A Language for UAV Traffic Rules in an Urban Environment and Decentralized Scenario

Giuseppe Lombardi, Eric Medvet, Alberto Bartoli
Department of Engineering and Architecture, University of Trieste, Italy
giuseppe.lombardi@studenti.units.it, {emedvet, bartoli.alberto}@units.it

***Abstract*—Unmanned Aerial Vehicles (UAVs) are becoming increasingly popular and the amount of UAV traffic in urban environments will largely increase in the future, due to profitable tasks which are particularly suited to UAVs, e.g., parcel delivery and surveillance, in particular in the context of smart cities. Trying to ensure the traffic safety and efficiency by acting on the UAV controller alone might be challenging, since the set of involved players (regulators, manufacturers, business users) is large and diversified. In this work, we address this problem by proposing a language for defining rules suitable for UAV traffic which can be enforced in a decentralized way by the UAVs themselves, without any need for communication and regardless of the UAV navigation algorithm. The language allows to express realistic rules, such as "when cruising, keep a minimum altitude", concisely and such that they can be processed online by each single UAV basing on its perception of the nearby environment. We experimentally validate the ability of our proposal to impact on the UAV traffic efficiency and safety by performing a large number of simulations with and without a set of realistic rules.**

***Index Terms*—Agents, Simulation, UAVs, Rules, Safety, Smart Cities**

I. Introduction and related work

Unmanned Aerial Vehicles (UAVs) are becoming increasingly popular because of their wide variety of applications: urban security, military use, traffic surveillance, and, recently, goods delivery. Most of them well fit the scenario of smart cities: in such an urban environment, an UAV must be able to take a number of decisions in real time, with a limited and possibly noisy perception of the environment. Such decisions must take into account the need of avoiding collisions with buildings and other physical objects while, at the same time, trying to perform the UAV specific task (e.g., reaching a final destination to deliver a parcel).

With the foreseeable increase of the UAV traffic, it is evident that the goal of ensuring the efficiency and safety of the traffic itself might not be pursued by working on the UAV controller alone, mainly because the set of players involved in the UAV scenario (regulators, manufacturers, business users) is large and diversified. A different option consists in devising and enforcing a common regulation, similarly to what has been done for the case of road traffic at the dawn of its development. However, currently there are no official, domain specific, and widely adopted rule systems for which the UAVs have to comply with; in fact different communities (e.g., big companies such as Google, Amazon, etc., or specific commissions) are still discussing about various scenarios where UAVs traffic could be regulated either by a centralized management system or independently on each UAV.

In this work, we attempt to address this problem and propose a language for defining rules suitable for UAV traffic in an urban environment which can be enforced by the UAVs themselves without the need for mutual communication. Our contribution is twofold. First, we propose and experimentally evaluate a model for urban airspace traffic including buildings, UAVs, and their controllers which try to abide by the rules. Second, we propose a language (syntax and semantics) to define the rules which can be enforced in our model: the compliance to the rules can be evaluated online and on board, regardless of the algorithm being used by the UAV controller for navigation and without requiring communications with other UAVs, nor with a centralized authority. Our language is expressive enough to define concisely rules of realistic complexity such as "when cruising, keep a minimum altitude" or "keep a minimum distance from buildings". Finally, we experimentally validate our model (with and without rules) by performing a large number of simulations: the results show that the introduction of the rules impacts on the UAV traffic efficiency and safety in a consistent way.

To the best of our knowledge, there is only one study which propose a methodology for regulating UAV traffic in a fully decentralized scenario, by means of Defeasible Logic [1]. The cited work differs from our proposal since the proposed ruling system requires the communication among UAVs (whenever conflicts have to be solved) and is not fully agnostic w.r.t. the navigation algorithm.

A language which is very close to our proposal, but focus on road traffic instead of UAVs, has been proposed in [2], in form of a context-free grammar, as in our case. The authors also use the proposed language to generate automatically, by means of Grammatical Evolution, new sets of rules which can improve the traffic efficiency and safety.

