A Framework for Analyzing Adaptive Autonomous Aerial Vehicles

Ian A. Mason[1], Vivek Nigam[2,3], Carolyn Talcott[1], Alisson Brito[2]

[1] SRI International, Menlo Park, USA, iam@csl.sri.com, clt@csl.sri.com
[2] Federal University of Paraíba, João Pessoa, Brazil, alisson@ci.ufpb.br
[3] fortiss, Munich, Germany vivek.nigam@gmail.com

Abstract. Unmanned aerial vehicles (UAVs), a.k.a. drones, are becoming increasingly popular due to great advancements in their control mechanisms and price reduction. UAVs are being used in applications such as package delivery, plantation and railroad track monitoring, where UAVs carry out tasks in an automated fashion. Devising how UAVs achieve a task is challenging as the environment where UAVs are deployed is normally unpredictable, for example, due to winds. Formal methods can help engineers to specify flight strategies and to evaluate how well UAVs are going to perform to achieve a task. This paper proposes a formal framework where engineers can raise the confidence in their UAV specification by using symbolic, simulation and statistical and model checking methods. Our framework is constructed over three main components: the behavior of UAVs and the environment are specified in a formal executable language; the UAV's physical model is specified by a simulator; and statistical model checking algorithms are used for the analysis of system behaviors. We demonstrate the effectiveness of our framework by means of several scenarios involving multiple drones.

1 Introduction

Unmanned aerial vehicles (UAVs), a.k.a. drones, have gained much attention in recent years not only for entertainment purposes, but also being used to carry out non-trivial tasks [37,22,24]. For example, UAVs are being used to autonomously deliver packages, monitor railroad tracks [6] and electricity lines, precision agriculture [10], automating inventory checking in large warehouses [31], and even air taxis [11]. The main reason for this increased interest derives from reduced costs and from the great improvement of UAV's control and flight mechanisms/algorithms.

Despite these successful applications, less attention has been given to how to devise a strategy for a UAV that is going to (autonomously) carry out a task. The current practice is, before UAVs are deployed and test flights are performed, to use simulators such as ArduPilot/SITL [34] to carry out simulations in order to check whether the strategy devised to accomplish the given task is sensible. While simulators implement reasonably faithful physical models of UAVs, including energy consumption, velocity and acceleration, etc, simulations many times do not take into account the unpredictable effect of winds and failures, such as GPS or equipment failures. Moreover even when such aspects are included in the simulation, only a few simulations are carried out, rather than systematic sampling of possibilities. This leads to very low coverage of the situations that might be encountered using the proposed strategy, possibly resulting in the discovery of failure in later stages of development such as during flight test, or worse during operation, when failures are more costly.

On the other hand, executable symbolic models, have been proposed [15] for analysing such systems. These models have been used to verify whether systems satisfy properties such as whether it is possible to carry out a task (realizability) or whether it is never possible for the UAV to fail to carry out a task (survivability). These models typically abstract from details of the physical environment, making simplifications that may be unrealistic, such as deterministic winds and no real account for the physical properties of the drone, such as energy consumption behavior, movement lag, etc.

This paper proposes bridging worlds allowing specifiers to design and specify strategies, analyse them symbolically, and by using the same specifications carry out these analyses using realistic models of the physical behavior of UAVs. Our framework has three main components (detailed in Section 2):

– **Executable Formal Specification of UAV Behavior:** The central piece of our framework is the executable formal specification of UAV behavior. The same specification can be used to carry out a number of analysis (as we describe below). For this paper, we use Soft Agents Framework recently proposed [36,35], but it should also be possible to use other formal agent specification languages. Soft agent strategies are specified symbolically as executable rewrite theories in Maude. An approximate UAV physics including energy consumption model, winds, (de-)accelaration, etc. is also specified as a rewrite theory;
– **UAV Simulator:** An interactive simulator implementing realistic UAV physical models is used to increase the precision of the analysis. For this paper, we use Ardupilot/SITL (or simply SITL) which is an open-source UAV simulator implementing many features, such as realistic UAV physical models (energy consumption, movement behavior, etc), of different types of UAVs, such as copters or airplanes.
– **Statistical Model Checker:** In order to verify systems with uncertainty, our framework also includes a statistical model checker. For this paper, we ported a number of statistical algorithms for statistical reachability analysis [33,21,23]. This simplified carrying out experiments, but in principle it is possible to use existing tools, such as PVeStA and MultiVeStA.

