) \tag{3.4}$$

$$TPP(x, y) \stackrel{def}{\equiv} PP(x, y) \wedge \exists\, z[EC(z, x) \wedge EC(z, y)] \tag{3.5}$$

$$NTPP(x, y) \stackrel{def}{\equiv} PP(x, y) \wedge \neg\exists\, z[EC(z, x) \wedge EC(z, y)] \tag{3.6}$$

$$TPP^{-1}(x, y) \stackrel{def}{\equiv} TPP(y, x) \tag{3.7}$$

$$NTPP^{-1}(x, y) \stackrel{def}{\equiv} NTPP(y, x) \tag{3.8}$$

Given a certain number of cars in space, we can compare their respective positions using the $RCC8$ model. Obviously, representing cars with points will not work here. Given two cars x

and y, the relations $PO(x, y)$, $TPP(x, y)$, $NTPP(x, y)$ (as well as their inverse relations) have no meaning since two points cannot overlap with each other or be inside one another.
We explain below how to interpret these relations in our model, assuming that all the cars move in the same direction.
First, we must represent a car with a specific shape. Without loss of generality, we choose to represent a car with a one-unit square as shown in Figure 4. The reference point of a car (i.e. square) is the center of the square.

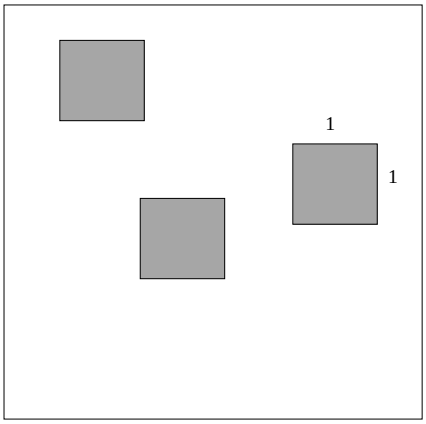

Figure 4: Three cars represented as 1-unit squares.

We are now ready to interpret all of the $RCC8$ relations in our discrete model.

Definition 3.1. Given two cars x and y, we have the following meanings at time t_i:

1. $EC(x, y)$: The distance between x and y is exactly two squares. This relation has no inverse since we assume that all the cars move in the same direction. Note that the number two corresponds to the fact that a car can increase its speed by at most two units at each timestep.
2. $DC(x, y)$: The distance between x and y is greater than three.
3. $PO(x, y)$: The paths of x and y overlapped, and thus a collision occurred between both cars.
4. $EQ(x, y)$: x and y represent the same car.
5. $TPP(x, y)$: x is less than three squares behind y, and is on the same lane of y.
6. $NTPP(x, y)$: x is less than three squares behind y, but is on an different lane than y.
7. $TPP^{-1}(x, y)$: y is less than three squares behind x, and is on the same lane of x.
8. $NTPP^{-1}(x, y)$: y is less than three squares behind x, but is on an different lane than x.

We should avoid collisions in case we wish to implement a safe automated system to monitor cars.
Figure 5 represents three cars moving randomly in a two-dimensional plane. At time t_i they

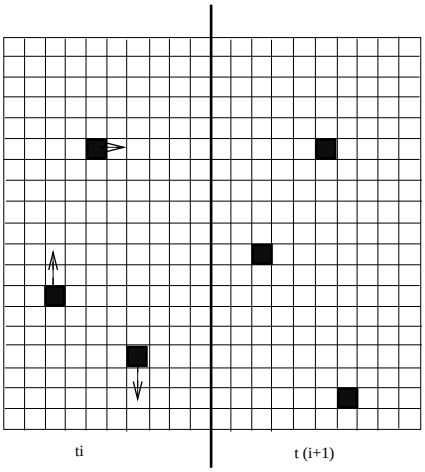

Figure 5: Three cars moving randomly in a plane.

decided to move one square to the right, two squares forward, and two squares backward, respectively. The resulting configuration is shown at time t_{i+1}.
In Figure 6, we show the resulting state of two cars moving on a three-lanes plane where the first car moves two squares forward and the second moves to the left, then one square forward. We notice that the cars collided together since their paths overlapped, although that at time t_{i+1} the cars appear to be behind each other.