Another attempt to rule the UAVs has been made in [3], where the authors propose a method based on a Genetic Algorithm (GA) for generating a set of instructions able to guide the UAVs. The cited paper considers a military scenario in which UAVs monitor enemies in an open environment, rather than an urban environment.

More research have been done instead on building new controllers—e.g., fuzzy controllers [4], by means of GA [5], for optimal path-finding [6], [7]—or centralized/decentralized systems for managing UAVs traffic [8], [9]: both this wealth of

research and our work share a common high-level goal, which is to shape how the UAV traffic will develop in order to ensure its effectiveness and safety.

II. The model

We consider a scenario with discrete space and discrete time in which buildings exist and a number of UAVs move, possibly colliding with other UAVs or buildings.

A. Space, buildings, and UAVs

The *space* is a finite, discrete set $\mathcal{S} = [0, l_x^{\mathcal{S}}] \times [0, l_y^{\mathcal{S}}] \times [0, l_z^{\mathcal{S}}] \subset \mathbb{N}^3$. We call *cell* a point of $\mathcal{S}$: a cell is defined by its coordinates (x, y, z).

A *building* is a subset of the space corresponding to a parallelepiped with the bottom face lying on the plane with $z = 0$, i.e., on the ground. The building is defined by a position x_0, y_0 and three sizes l_x, l_y, l_z: a cell (x, y, z) belongs to a building if and only if $x \in [x_0, x_0 + l_x]$, $y \in [y_0, y_0 + l_y]$, and $z \in [0, l_z]$. We denote by $\mathcal{B}$ the set of all the buildings in the space $\mathcal{S}$.

An *UAV* is an agent which can move in the space. The UAV is defined by its *position* (x, y, z), its *status* $s \in \{\text{alive}, \text{collided}\}$, its *controller*, and its *target cell* $(x_f, y_f, 0)$. At each time step, an UAV may stochastically be involved in a *collision*, which results in its status being set to collided. In particular, a collision occurs if any of the following conditions is met:

- the UAV position belongs to a building (i.e., $\exists B \in \mathcal{B} \mid (x, y, z) \in B$) and a random number in $[0, 1]$ is lower than $p_{B,\text{inside}}$;
- the UAV position is adjacent to a cell belonging to a building (i.e., such that the Manhattan distance between the position and the building cell is exactly 1) and a random number in $[0, 1]$ is lower than $p_{B,\text{close}}$;
- for each other UAV in the same position, a random number in $[0, 1]$ is lower than $p_{U,\text{same}}$;
- for each other UAV in an adjacent cell, a random number in $[0, 1]$ is lower than $p_{U,\text{close}}$.

In the latter two cases (collisions with other UAVs), the status of the other UAV is set to collidedtoo.

Upon the check for collisions, each UAV position is updated according to the following procedure. If the UAV status is $s = \text{collided}$ and $z > 0$, then $(x, y, z) \to (x, y, z - 1)$—i.e., the UAV falls. If the UAV status is $s = \text{alive}$, the new position is determined by the controller output $m \in \mathcal{M} = \{+1_x, -1_x, +1_y, -1_y, +1_z, -1_z, \varnothing\}$, where $m = +1_x$ results in $(x, y, z) \to (x + 1, y, z)$, $m = -1_x$ results in $(x, y, z) \to (x - 1, y, z)$, …, $m = \varnothing$ results in $(x, y, z) \to (x, y, z)$. Otherwise (i.e., if the UAV is collided and already on the ground), the position remains unchanged. In other words, the speed of an UAV is at most 1 cell for each time step.

B. UAVs controller

The UAV *controller* is an algorithm which, at each time step, processes an input corresponding to the UAV view of the space around it in order to produce an output $m \in \mathcal{M}$ which itself determines how the UAV position will be updated (see Section II-A). The controller may be stateful (that is, its output may depend also on an internal state which is the result of previous processing) and non-deterministic.