We call our framework $\mathcal{SA}^2$. $\mathcal{SA}^2$ allows combining the three components listed above to carry out different types of analyses of UAV flight strategies (see Section 3):

– **Purely Symbolic:** By using the abstract UAV physics specified in Maude, purely symbolic analyses (without uncertainty) can be carried out to determine (symbolically) whether the specified strategy satisfies properties such as *realizability*, *i.e.*, the goal can be achieved, and *survivability*, *i.e.*, the goal is always achieved [15]. Such analysis can help to find flaws in early stages of development.
– **Simulations:** $\mathcal{SA}^2$ can also simulate the UAV behavior specified as a soft agent system in Maude using SITL. SITL provides a more realistic modeling of UAV physics as opposed to the symbolic (discrete) Maude physics, and supports interaction with external components to receive commands and provide UAV status information (sensor readings);

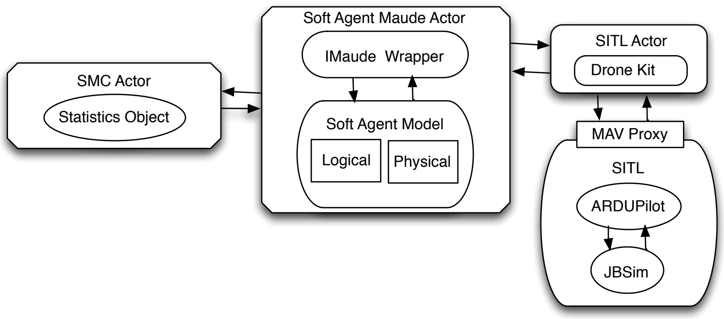

Fig. 1. $\mathcal{SA}^2$'s Architecture

- **Statistical Model Checking:** It is possible to add uncertainty to the environment model, such as winds, sensor failure, etc. Then, by applying the implemented statistical model checking algorithms, specifiers can obtain a quantitative evaluation of how well UAVs perfomed with a specified confidence. For example, what is the minimum energy that UAVs reached before landing; how many were able to return back home, or how much of the goal was achieved.

Our main contribution (Sections 2,3) is thus to demonstrate how different verification methods (symbolic, simulation and SMC) can be integrated and used to carry out different analysis in different stages of UAV development. We discuss related work in Section 4 and point to future work in Section 5.

Running Example: Consider as running example, a set of n drones that start from possibly distinct home bases. Each drone d is assigned a set of points $\mathcal{P}_d$. Their goal is to visit their set of points and return close to home without running out of energy.

2 Soft Agents Squared ($\mathcal{SA}^2$)

As already mentioned our framework has three key components: (1) a formalism for executable symbolic specification of UAV behavior; (2) a UAV simulator; and (3) Statistical Model Checking (SMC) algorithms/tools. As we show in more detail in Section 3, the combination of these three components allows specifiers to develop and test UAV strategies in different development stages thus increasing the confidence in the proposed UAV strategies for accomplishing some task. In this paper, we report experience using a specific instance of the framework called Soft Agents Squared ($\mathcal{SA}^2$). This instance uses the Soft Agents framework [35,36] as the executable specification language implemented in Maude [7], Ardupilot/SITL [34] for the UAV simulator, and SMC algorithms ported from [33,21,23]. We expect that other tools could be used to instantiate the framework as well. As shown in Figure 1 the components are integrated by message passing. For this we use the IOP (InterOPerability) framework [26] which is based on the actor model. [4] Each component in embedded in an actor that coordinates its interactions with the other components. The SMC actor is responsible for managing interactions with the statistical model checking algorithms, the Maude Soft Agent actor coordinates interactions with