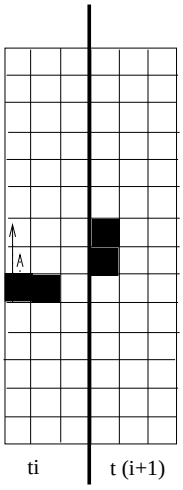

Figure 6: Two cars colliding together.

We next give two informal definitions that are related to some of the $RCC8$ base relations.

Definition 3.2. A car x is at a **safe distance** from car y if more than three squares separate the vehicles.
In other words,

$$S(x, y) \Leftrightarrow DC(x, y) \tag{3.9}$$

Our definition is consistent with the fact that the maximum speed of a car is three and that all the cars have the same dimensions (in this case a 1-unit square). So even if y is standing

still and x is moving at its maximum speed, no collision may occur at time t_{i+1} in case we had $DC(x, y)$ at time t_i.

Definition 3.3. Two cars are **dangerously close** to each other if the distance that separates them is less than or equal to three in any direction.
In other words,

$$D(x, y) \Leftrightarrow (EC(x, y) \vee TPP(x, y) \vee NTPP(x, y) \vee TPP^{-1}(x, y) \vee NTPP^{-1}(x, y)) \quad (3.10)$$

This is the case when we have one of the following relations between cars x and y: $EC(x, y)$ or $TPP(x, y)$.
For example, in Figure 7 cars x and y (and also y and z) are dangerously close to each other, whereas cars x and z are at a safe distance.

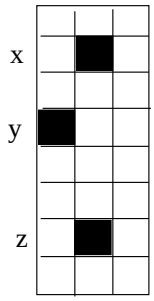

Figure 7: Cars x and y (and also y and z)are dangerously close to each other.

No matter how x and z move, they will not hit each other at the next timestep. On the other hand, if y does not move while z moves to the left and, after several steps it moves three squares forward, then y and z will collide together.

4 Temporal reasoning applied to cars

In this section we introduce a well-known formalism that allows us to capture qualitative information about cars.
Propositional linear temporal logic (PLTL) enables us to make statements such as: ”from now on, it will always be true that”, ”some time in the past, event A occured”, or ”event A will occur until B occurs”. It was designed by philosophers to study the way that time is used in natural language arguments. It also interested logicians, from Aristotle to Prior (Prior, 1957). A fundamental contribution to the theory was made by Kamp (Kamp, 1968) when he introduced the operators *Until* and *Since*. Pnueli (Pnueli, 1977) brought it to computer science and used it in concurrent systems. We next give a brief overview of PLTL.
The language of PLTL is defined with means of :

- A set of propositional variables $Prop = \{p, q, r, \ldots\}$.
- Logical constants: $\neg$, $\vee$, $\wedge$, $\rightarrow$, $\leftrightarrow$, True ($\top$), False ($\bot$).

- Temporal operators: $\bigcirc$, $\odot$, $\mathbf{F}$, $\mathbf{G}$, $\mathbf{P}$, $\mathbf{H}$, $\mathcal{U}$, $\mathcal{S}$.
- Rules:
  1. All propositional variables are formulas.
  2. If ϕ and ψ are formulas, then $\neg\phi$, $\phi \vee \psi$, $\phi \wedge \psi$, $\phi \rightarrow \psi$, $\phi \leftrightarrow \psi$, $\bigcirc\phi$, $\odot\phi$, $\mathbf{F}\phi$, $\mathbf{G}\phi$, $\mathbf{P}\phi$, $\mathbf{H}\phi$, $\phi\,\mathcal{U}\,\psi$, $\phi\,\mathcal{S}\,\psi$ are also formulas.
  3. Each formula is obtained by applying clauses (1) and (2) a finite number of times.

Intuitively, the respective meanings of the temporal operators $\bigcirc\phi$, $\odot\phi$, $\mathbf{F}\phi$, $\mathbf{G}\phi$, $\mathbf{P}\phi$, $\mathbf{H}\phi$, $\phi\mathcal{U}\,\psi$, $\phi\,\mathcal{S}\,\psi$ are: "ϕ is true next time", "ϕ was true at the previous time", "ϕ is sometime true in the future", "ϕ will always be true", "ϕ was sometime true in the past", "ϕ was always true", " ϕ is true until ψ is true", and "ϕ is true since ψ was true".