More in detail, the input of the controller of a UAV u consists of:

1) the set $\mathcal{B}$ of buildings and the size $l_x^{\mathcal{S}}, l_y^{\mathcal{S}}, l_z^{\mathcal{S}}$ of space $\mathcal{S}$;
2) the positions of all the other UAVs whose positions belong to the cube with side length equal to $2v + 1$ and center in u position (i.e., the position (x', y', z') of each UAV such that $x' \in [x - v, x + v]$, $y' \in [y - v, y + v]$, and $z' \in [z - v, z + v]$);
3) the target cell $(x_f, y_f, 0)$ of u;
4) a function $r : \mathcal{S} \to \{\text{allowed}, \text{denied}\}$ which maps each cell of the space to a binary value representing if that cell, at the current time step and by the UAV u, may (allowed) or may not (denied) be occupied.

The four components of the input represent, respectively: (i) an unlimited knowledge of the non-moving objects in the space (i.e., buildings)—in other words, we assume that a map of the space is available to the controller; (ii) the goal of the controller itself; (iii) a limited knowledge of moving object (i.e., other UAVs) around the UAV; (iv) the knowledge of the current effects of the rules, from the point of view of the UAV itself.

With respect to the function r defining allowed and denied regions of the space, we remark three considerations—the description about how r is determined is given in Section II-C.

First, the function itself may be different for different UAVs or even for the same UAV at different time steps: for instance, a rule stating, informally, that "a minimum safety distance should be kept to the closest UAV" will result in different denied regions at different time stamps, if the closest UAV is moving.

Second, since we chose to model the concept of regulation with a function r which, basically, marks some regions of the space as denied, instead of marking the actions themselves as denied (e.g., "you cannot move up"), our choice may be easily extended to more sophisticated abstractions, such as, e.g., continuous space, or UAVs moving at different speeds.

Third, the denied regions defined by r are indeed just an input of the controller: a specific controller may actually ignore this input. This allows to model, and hence investigate, interesting interactions involving the controller trade-off between need to reach the target cell and will of complying with the rules and global indexes such as traffic efficiency and safety.

In the present study, we considered two different controllers, whose descriptions follow.

1) A controller:* A* is an algorithm commonly used for path-finding [10], which is an extension of the Dijkstra's algorithm. Given a start position, A* explores all the possible paths leading to the target position, considering at first the ones which have lower costs. Because of this reason, this algorithm is able to find the shortest path avoiding obstacles, so it is a good candidate as a controller for the UAVs in our model.

In our case, A* considers as an obstacle a cell which belongs to a building, or is occupied by one or more other UAVs, or

is denied by the rules. Since, according to this definition, the obstacles could change with time, this controller re-applies the A* algorithm at each time step.

2) Square-arc (SA) controller: This controller is a simpler controller which realizes the following behavior: (i) reach a safe altitude by moving only up ($m = +1_z$); (ii) if already at a safe altitude, reach the x-coordinate x_f of target cell by moving only on the x axis ($m \in \{+1_x, -1_x\}$); (iii) if already at a safe altitude and with $x = x_f$, reach y_f by moving only on the y axis; (iv) if above the target cell, reach the ground by moving only down ($m = -1_z$).

The safe altitude z_s is determined by considering all the buildings, cells occupied by one or more other UAVs and denied regions whose projection intersect the planned path on the x, y-plane (which is always an "L-shaped" path): z_s is set to the lowest free cell or, in case such definition would lead to $z_s > n_z$, to n_z. Since denied regions could change over the time, the safe altitude is recomputed at each time step: note, however, that the UAV will not go down unless already above the target cell.

C. The rules

We here describe how the function r is built which determines the regions of the space that are denied, at the current time step and for a specific UAV u. In brief, r is built starting from a set $\mathcal{R}$ of rules, expressed according to a predefined language, and evaluated on the base of the UAV context (position, building, other close UAVs).

A *rule* is a pair $R = (P, c)$, where P is a non-empty set of boxes and c is a condition. A *box* is defined by a tuple (x_0, y_0, z_0, l, t, i), where x_0, y_0, z_0 represent the position of the center of the box, $l \geq 0$ determines the side length $2l + 1$ of the box, $t \in \{\text{full}, +_x, -_x, +_y, -_y, +_z, -_z\}$ represents the shape of the box, and $i \in \{\text{incl}, \text{excl}\}$ represents the fact that the center is included or not in the box. The actual shape of the resulting box derives from a cube with side length equal to $2l + 1$ and center in x_0, y_0, z_0, as follows. The cube is full, if $t =$ full, or halved, otherwise: $t = +_x$ means the half with $x \geq x_0$ or $x > x_0$, with i equals to incl or excl, respectively, and similarly for the other values of t—i does not impact the box when $t =$ full. For example, $(5, 5, 5, 2, +_y, \text{excl})$ is the parallelepiped going from $(3, 6, 3)$ to $(7, 7, 7)$, both cells included.