[4] IOP binaries and documentation are available at `https://jlambda.com/~iop/`

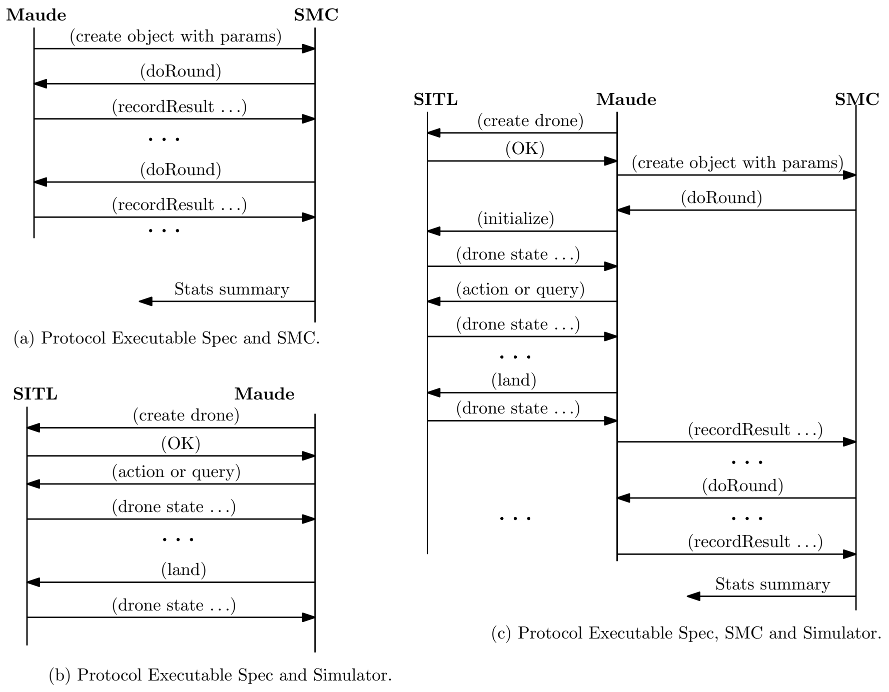

Fig. 2. Messages exchanged used between the Executable Specification, Simulator and Statistical Model Checker (SMC).

Soft Agent specifications, and the SITL actor manages creation and interaction with ArduPilot/SITL drone instances.

2.1 Combining Symbolic Reasoning, Simulation, and Statistical Model Checking

Our framework supports analysis from different perspectives useful at different stages of system design and operation: Soft Agents alone, Soft Agents + SMC, SITL alone, Soft Agents + SITL, or Soft Agents + SITL + SMC.

At the center is the executable specification. Soft Agents alone can be used to carry out symbolic analysis. For example, the Maude search engine can be used for reachability analysis, looking for executions satisfying or violating given properties. This is useful in early stages to see that proposed strategies at least work under ideal conditions. Uncertainty can be added to the environment model (for example wind, GPS or motor failure ...), and Soft Agents + SMC can be used to evaluate the probability of successful behavior under different unpredictable environment, sensor and actuator models. SMC can be tuned to achieve increasing precision/confidence at the cost of more time/executions. Our framework allows users to carry out executions of scenarios specified as Soft Agents dwhere the effects of commands are computed by SITL's more realistic simulation, rather than using the abstract logical model. STIL also supports visualization of drone trajectories useful for identifying certain problems. Finally, all three components can be used together (Soft Agents + SITL + SMC) in order to analyze scenarios with uncertainty using more realistic physical models.

Figure 2 depicts the protocols used to communicate between the Maude (Soft Agents), SITL and SMC actors in the different combinations. In each case the Maude actor inititates the analysis. Figure 2(a) shows the interaction of the Maude actor and the SMC to carry out statistical model checking for a specific scenario. The Maude actor sends a message (create object with params) to SMC to create a statistics object for the desired statistical algorithm. Params include the algorithm identifier, error bounds, and confidence levels. SMC then sends a message (doRound) to the Maude actor to execute the scenario being analyzed. When the execution finishes, Maude sends a (recordResult . . .) message to SMC containing the results of the execution (energy remaining, number of points visited, . . .) to be recorded by the statistics object. The (doRound,recordResult) loop is repeated until the statistics object determines that enoungh rounds have been done. At this point the Stats summary is reported. The result summary is also available by programatic query to the SMC actor.

Figure 2(b) shows the interactions of the Maude and SITL actors to execute a scenario. The Maude actor initiates the execution by sending a (create drone) message to the SITL actor for each drone in the scenario. In the figure we consider a scenario with a single drone. Once the drone object has been created and initialized, SITL replies to Maude with an (OK) message. Now Maude starts the execution. This consists of a series of (action or query) messages from Maude to SITL followed by (drone state ...) messages from SITL to Maude. The action messages control the drones motion during the scenario. The drone state sent by SITL includes the drone's energy level and position. A scenario parameter determines the frequency at which messages are sent to SITL, typically every 1-3 seconds. A scenario ends with a (land) message to SITL to land the drone followed by a final (drone state ...) report from SITL to Maude.