A *model* M is a couple $M = (S, V)$, where $S = \{i_0, i_1, \ldots, i_n, \ldots\}$ is a set of *dates* or *instants*, and $V : Prop \longrightarrow 2^S$ (i.e. V is a mapping from $Prop$ to the set of subsets of S). $V(p)$ contains all the instants for which the propositional variable p is true.
Given a model M and a date i in S, we say that M satisfies a formula at instant i if:

$$M, i \models \top \quad and \quad M, i \not\models \bot \tag{4.1}$$

$$M, i \models p \quad iff \quad i \in V(p), \quad with \quad p \in Prop \tag{4.2}$$

$$M, i \models \neg\phi \quad iff \quad M, i \not\models \phi \tag{4.3}$$

$$M, i \models \phi \vee \psi \quad iff \quad M, i \models \phi \quad or \quad M, i \models \psi \tag{4.4}$$

$$M, i \models \phi \wedge \psi \quad iff \quad M, i \models \phi \quad and \quad M, i \models \psi \tag{4.5}$$

$$M, i \models \bigcirc\phi \quad iff \quad M, i+1 \models \phi \tag{4.6}$$

$$M, i \models \mathbf{F}\phi \quad iff \quad \exists\, j,\ j \geq i \quad such\, that \quad M, j \models \phi \tag{4.7}$$

$$M, i \models \mathbf{G}\phi \quad iff \quad \forall\, j,\ j \geq i \quad we\, have \quad M, j \models \phi \tag{4.8}$$

$$M, i \models \phi\,\mathcal{U}\,\psi \ \ iff \ \ \exists\, k,\ k \geq i\ such\ that\ M, k \models \psi\ and\ \forall\, j, i \leq j < k\ we\ have\ M, j \models \phi \tag{4.9}$$

Definition 4.1.

- A formula ϕ is satisfied in a model M (or M satisfies ϕ) if $M, 0 \models \phi$ holds.
- A formula ϕ is satisfied if there is a model that satisfies it.
- A formula ϕ is valid if it is satisfied in any model.

In our case, the model we consider is a set of cars that move on a grid. We assumed that a car can move between zero and three squares. The possible moves in the general case are: up, down, left, right; whereas all the cars follow the same direction on a three-lanes plane. Both cases are simplified versions of a real-case scenario. They are meant to help us understand

the requirements to avoid car accidents.

Our model allows us to reason about any random configuration of cars and uses propositional linear temporal logic to study their relationships. For example, if $\phi = TPP(x, y)$ and we have **F**ϕ, then this means that sometime in the future x and y will get dangerously close to each other. Moreover, and on a three-lanes plane, we know that x will be behind y . An example of such a situation is shown in Figure 8 at time t_{i+j}.

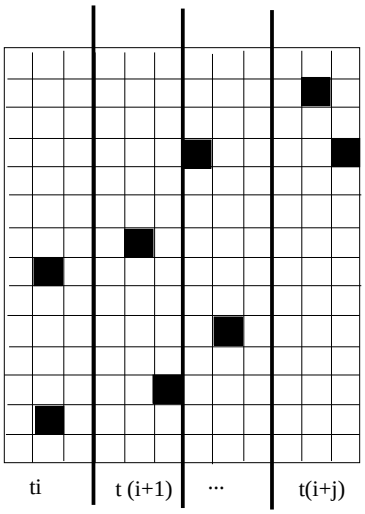

Figure 8: Cars x and y getting dangerously close to each other at time t_{i+j}.

We next discuss our models and provide algorithms for solving our problem.

5 Algorithms

In this section we describe two algorithms to monitor a finite set of cars. We shall use the eight basic relations we defined in Section 3. Our objective here is to capture the relationship of the cars at any given point in time. As stated in the Introduction, we study the evolution of a set S of cars in a two-dimensional plane first, then on a three-lanes highway. In the latter case, the cars are placed on a $2n \times 3$ grid. The position of a car is determined by its row and column, where the rows are numbered from 0 to $2n - 1$ and the columns are numbered from 0 to 2. A simple example is given in Figure 9.

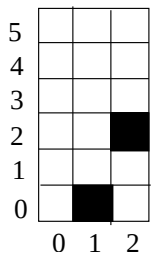

Figure 9: A car at position $(0, 1)$ and another one at position $(2, 2)$.