A *condition* is a predicate operating on a set of variable concerning the UAV position, its target cell, the closest other UAV, and close buildings. More in detail, a condition is the constant true or a conjunction of one or more basic conditions or negated basic conditions. A basic condition is a coordinate condition or a distance condition, which respectively operate on (possibly separately) coordinates of the target cell, the UAV, or closest UAV, and distances to the closest UAV or building.

Figure 1 shows the context-free grammar, in the Backus-Naur Form, which defines the language of all possible rules.

The function $r : \mathcal{S} \to \{\text{allowed}, \text{denied}\}$ marking each cell of the space as allowed or denied is defined by a set $\mathcal{R}$ of rules basing on the coordinates x, y, z of the UAV being controlled, its target cell $x_f, y_f, z_f = 0$, its Manhattan distances $d_{U,\text{closest}}$ and $d_{B,\text{closest}}$ to the closest other UAV and building, respectively, and the coordinates $x_{U,\text{closest}}, y_{U,\text{closest}}, z_{U,\text{closest}}$ of the closest other UAVs—in case two or more other UAVs are at the same shortest distance, a random one of them is considered. Note that the distances and the closest UAV coordinate can be obtained by processing the controller input which includes (see Section II-B) the set of all buildings $\mathcal{B}$ and the coordinates of all other close UAVs. The subset of $\mathcal{S}$ to be set to denied is determined as follows: for each $R \in \mathcal{R}$, if the condition is met, all the cell belonging to the boxes are set to denied.

$$
\begin{aligned}
R &::= (\{\langle\text{boxes}\rangle\}, \langle\text{conditions}\rangle)\\
\langle\text{boxes}\rangle &::= \langle\text{boxes}\rangle, \langle\text{box}\rangle \mid \langle\text{box}\rangle\\
\langle\text{box}\rangle &::= (\langle\text{x-coord}\rangle, \langle\text{y-coord}\rangle, \langle\text{z-coord}\rangle, \langle\text{digit}\rangle,\\
&\qquad \langle\text{shape}\rangle, \langle\text{center-inclusion}\rangle)\\
\langle\text{digit}\rangle &::= 0 \mid 1 \mid 2 \mid 3 \mid 4 \mid 5 \mid 6 \mid 7 \mid 8 \mid 9\\
\langle\text{shape}\rangle &::= \text{full} \mid +_x \mid -_x \mid +_y \mid -_y \mid +_z \mid -_z\\
\langle\text{center-inclusion}\rangle &::= \text{incl} \mid \text{excl}\\
\langle\text{x-coord}\rangle &::= x \mid x_f \mid x_{U,\text{closest}} \mid \langle\text{digit}\rangle\langle\text{digit}\rangle\\
\langle\text{y-coord}\rangle &::= y \mid y_f \mid y_{U,\text{closest}} \mid \langle\text{digit}\rangle\langle\text{digit}\rangle\\
\langle\text{z-coord}\rangle &::= z \mid z_f \mid z_{U,\text{closest}} \mid \langle\text{digit}\rangle\langle\text{digit}\rangle\\
\langle\text{conditions}\rangle &::= \langle\text{conditions}\rangle \wedge \langle\text{condition}\rangle \mid \langle\text{condition}\rangle \mid \text{true}\\
\langle\text{condition}\rangle &::= \langle\text{basic-condition}\rangle \mid \neg\langle\text{basic-condition}\rangle\\
\langle\text{basic-condition}\rangle &::= \langle\text{coord-condition}\rangle \mid \langle\text{dist-coondition}\rangle\\
\langle\text{coord-condition}\rangle &::= \langle\text{x-coord}\rangle\langle\text{op}\rangle\langle\text{x-coord}\rangle \mid \langle\text{x-coord}\rangle\langle\text{op}\rangle\langle\text{digit}\rangle\langle\text{digit}\rangle \mid\\
&\qquad \langle\text{y-coord}\rangle\langle\text{op}\rangle\langle\text{y-coord}\rangle \mid \langle\text{y-coord}\rangle\langle\text{op}\rangle\langle\text{digit}\rangle\langle\text{digit}\rangle \mid\\
&\qquad \langle\text{z-coord}\rangle\langle\text{op}\rangle\langle\text{z-coord}\rangle \mid \langle\text{z-coord}\rangle\langle\text{op}\rangle\langle\text{digit}\rangle\langle\text{digit}\rangle\\
\langle\text{op}\rangle &::= = \mid <\\
\langle\text{dist-condition}\rangle &::= \langle\text{dist}\rangle \leq \langle\text{digit}\rangle\\
\langle\text{dist}\rangle &::= d_{U,\text{closest}} \mid d^{+_x}_{B,\text{closest}} \mid d^{-_x}_{B,\text{closest}} \mid d^{+_y}_{B,\text{closest}} \mid d^{-_y}_{B,\text{closest}} \mid\\
&\qquad d^{+_z}_{B,\text{closest}} \mid d^{-_z}_{B,\text{closest}}
\end{aligned}
$$