Figure 2(c) shows the messages exchanged when the three components are combined to carry out statistical model checking using the simulated drone physics. The message exchange is basically a merge of the diagrams of the pairwise combinations shows in Figure 2(a,b). Again we consider messages for a single drone scenario. The Maude actor initializes the analysis by sending a (create drone) message to SITL, and once SITL confirms (OK) the creation of the drone object, Maude initializes the SMC with the configuration parameters (create object with params) for the statistics algorithm. As for the Soft Agent + SMC scenario, the SMC actor then proceeds to drive the analysis, by sending a (doRound) message to Maude. In response Maude initializes SITL (initialize), (re) starting the drone with its initial parameters. Then Maude executes the scenario as in the SoftAgents + SITL combination, sending (action or query) messages and receiving (drone state ...) updates. When the execution is complete (land, drone state ...) Maude sends a (recordResult ...) message to SMC and the (doRound, recordResult) process is repeated until the statistics object determines that the analysis is done. At this point, the SMC displays the result summary.

Although, for simplicity, illustrated the interactions for single drone scenarios, in general there can be mulitple drones. We note that system analysis can be sped up by using a concurrent version of the SMC actor that runs multiple simulations in parallel. In addition to the parameters for the simulation system one only needs to specify the

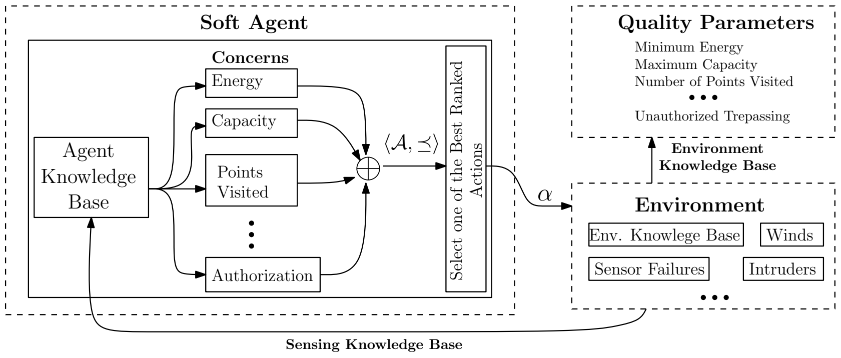

Fig. 3. A Soft Agent uses its sensing knowledge and its policies to decide which actions it will take. Here, $\mathcal{A}$ is the set of possible actions that a soft agent can perform, $\preceq$ is a pre-order on the set $\mathcal{A}$, and $\alpha \in \mathcal{A}$ is the selected action according to the specified concerns. The quality parameters indicate how well a soft agent has performed.

number of parallel subsystems to be started. This can be used for Soft Agent-SMC analyses or for Maude-SITL-SMC combinations. This is particularly useful when SITL is used, as it is does much more computation and hence is much slower than Maude.

In the remainder of this section we provide brief overviews of the specific components we are using.

2.2 Soft Agents

Soft Agents is a rewriting logic framework for specifying and analysing (autonomous) agents. The core framework is specified in the rewriting logic language Maude [7]. It provides generic data structures for representing system state (cyber and physical), interface sorts and functions to be used to specify the environment, agent capabilities and effects of actions (physical), and agent behavior (cyber). The semantics, how a system evolves, is given by a small number of rewrite rules defined in terms of these sorts and functions. Details can be found in [36,35]. Figure 3 depicts the general architecture of a soft agent. A soft agent has a local knowledge base which may include, *e.g.*, its perceived location, energy status, velocity and other data obtained from sensing information. Using the local knowledge base, it decides which action to perform according to its concerns specified as a soft constraint problem [5]. For example, if its energy has reached some low level, activating the energy concern, it may decide to go back home. There may be several possible actions equally preferred in which case one is selected non-deterministically. As soft constraints subsume other constraint systems, *e.g.*, probabilistic, fuzzy, or classical constraint systems [5], it is possible to specify a wide range of decision algorithms.

For our running example, we consider two simple strategies for picking the next point to visit to illustrate the types of analysis enabled by our framework. More advanced features, such as collision avoidance algorithms and its verification can also be specified using Soft Agents.