5.1 Basic case

In this scenario, the cars move independently of each other on a n by n grid of fixed size. For practical reasons, we use an array A_{t_i} to represent the position of the cars on the grid. Every car $X_i \in S$ has specific coordinates at time t_i. $A_{t_k}(x_i, x_j) = 1$ means there is a car at coordinates (x_i, x_j); and $A_{t_k}(x_i, x_j) = 0$ means the slot is empty. At each iteration, a car randomly selects one of the following actions: move to the left, to the right, up, down, or remain still. The default action is to stay on the same lane in case an action cannot take place. For example, if a car is on column 0 at time t_i and the random choice is to move to the left, then the car remains in its current position at t_{i+1}.
Next, a random number a is assigned to a car, where a is an integer between -2 and 2. The value corresponds to a decrease or an increase of speed. Note that at anytime the speed must always be between 0 and 3, which corresponds to the of squares a car can cross in one timestep. All the cars which paths overlap (i.e. collide together) are removed at the end of each iteration. The iterations stop when at least 10% of the cars have collided together. We next give a high-level algorithm for this case.

Algorithm 1 Zero-constraints algorithm

```
Randomly distribute n^2/4 different cars on a n × n grid
while there is still more than 90% of the cars on the grid do
  for i = n − 1 to 0 do
    for j = 0 to n − 1 do
      Randomly assign a move to car X_i: left, right, up, down, or no move.
      Assign a random number a, where a ∈ {−2, −1, 0, 1, 2} and such that the new speed
      s_{i+1} = s_i + a satisfies the following: 0 ≤ s_{i+1} ≤ +3
      Remove the cars which paths have overlapped (i.e. PO(X_i, X_j))
    end for
  end for
end while
```

Running time analysis
Let $T_1(n)$ be the running time of Algorithm 1. It is easy to see that it requires $\Theta(n^2)$ to initialize the problem, plus a quadratic time for the two nested loop. Therefore we have:

$$T_1(n) = \Theta(n^2 + kn^2) = \Theta(kn^2)$$

where k is equal to the number of iterations required to reach the 90% threshold of the remaining cars.

5.2 Simulation on a three-lanes highway

In this case we use a $2n \times 3$ grid for the cars. Initially, $n^2/4$ cars are distributed over the first n rows. Since the maximum speed of a car is three, the topmost ones require at least $m = n/3$ timesteps to reach the upper limit of the grid. As we did in the previous subsection, we use an array A_{t_i} to represent the position of the cars on the grid. At each iteration, a car X_i randomly

selects one of the following actions: move to the left, to the right, or remain on the same lane. Before determining a random value a (corresponding to the increase/decrease in speed), we build and solve a constraint satisfaction problem (CSP) that involves $X_i = (x_i, x_j)$ with all the cars that are at levels $j, j+1, j+2$. This would help us specify an adequate value for a in order to avoid a collision with an adjacent car. In case all the values of a lead to a collision we backtrack by assigning another direction for X_i.An example is given in Figure 10, and the corresponding CSP is represented in Figure 11.

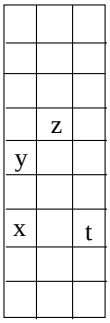

Figure 10: Three cars moving on a three lanes highway.

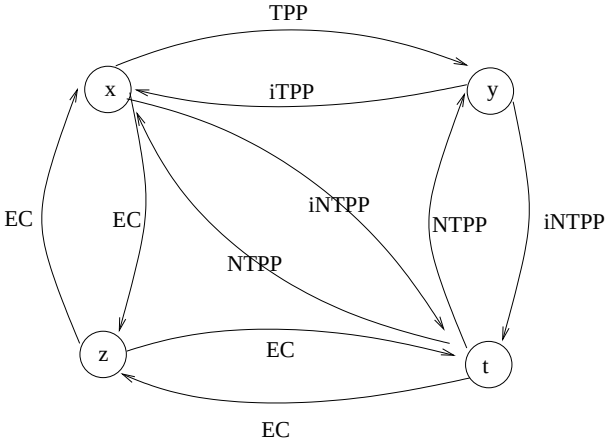

Figure 11: The CSP of the previous figure, where iNTPP and iTPP are the inverses of the relations NTPP and TPP.