Fig. 1. Backus-Naur Form of the context-free grammar for the traffic rules.

D. Examples of rules

In order to ease the comprehension of the rule syntax and semantics, we provide here three examples of (set of) rules which express realistic limitations or constraints that can be concisely, yet ambiguously, expressed using the natural language. The examples also allow to appreciate the expressiveness and conciseness of the our proposed language.

$$
\begin{aligned}
P =& \{(x, y, z, 1, -_z, \text{excl}), (x, y, z, 1, -_x, \text{excl}),\\
&(x, y, z, 1, +_x, \text{excl}), (x, y, z, 1, -_y, \text{excl}),\\
&(x, y, z, 1, +_y, \text{excl})\}\\
R_1 =& (P, z < 4 \wedge \neg x = x_f)\\
R_2 =& (P, z < 4 \wedge \neg y = y_f)
\end{aligned}
$$

The two rules above express together the constraint saying "flight at a min altitude of 4". Intuitively, the rules enforce a minimum altitude of $z = 4$ by combining 5 half boxes, the only excluded box (among the 6 possible halves) being the above half ($t = +_z$), which results in all the cells "around" the UAV

to be denied, with the exception of those being above the UAV. The constraint takes two rules, which differ in the condition, to be expressed. This reflects the fact that the constraint has not to be applied when the current z of the UAV is already ≥ 4, nor when the UAV is currently above its target cell (i.e., when $x = x_f \wedge y = y_f$).

$$R = (\{x_{U,\text{closest}}, y_{U,\text{closest}}, z_{U,\text{closest}}, 1, \text{full}, \text{incl}\}, \text{true})$$

The rule above expresses the concept of "keep a min distance of 2 from other UAVs". This rule is simple since it has no condition (i.e., the boxes are always present, provided that at least one other UAV is viewed by the current UAV) and the denied region is determined by a single box.

$$\begin{aligned}
R_{-_x} &= (\{(x, y, z, 1, -_x, \text{excl})\}, d^{-_x}_{B,\text{closest}} \leq 2) \\
R_{+_x} &= (\{(x, y, z, 1, +_x, \text{excl})\}, d^{+_x}_{B,\text{closest}} \leq 2) \\
R_{-_y} &= (\{(x, y, z, 1, -_y, \text{excl})\}, d^{-_y}_{B,\text{closest}} \leq 2) \\
R_{+_y} &= (\{(x, y, z, 1, +_y, \text{excl})\}, d^{+_y}_{B,\text{closest}} \leq 2) \\
R_{-_z} &= (\{(x, y, z, 1, -_z, \text{excl})\}, d^{-_z}_{B,\text{closest}} \leq 2)
\end{aligned}$$

Finally, the 5 rules above express together the constraint saying "keep a min distance of 2 from buildings". The key idea is to impose a denied half cube on a given direction and centered (with center excluded) on the UAV whenever the UAV is too close to a building in that direction. The rules are 5 instead of 6 because our model for the buildings makes impossible the event of an UAV moving below a building.