– **Waypoint Strategy:** A drone d visits the points in $\mathcal{P}_d$ in some pre-specified order;

– **Closest Point Strategy:** A drone d chooses (non deterministically) the next point in $\mathcal{P}_d$ to visit from the set of points it has not yet visited that are closest to its current location.

In both strategies, however, if a drone's energy level reaches some particular caution level, $caution$, the drone heads back home regardless if it has visited all assigned points. Moreover, if its energy level reaches some critical value $critical < caution$, typically 5% of its battery level, the drone lands regardless of whether it is close to its home base. We specify such strategies by defining the concern "Points Visited" and combining it with concern "Energy". The "Energy" concern overrides the "Points Visited" concern whenever a drone's energy level reaches $caution$. Other examples of Soft Agent specifications can be found in [35].

An action's physical effects, *i.e.*, how it changes the position, energy, etc., of a drone, are also specified in the Soft Agent Framework in Maude. For example, the following equation specifies how much energy per time unit a drone consumes depending on its speed:

```
eq costMv(v) = (if v < 3.0 then 1.04 else 1.19 fi) .
```

If the drone's speed is less than 3.0, then it consumes 1.04% of energy per time unit, otherwise 1.19%. There are similar equations specifying a drone's moving model, *e.g.*, (de-)acceleration, top speed. The values in these equations will depend on the particular physical properties of the considered drone. For instance, the values used in the equation above were calibrated to correspond *roughly* to the energy model used by SITL's copter.

Once an action is selected, execution updates the environment's knowledge base, *e.g.*, the actual position of all drones, their energy, speed according to the specified physics. It may happen due to uncertainties that the perceived local knowledge base does not match the actual values stored in the environment knowledge base. Such uncertainties may be caused by non-malicious factors, such as winds and sensor failures, or by malicious intruders. This paper only considers non-malicious factors. (Scenarios with malicious intruders is still subject of intensive research [28].)

Finally, a soft agent system is a collection of soft agents, each one with its own local knowledge base and concerns, interacting with the same environment. They may share knowledge whenever some specified conditions are satisfied.

2.3 Ardupilot/SITL

ArduPilot SITL (Software In The Loop) [34] is a simulator that allows one to simulate a Plane, Copter or Rover without any actual hardware. Ardupilot [1] is a C/C++ autopilot software package for controlling a variety of vehicle systems ranging from conventional airplanes, multirotors, and helicopters,to boats and even submarines. The physics of the simulation is provided by the JSBSim software package [14]. We use the python library DroneKit-SITL to communicate to a ArduPilot SITL instance. The communication uses a MavProxy process that forwards MavLink commands [27] over the local IP network.

The SITL API provides many features that are usually available in actual UAV devices. It is possible to specify the direction, yaw, and velocity of a drone, and to monitor the energy levels, and GPS position of a vehicle and set their operational *modes*, which determine the level of assistance provided by its control mechanisms. (In fact, it is possible to simulate any command available in an normal UAV remote controller.)

SITL also provides an interface with maps and controller information that allow one to easily monitor a vehicle's location and state during a simulation.

2.4 Statistical Model Checking

Our statistical model checker (SMC) supports reachability / counting analysis. It draws on ideas and algorithms from Vesta [33] and XTune [21] (see Section 4). The core of our SMC is the notion of Statistics Object (implemented as an abstract Java class). Each concrete class implements a specific statistical algorithm for computing the expected value of a variable from a sampled set of measurements. A statistics object is created with parameters depending on the underlying algorithm, that generally include some measure of desired precision and/or confidence. It has a method to record the results of a run, the value whose expected value/average is being estimated. This could be as simple as 0 or 1 representing some notion of failure or success, or a tuple of quality parameters. A statistics object also has a method to inquire if the analysis is done (enough runs have been recorded) and to provide a summary of the results, which includes the number of runs recorded, the average of the recorded values, and possibly other information. We have ported the 5 algorithms provided in the NCPS framework [18,17], that builds on the XTune architecture ideas. For simplicity we focus here on the algorithm called GenericApproximation [23]. The parameters are ϵ (the error parameter, using additive approximation) and δ (the confidence parameter). The algorithm computes the number of rounds needed to achieve the precision and confidence specified by these parameters assuming a Bernoulli distribution. In our experiments, we set ϵ and δ to 0.25. Although these confidence/error levels are laughable in usual cases, given the uncertain drone behavior and small number of drones, as we discuss in Section 3, they allow us to get some idea of the range of behaviors in a modest amount of time.