When processing the car x, a quick analysis of the CSP shows that the following two cases lead to a collision:

1. x keeps the same direction and has a new speed of at least 2.
2. x moves to the right and has a speed of 3.

Example 1 Assume the below random choices :

- The previous speed of x was 1,
- $a = +1$,
- The direction is the same.

The new speed of car x becomes 2, so a collision with y is expected. We backtrack once, taking into account the information derived from the CSP. A possible solution is to make x move to the right in order to avoid an accident.

Example 2 Assume the below random choices :

- The previous speed of x was 3,
- $a = 0$,
- The direction is the same.

Again, a collision with y is expected. We backtrack once, taking into account the information derived from the CSP. We notice that changing the direction of x still leads to a collision. In this case we must choose $a > 0$ to avoid an accident.
We next give a high-level algorithm for this second case.

Algorithm 2 Three-lanes highway

Randomly distribute $n^2/4$ different cars on a $2n \times 3$ grid, using just the first n rows
$numberOfCollisions = 0$
$maxRow$ = The row of a topmost car on the grid
Let k be an integer less than m
for $l = 1$ to k **do**
  for $i = maxRow$ to 0 **do**
    for $j = 0$ to 2 **do**
      Randomly assign a move to car X_i: left, right, up, or no move
      Build and solve a CSP that involves $X_i = (x_i, x_j)$ with all the cars at levels $j, j+1, j+2$
      if The set of possible values for a is not empty **then**
        Randomly select one of these values to be the speed variation for X_i
      else
        Select another appropriate direction for X_i (i.e. left, right, up, or no move)
        Build and solve the corresponding CSP
        if The set of possible values for a is empty **then**
          Remove the cars which paths have overlapped and increment $numberOfCollisions$
        else
          Update the position and speed of car X_i, and if necessary $maxRow$
        end if
      end if
    end for
  end for
end for

Running time analysis

Let $T_2(n)$ be the running time of Algorithm 2. Building and processing a CSP is done at most

Iteration t_i	Number of cars	Time to compute t_i	Dangerously close cars
0	2.500	2.1	1987
10	2443	1.9	1768
20	2412	1.77	1646
30	2378	1.71	1545
40	2322	1.62	1456
50	2301	1.58	1347
100	2269	4.54	1315
150	2254	3.51	1295
158	2247	2.49	1283

Table 1: Iterations with an initial set of cars equal to 10.000

twice during an iteration. We use a constant amount of space with respect to the size of the problem. Indeed, we may backtrack at most once and the number of nodes of the CSP is at most nine. Therefore the overall running time of this algorithm is:

$$T_2(n) = \Theta(3kn) = \Theta(kn)$$

We implement our algorithms in the following section.

6 Simulation and results

We implemented both algorithms in Python and ran the code on a desktop machine with 8 GB RAM and a AMD 4.2 Ghz quad-core processor. We tracked the number of remaining cars, the time to reach the ith iteration (in milliseconds), and the overall number of cars that are *dangerously close to each other*; in other words, all the couples of cars that are related by one of the relations: $\{TPP, TPP^{-1}, NTPP, NTPP^{-1}\}$.

6.1 Implementation of Algorithm 1

We considered a small grid of dimensions 100 by 100 and another large one of dimensions 10.000 by 10.000. Tables 1 and 2 summarize the results obtained for these two different sets of cars. The iterations stopped after reaching the 90% threshold of the number of initial cars. Figures 12 and 13 represent the relationships between the initial number of cars and the number of iterations to reach the abovementioned threshold.
The number of cars that are involved in an accident at step t_i can be deduced by computing $t_i - t_{i+1}$. The running time of this algorithm becomes more important for a large set of cars. Indeed, this is because we need to make $O(n^2)$ comparisons at each time step to determine the relationships between all the cars.