III. Experimental evaluation

We performed a thorough set of experiments with a twofold aim. First, validate our model for the space, buildings, UAVs, and collisions. Second, verify the ability of our proposed rules syntax and semantics to express a set of rules which can effectively impact on the (simulated) UAV traffic.

To this end, we implemented a simulator and run a number of simulations by varying the conditions and the amount of injected traffic. In particular, we considered a single space $\mathcal{S}$ with $l^{\mathcal{S}}_x = l^{\mathcal{S}}_y = l^{\mathcal{S}}_z = 10$ and 4 sets $\mathcal{B}$ of buildings, such that the ratio of space occupied by buildings was $\gamma_B = \frac{\sum_{B \in \mathcal{B}} |B|}{|\mathcal{S}|} \in \{0.01, 0.05, 0.1, 0.15\}$. Each simulation lasted $T = 3000$ steps and was run as follows, at each time step.

1) The current number n of UAVs in the simulation (initially, $n = 0$) and a predefined target number n_{max} of UAVs are compared: if $n_{\text{new}} = r_{\text{new}}(n_{\text{max}} - n) > 0$, a number n_{new} of UAV are generated and added to the simulation. For each new UAV, the initial position is set at $z = 0$ and randomly concerning x and y, avoiding all the cells belonging or adjacent to buildings; the target cell is chosen randomly with the same criteria.
2) Then, the position of each UAV is updated according to the procedure described in Section II-A.
3) Finally, each UAV which meets at least one of the following conditions is removed from the simulation: (a) the UAV is in its target cell; (b) the UAV status is collided and its position belongs to a building; (c) the UAV status is collided and its $z = 0$ (i.e., it hit the ground).

In other words, after a "slow start" aimed at avoiding to crowd the ground at the beginning of the simulation, the number n of UAVs, i.e., the injected traffic, is kept constant.

We experimented with $r_{\text{new}} = 0.2$, $v = 2$, $p_{B,\text{inside}} = 1$, $p_{B,\text{close}} = 0.25$, $p_{U,\text{same}} = 0.75$, and $p_{U,\text{close}} = 0.5$ and performed 30 simulations (with different random seed) for each of the value of $n_{\text{max}} \in \{1, 5, 10, \ldots, 75, 80\}$.

Moreover, in order to investigate the impact of the controller algorithm, we experimented with 3 different policies for the choice of the controller algorithm when generating a new UAV: always A*, always SA, A* or SA with equal probability (we denote this case by "Mix"). Finally, we repeated the experiments by considering the case of no rules ($\mathcal{R} = \emptyset$) and a case with $\mathcal{R}$ including the example rules of Section II-D. We hence performed a total of $30 \times 17 \times 4 \times 3 \times 2 = 12\,240$ runs.

After each simulation, we measured the overall number $\bar{n}_{\text{collision}}$ of collisions and the overall *ground traveled distance* $\bar{d}_t$: the latter is the sum of the Manhattan distances between the starting cell and the target cell for each UAV which reached the target in the alive status during the simulation. The two indexes represent the safety and the efficiency of the transport system as a whole and depend on the amount n_{max} of injected traffic—clearly, the greater the number of circulating UAVs, the longer the distance, the more frequent the collisions. In order to mitigate this dependency, hence allowing for a different perspective in analyzing the results, we derived two other indexes: the *average accidentality* $\text{AA} = \frac{1}{n_{\text{actual}}} \sum_{j=1}^{n_{\text{actual}}} \frac{n^j_{\text{collision}}}{k^j}$ and the *average ground speed* $\text{AGS} = \frac{1}{n_{\text{actual}}} \sum_{j=1}^{n_{\text{actual}}} \frac{d^j_t}{k^j}$ where n_{actual} is the number of all UAVs which lived in the simulation, $n^j_{\text{collision}}$ is the number of collisions in which the j-th UAV was involved, k^j is the number of time steps it lived in the simulation, and d^j_t is its ground travelled distance (set to 0 for collided UAVs).