3 Analyses

This section illustrates the main types of analyses that can be carried out within $\mathcal{SA}^2$ using different combinations of its components. The table in Figure 4 summarizes the main (qualitative) properties of the possible analyses.

Recall our example where drones are assigned a set of points and should visit these points and return to home base without running out of energy. Moreover, we consider two different strategies of how to accomplish this task: *Waypoint Strategy* and *Closest Point Strategy*. These strategies come with two parameters $caution$ and $critical$ which specify when a drone should head back home and when it should land (see Section 2).

It is certainly possible to improve both strategies by adding more soft agent concerns and while this is an interesting research question, it is not the purpose of this paper. Our goal is to illustrate the types of analysis one can carry out in $\mathcal{SA}^2$. We use a small scenario with two drones and 8 points. In order to illustrate the analyses with uncertainty, we considered scenarios with varying chances of wind and intensity.

Finally, while we also describe how the different types of analyses available in $\mathcal{SA}^2$ scale with the number of drones, these results should be taken with a grain of salt as

Combination	**Result**	**Speed**	**System Size**	**Physics**	**Type**
SA	yes/no	fast	$\leq$ 10 drones	no uncertainty	Symbolic
SA + SMC	quantitative	fast	$\leq$ 60 drones	possibly uncertain	Statistical
SA + SITL	yes/no	slow	$\leq$ 10 drones	possibly uncertain	Simulation
SA + SMC + SITL	quantitative	slow	$\leq$ 4 drones	possibly uncertain	Statistical

Fig. 4. Qualitative comparison of the analyses available by using different combinations of the components of $\mathcal{SA}^2$. Here **SA** and **SMC** stand, respectively, for soft agents and statistical model checker. Speed is fast if runs take time much smaller (circa 1000 times faster) than actual flight time and slow if runs take time similar to (circa 3 times faster than) actual flight time. The system size is based on experiments carried out using a virtual machine built on top of 2.7 GHz ei5 processor with 8 GB of memory. It should be possible to analyse larger systems with more powerful hardware.

they are preliminary. There are many optimizations that should be investigated and implemented, *e.g.*, how to maximize the number of executions running in parallel. Nevertheless, our preliminary results are promising.

3.1 Soft Agents (Purely Symbolic)

By using Maude's built-in rewrite and search engines, we can analyse scenarios using the physics specified in Maude without uncertainty. Using this machinery is useful in early stages of development as specifiers can early on check whether some fail state can be reached (even without uncertainty), *e.g.*, drones not being able to visit all points or landing far from home or even running out of energy while flying and thus crashing. If so, specifiers may consider using more drones or drones with more energy or rethink the assignment of points.

For our running example, we can specify the following functions in Maude:

- `success(C)`: The configuration `C` is a success if all drones were able to land back home and visit all assigned points;
- `hardFail(C)`: The configuration `C` is a hard fail if at least one drone crashed, *i.e.*, ran out of energy during flight, or a drone landed very far away from its home base;
- `softFail(C)`: The configuration `C` is a soft fail if it is not a hard fail and if all drones were able to reach back home, but at least one point was not visited.

Clearly a hard fail is worse than a soft fail.

Using the Soft Agent machinery, it is possible to check if drones can realize the assigned task by executing the following command where `I` is the initial configuration:

```
search I =>* C such that success(C)
```

Similarly, we can check whether hard or soft fail configurations can be reached.

Indeed while developing the Waypoint and Closest Point strategies, we initially assigned too many points (8 points) to each drone. Then, by using Maude's machinery, we quickly determined that two drones were not able to visit all the assigned points. Therefore, we reduced the number of assigned points to four per drone. If we did not perform this check early on, it is very likely that we would have spent a great deal of time further developing the strategies tuning the "Energy" and "Visited Points" concerns without realizing that it is not possible for the drones to visit so many points.

For small systems such as our running example, Maude can rapidly (less than 20 seconds) return yes/no answers. However, as state space increases exponentially, for larger systems (with more than 10 drones), Maude takes much longer times (more than 2 hours) to traverse the whole search tree.