6.2 Implementation of Algorithm 2

We consider here a more realistic example consisting of a set of 10.000 cars moving on a three-lanes highway. Prior to processing a car X_i from the set, we build a CSP on the fly, by

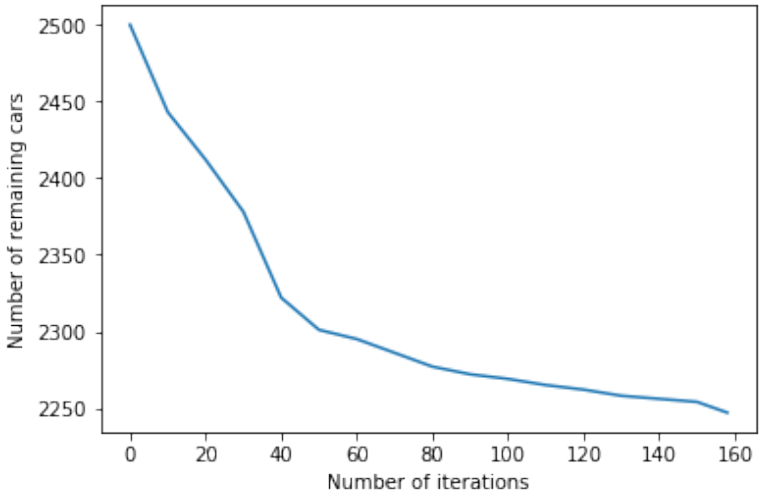

Figure 12: Initial set of 2500 cars decreasing over time

Iteration t_i	Number of cars	Time to compute t_i	Dangerously close cars
0	25.000.000	3215	22.230.430
100	24.734.454	3104	21.435.569
200	24.104.184	3078	20.455.346
300	23.899.345	2987	19.345.495
400	23.744.531	2912	18.947.431
500	23.434.567	2897	18.564.432
600	23.109.345	2765	18.234.958
700	22.760.249	2701	17.834.235
800	22.541.343	2654	17.567.458
816	22.483.345	543	17.345.395

Table 2: Iterations with an initial set of cars equal to 25.000.000

taking into account only the adjacent cars to X_i. More precisely, we do not include in our CSP the cars that are in the relation DC (since the space separating two cars is greater than the maximum speed a car can have). This is more space efficient than building another $2n \times 3$ grid and use it to store the CSPs of the cars. Table 3 provides us with the number of remaining cars for different timesteps, where $m = n/10$

The graph in Figure 14 represents the number of cars that did not collide with any other ones. We notice that our model did not forbid accidents from happening, but that the number of collisions is very low. This is due to our modified algorithm that involves solving a constraint satisfaction problem when processing a car. Note also that as time passes, the size of the grid expands (i.e. the cars are moving forward); which leaves more room for the remaining cars to manoeuver and avoid collisions. After a while, the number of accidents tends towards zero.

One may refer to Figure 10 in order to understand why some collisions are unavoidable in our model. Assume in this case that y is just in front of x and that z is to the right of y. Moreover, let the current speed of x be equal to three. No matter which choices we select for x (e.g.

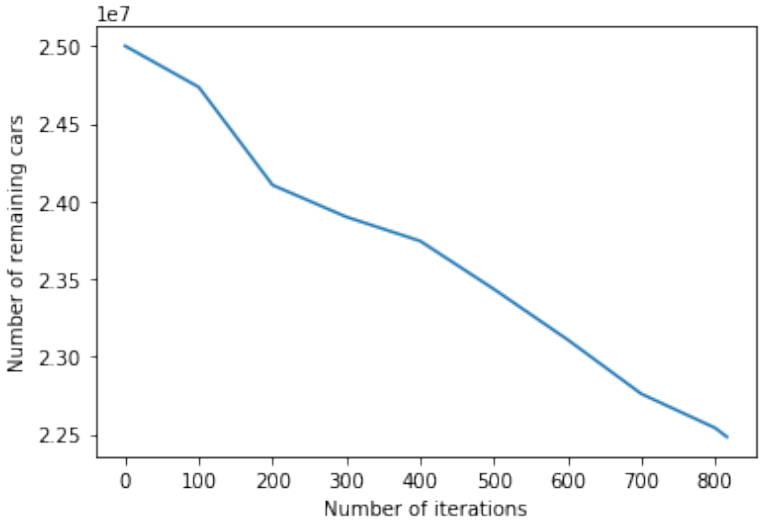

Figure 13: Initial set of 25.000.000 cars decreasing over time

Iteration t_i	Number of cars	Time to compute t_i	Dangerously close cars
0	10000	43	9549
100	9968	31	9123
200	9949	28	8652
300	9938	27	8209
400	9931	21	7120
500	9927	24	5234
600	9925	21	4785
700	9924	28	3128
800	9924	25	2876
900	9923	23	2548
1000	9923	19	2482

Table 3: Union-Find used with 1.000.000 cars

changing direction and/or decreasing its speed by two units) it will collide with y or z. This is due to what is known by the *horizon problem* in artificial intelligence, and reflects what may happen in a real case scenario where a driver is unable to practically avoid an accident. Hence the importance of enforcing a *safe* distance between cars on a highway, which value depends on the maximum speed of the cars.