A. Results

Figures 2 and 3 present the results of our experimental analysis: for both, each point is the result of the average of the index across the 30 simulations in the same conditions.

Figure 2 shows the average ground speed AGS (above) and average accidentality AA (below) vs. injected traffic n_{max}, in different conditions. Two interesting considerations may be done. First, it can be seen that, as expected, the introduction of the example rules causes a decrease in the accidentality and in the average ground speed: this confirms that our proposed model and rules allow to impact on the trade-off between efficiency and safety of the simulated UAV traffic in a consistent way. Second, it can be seen that, in particular for the AA index, the differences among the three controller policies tend to become negligible when using the rules: in other words, ruling the system appear as a way for making the traffic safety more predictable, regardless of the actual UAV controller.

Figure 3 shows the overall ground travelled distance $\hat{d}_t$ vs. overall number $\hat{n}_{\text{collision}}$ of collisions for the Mix controller

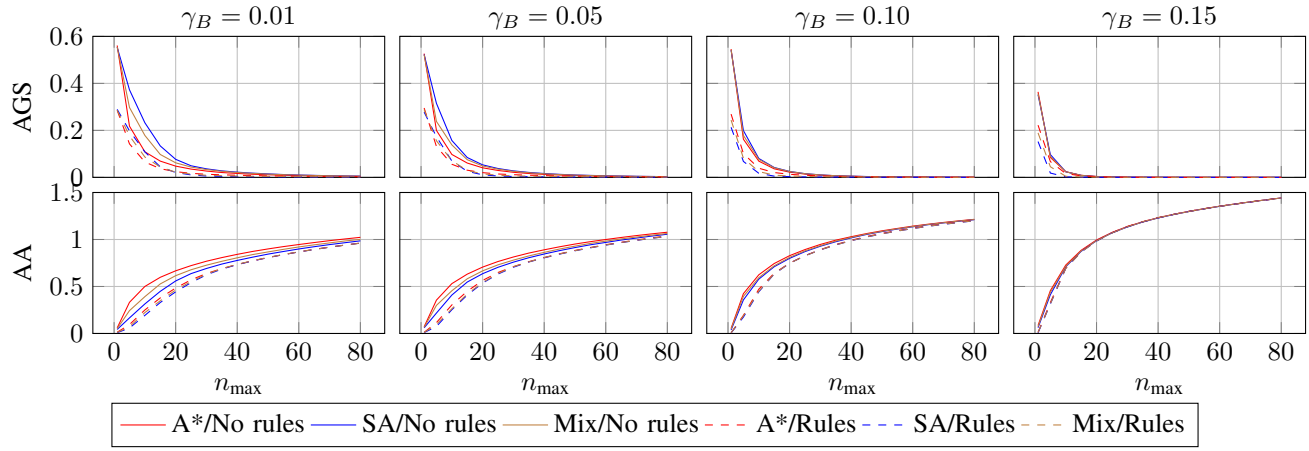

Fig. 2. Average ground speed AGS (above) and average accidentality AA (below) vs. injected traffic $n_{\max}$ for different controllers (line color), rate of buildings γ_B (plot column), and rules (line style).