3.2 Soft Agents and SMC

Combining soft agents and the statistical model checker (SMC) enables users to analyze much larger systems (up to 60 drones) and analyze systems against uncertainties, *e.g.*, changing winds. Moreover, users can also obtain quantitative information on how their system performed. For example:

- **Back home (H):** How many drones on average were able to return back home?
- **Minimum Energy (E):** What is the minimum energy remaining for drones that were able to land (at home or not)?
- **Points Visited (V):** How many points on average were visited by the drones?

The combination of such quality parameters and the soft agents concern-based architecture turns out to be powerful: If the SMC returns low values for minimum energy, the specifier may consider to calibrate the "Energy" concern by increasing $caution$. On the other hand, if the average of number of points visited is low, the specifier can change "Points Visited" concern by assigning less points to a drone and increasing the number of drones or decrease $caution$.

As described in Section 2.4, statistical confidence and other parameters (ϵ and δ) can be configured. Analyses requiring higher confidence take more time as more runs have to be carried out. This does not mean, however, that lower confidence results are not useful for a specifier. They can be used to quickly find problems in the flight strategy and make corrections or fine tuning before running $\mathcal{SA}^2$ for higher confidence results.

For example, setting ϵ and δ set to 0.25, only 17 runs are needed which takes the $\mathcal{SA}^2$ Soft Agents + SMC only 8 seconds. In contrast setting ϵ and δ to 0.05 requires 738 runs taking 500 seconds and returning higher confidence results. As the results below illustrate, lower confidence results can still be useful for tuning flight strategies. When analyzing our running scenario using the Closest Point Strategy and $caution = 40$ in an environment with 25% chance of wind we get values for H, E, V described above:

with $\epsilon = \delta = 0.25$: $H = 2.0, E = 28, V = 7.0$
with $\epsilon = \delta = 0.05$: $H = 2.0, E = 29, V = 6.96$

This means that both drones were able to make it back home with more than 25% energy remaining and were able to visit on average 7 of 8 points. This result suggests to a specifier that in such an environment, drones can adopt a less conservative strategy by reducing their $caution$ level. Running the same scenario with $caution = 20$, we get:

with $\epsilon = \delta = 0.25$: $H = 2.0, E = 20, V = 8.0$
with $\epsilon = \delta = 0.05$: $H = 1.97, E = 20, V = 7.99$

where the drones are able to return home and visit practically all points without reaching dangerously low energy levels. It is possible to carry out such analyses with different $caution$ levels and scenarios with different chances of winds. We postpone due to space restriction showing the results of such analyses to Section 3.4 (see Figure 6) when we use all three components Soft Agents, SMC and SITL.

In practice, a specifier may consider carrying out lower confidence, but still meaningful experiments to quickly find ways to tune strategies and then run higher confidence experiments to further fine tune their strategies. For the example above, the specifier could consider increasing caution slightly in order for both drones to return back home and not 1.97 as the higher confidence results indicate.

Scalability The combination of Soft Agents and SMC does not require many computational resources. This not only means that for small scenarios users obtain results quickly, but also scales well to larger scenarios. For example, while using only soft agents we are able to analyze symbolically scenarios with up to 10 drones, by combining soft agents with the SMC, we are able to analyze scenarios with up to 60 drones. The time for running the analysis is depicted in Figure 5 taking around 3000 seconds with a scenario with 60 drones. Running higher confidence results would take much longer, but may be turned tractable by executing runs in parallel. Such optimizations are left to future work.

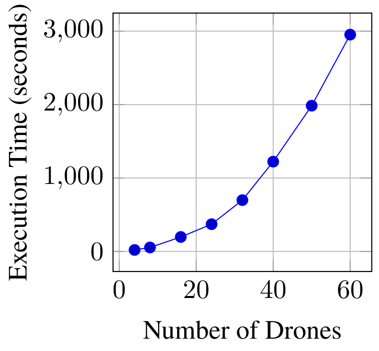

Fig. 5. Execution time for analyzing scenarios with soft agents and the SMC using $\epsilon = \delta = 0.25$.

3.3 Soft Agents and SITL

As described in Section 2, SITL provides faithful physical models of several types of drone along with useful features like Google-like maps and the console with further drone information, *e.g.*, AirSpeed, Battery Level, etc. These features can be helpful in further finding bugs in particular those bugs due to mismatches between the Soft Agents discrete physical model and the more faithful SITL model.