7 Conclusion and future work

We proposed in this paper two algorithms that simulate the behavior of a set of cars in a two-dimensional space. We based our study on two spatial and temporal formalisms, namely RCC8 and PLTL. We started by generating a random number of cars that we distributed on a $n \times n$ grid, then considered the behavior of a set of cars on a three-lanes highway. Our approach allowed us to answer questions such as: *Is car x at a safe distance from car y*? or

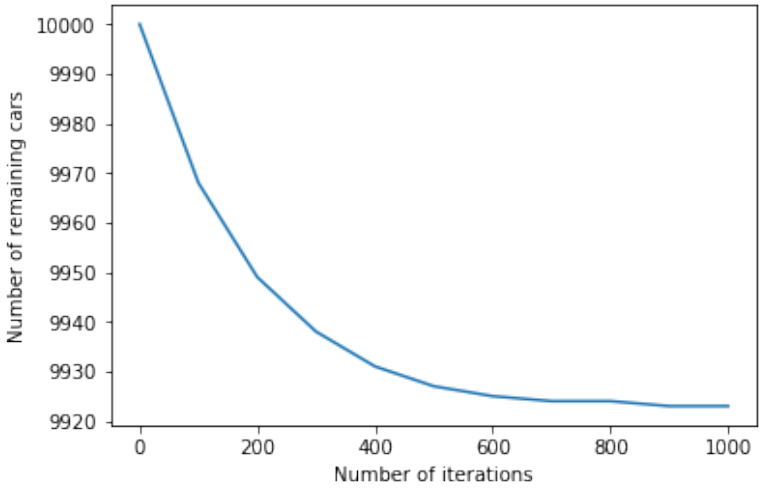

Figure 14: As time passes, the number of collisions tends to zero.

Is car x going to have an accident in the future? We implemented and tested the algorithms for several inputs. We used in the second one a constraint satisfaction problem that captures the relative positions of cars at a given timestep. This helped us make adequate choices at a given timestep, which reduced drastically the number of collisions. In the future, we intend to implement a model where cars may have different sizes and also enforce a safe distance between them. The value for that distance shall depend on the maximum speed a car can have, and should be computed in such a way to reach the zero-accidents target we fixed earlier in this paper.

References

Amaneddine, N. and Challita, K. 2016. Polynomial realization of ord-horn constraints, *International Journal of Artificial Intelligence* pp. 60–69.

Arnold, T. and Tilton, L. 2015. Natural language processing, *Humanities Data in R* pp. 131–155.

Bakefayat, A. S. and Tabrizi, M. M. 2016. Lyapunov stabilization of the nonlinear control systems via the neural networks, *Applied Soft Computing* **42**: 459–471.

Bennett, B. 1994. Spatial reasoning with propositional logics, *In Proceedings of the Fourth International Conference on Principles on Knowledge Representation and Reasoning (KR-94)* pp. 165–176.

Biddle, E., Luke, J., Magowan, J. and White, G. 2013. Natural language processing, *Google patents* .

Bonham-Carter, G. 2014. Geographic information systems for geoscientists: modelling with gis, *Elsevier* .

Challita, K. 2009. Reasoning with lines in the euclidean space, *In Proceedings of the 21st International Joint Conference on Artificial Intelligence (IJCAI 2009)* pp. 462–467.

Challita, K. 2012. A semi-dynamical approach for solving qualitative spatial constraint satisfaction problems, *Theoretical Computer Science* pp. 29–38.

Challita, K. 2017. Infinite rcc8 networks, *International Journal of Artificial Intelligence* **14**(1): 147–162.

Cui, Z., Cohn, A. and Randell, D. 1993. Qualitative and topological relationships in spatial databases, *Advances in Spatial Databases, Lecture Notes in Computer Sciences* pp. 293–315.