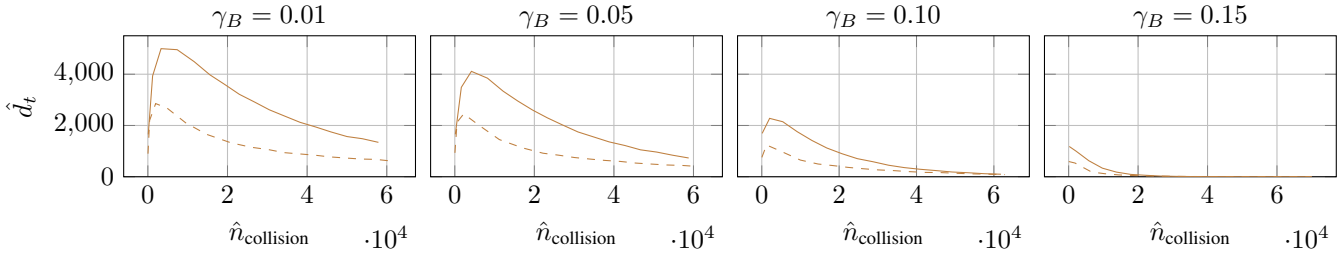

Fig. 3. Overall ground travelled distance $\hat{d}_t$ vs. overall number $\hat{n}_{\text{collision}}$ of collisions for the Mix controller policy, different rate of buildings γ_B (plot column), and rules (line style).

policy, for different values of the rate of buildings γ_B. It can be seen that there is a point (i.e., a value for the injected traffic $n_{\max}$) after which adding more UAVs does not result in an increased $\hat{d}_t$, whereas it leads to an increased number of collisions. This point is the *point of congestion* of the space and, as expected, the overall ground travelled distance (a proxy for the traffic efficiency) decreases with the increase of γ_B: in other words, the lower the free flight space, the lower the overall ground travelled distance a swarm of UAV can serve, regardless of the swarm size.

IV. Concluding Remarks

We have proposed a language (syntax and semantics) for defining rules for UAV traffic in an urban environment, a scenario whose importance is going to greatly increase in the near future, in particular for smart cities. The language builds on a model for space, buildings, UAV controllers, and collisions which we proposed and validated experimentally through a large number of simulations. The experiments show that the introduction of a small set of hand-written rules of realistic complexity, which can be concisely written using our proposed language, leads to consistent and predictable effects on UAV traffic efficiency and safety.

References

[1] H.-P. Lam, S. Thakur, G. Governatori, and A. Sattar, "A Model to Coordinate UAVs in Urban Environments Using Defeasible Logic.," in *RuleML Challenge*, 2009.

[2] E. Medvet, A. Bartoli, and J. Talamini, "Road Traffic Rules Synthesis Using Grammatical Evolution," in *European Conference on the Applications of Evolutionary Computation*, pp. 173–188, Springer, 2017.

[3] J. A. Marin, R. Radtke, D. Innis, D. R. Barr, and A. C. Schultz, "Using a genetic algorithm to develop rules to guide unmanned aerial vehicles," in *Systems, Man, and Cybernetics, 1999. IEEE SMC'99 Conference Proceedings. 1999 IEEE International Conference on*, vol. 1, pp. 1055–1060, IEEE, 1999.

[4] J. F. Gomez and M. Jamshidi, "Fuzzy adaptive control for a UAV," *Journal of Intelligent & Robotic Systems*, vol. 62, no. 2, pp. 271–293, 2011.

[5] S. Khantsis and A. Bourmistrova, "UAV controller design using evolutionary algorithms," *Lecture notes in computer science*, vol. 3809, p. 1025, 2005.

[6] B. M. Sathyaraj, L. C. Jain, A. Finn, and S. Drake, "Multiple UAVs path planning algorithms: a comparative study," *Fuzzy Optimization and Decision Making*, vol. 7, no. 3, pp. 257–267, 2008.

[7] T. Liao, *UAV collision avoidance using a* algorithm*. PhD thesis, 2012.

[8] E. Sunil, J. Hoekstra, J. Ellerbroek, F. Bussink, D. Nieuwenhuisen, A. Vidosavljevic, and S. Kern, "Metropolis: Relating airspace structure and capacity for extreme traffic densities," in *ATM seminar 2015, 11th USA/EUROPE Air Traffic Management R&D Seminar*, 2015.

[9] R. E. Weibel and R. J. Hansman, "Safety considerations for operation of different classes of UAVs in the NAS," in *AIAA 4th Aviation Tehcnology, Integration and Operations Forum, AIAA 3rd Unmanned Unlimited Technical Conference, Workshop and Exhibit*, 2004.

[10] P. E. Hart, N. J. Nilsson, and B. Raphael, "A formal basis for the heuristic determination of minimum cost paths," *IEEE transactions on Systems Science and Cybernetics*, vol. 4, no. 2, pp. 100–107, 1968.