Fernandez-Gauna, B., Lopez-Guede, J., Zulueta, E., Echegoyen, Z. and Grana, M. 2011. Basic results and experiments on robotic multi-agent system for hose deployment and transportation, *International Journal of Artificial Intelligence* **6**(S 11): 183–202.

Ginter, V. and Pieper, J. 2011. Robust gain scheduled control of a hydrokinetic turbine, *IEEE Transactions on Control Systems Technology* **19**(4): 805 – 817.

Goodchild, M. 2009. Geographic information system, *Encyclopedia of Database Systems, Springer* pp. 1231–1236.

Kamp, H. 1968. Tense logic and the theory of order, *Thèse de l'université de Californie, Los Angeles* .

Kümmerle, R., Ruhnke, M., Steder, B., Stachniss, C. and Burgard, W. 2015. Autonomous robot navigation in highly populated pedestrian zones, *Journal of Field Robotics* pp. 565–589.

Li and Wang 2006. Rcc8 binary constraint network can be consistently extended, *Artificial Intelligence* pp. 1–18.

Pnueli, A. 1977. The temporal logic of programs, *In Proceedings of the eighteenth annual Symposium on Foundations of Computer Science (IEEE)* pp. 46–57.

Precup, R.-E., David, R.-C., Petriu, E. M., Preitl, S. and Radac, M.-B. 2011. Gravitational search algorithms in fuzzy control systems tuning, *Proceedings of 18th IFAC World Congress* pp. 13624–13629.

Precup, R.-E., Preitl, S., Petriu, E. M., Tar, J. K., Tomescu, M. L. and Pozna, C. 2009. Generic two-degree-of-freedom linear and fuzzy controllers for integral processes, *Journal of the Franklin Institute 346* **10**: 980–1003.

Preitl, S., Precup, R.-E., Fodor, J. and Bede, B. 2006. Iterative feedback tuning in fuzzy control systems. theory and applications, *Acta Polytechnica Hungarica* **3**(3): 81–96.

Prior, A. 1957. Time and modality, *Presse de l'université d'Oxford* .

Randell, D., Cui, Z. and Cohn, A. 1992. A spatial logic based on regions and connection, *In Proceedings of the Third International Conference on Principles of Knowledge Representation and Reasoning* pp. 165–176.

Renz, J. and Nebel, B. 1997. On the complexity of qualitative spatial reasoning: A maximal tractable fragment of the region connection calculus, *Proceedings of the 15th International Joint Conference on Artificial Intelligence* pp. 522–527.

Sule, S., Gupta, K. and Desai, V. 2015. Autonomous cars: The future of raodways, *International Journal of Students' Research in Technology & Management* pp. 203–206.

Sweeney, M. 1999. Geographic information systems, *Water Environment Research* pp. 551–556.

Tomescu, M. L., Preitl, S., Precup, R. E. and Tar, J. K. 2007. Stability analysis method for fuzzy control systems dedicated controlling nonlinear processes, *Acta Polytechnica Hungarica* **4**(3): 127–141.

Trautman, P., Jiaxin, M., Murray, R. and Krause, A. 2013. Robot navigation in dense human crowds: the case for cooperation, *IEEE International Conference on Robotics and Automation (ICRA)* pp. 2153–2160.

Voda, A. B. 1998. Iterative auto-calibration of digital controllers: Methodology and applications, *Control Engineering Practice* **6**(3): 345–358.

Weyer, J., Fink, R. and Adelt, F. 2015. Human–machine cooperation in smart cars. an empirical investigation of the loss-of-control thesis, *Safety science* pp. 199–208.

Wolter, F. and Zakharyaschev, M. 2000. Spatio-temporal representation and reasoning based on *RCC8*, *In Proceedings of the Seventh International Conference on Principles of Knowledge Representation and Reasoning* pp. 3–14.

Yang, C., Kafatos, M., Wong, D., Wolf, H. and Yang, R. 2010. Geographic information system, *Google patents* .

Zato, C., Villarrubia, G., Sanchez, A., Bajo, J. and Corchado, J. M. 2013. Pangea: A new platform for developing virtual organizations of agents, *International Journal of Artificial Intelligence* **11**(13 A): 93–102.