For a concrete example, after analyzing our strategies using SA and the SMC, we ran the same strategies, but using SITL. By observing the map provided by SITL, we noticed that when a drone reached close to an assigned point it would be circling around it not being able to visit it. Such flaw was not noticed using the discrete SA's physical model as the drones were moving discretely from one grid point to the next. In SITL, where drones are not guaranteed to be located at discrete points, the same actions might cause a drone to overshoot a target point. Then, continuing at the same speed it tries to turn and correct, which can result in circling around the point and never reaching it.

Therefore, we needed to modify both flight strategies: whenever a drone is close to a point that it is attempting to visit, it should reduce its speed in order to allow it to make a more careful approach and ultimately be able to visit the point.

The downside of using SITL is that while one simulation using Maude only took seconds, one simulation using SITL takes time similar to the actual flight of drones. (It

Waypoint Strategy (Closest Point Strategy)				
Wind	**Caution**	**Back Home**	**Minimum Energy**	**Points Visited**
12.5% chance	20	1.3 (2,0)	0.0 (36.1)	8.0 (8.0)
	30	2.0 (2.0)	11.3 (36.1)	8.0 (8.0)
	40	2.0 (2.0)	17.6 (40.3)	8.0 (7.6)
25% chance	20	0.6 (1.9)	0.0 (31.1)	8.0 (7.9)
	30	1.3 (2.0)	0.0 (33.5)	8.0 (7.6)
	40	2.0 (2.0)	18 (36.8)	8.0 (7.1)
50.0% chance	20	0.6 (1.6)	0.0 (8.4)	7.3 (6.3)
	30	1.3 (1.6)	3.0 (5.11)	6.3 (4.9)
	40	1.3 (1.8)	3.6 (14.5)	5.6 (4.5)
62.5% chance	20	0.0 (0.3)	0.0 (0.0)	3.6 (3.2)
	30	0.3 (0.5)	0.0 (0.1)	2.0 (3.0)
	40	0.6 (0.8)	0.0 (2.0)	2.0 (2.6)

Fig. 6. Experiments using Soft Agents, SITL and SMC. $\epsilon = \delta = 0.25$.

is possible to increase speed up to be 3 to 4 times faster than actual flight, but simulations still take some minutes.)

3.4 Soft Agents, SITL and SMC

Finally, it is possible to use all of $\mathcal{SA}^2$'s components in order to obtain similar results as described in Section 3.2, but instead of using the discrete physical model, use the more realistic SITL drone model.

The table in Figure 6 contains some of the results obtained by using soft agents in combination with the statistical model checker and SITL. We varied wind chance and considered several values for $caution$. The results for each choice of wind chance and caution takes around 900 seconds, *i.e.*, it is around 110 times slower than using Soft Agents and SMT alone. This is expected as runs take time similar to actual flight.

We observe that, as expected, when increasing the wind chance the performance of the drones deteriorates. While with low chances of wind (12.5%), the drones are able to visit most of the points and safely return home, with higher chances of wind (62.5%) drones visit less than 4 of the 8 assigned points and they fail to return home. Moreover, by increasing the caution level, more drones are able to return to home with higher energy levels, but then the number of visited points reduce.

When comparing the two strategies, in scenarios with less wind the Waypoint Strategy seems to perform better than the Closest Point Strategy in terms of visited points, but at the expense of drones using more energy, even reaching dangerously low levels. This is also reflected in the number of drones that are able to return back home, especially when there is a greater wind chance. On the other hand, the Closest Point Strategy seems more conservative in terms of energy consumption, accomplishing in many cases the task of visiting all eight points and still having high level of energy (above 30%) at the end. This is also reflected in the number of drones returning home.

Given these results, specifiers can further tune the specified strategies. For example, one could set the Closest Point Strategy to be less conservative by tuning the "Energy Concern" as it seems over-conservative.

4 Related Work

Formal executable models can provide valuable tools for exploring system designs, testing ideas, and verifying aspects of a systems expected behavior. Executable models are often cheaper and faster to build and experiment with than physical models, especially in early stages as ideas are developing. For example, [9,8] illustrates the value of using formal executable models in the process of designing and deploying network defenses.

The notion of soft agent system is similar to the notion of *ensemble* that emerged from the Interlink project [12] and that has been a central theme of the ASCENS (Autonomic Service-Component Ensembles) project [2]. In [13] a mathematical system model for ensembles is presented. Similar to soft agents, the mathematical model treats both cyber and physical aspects of a system. A notion of fitness is defined that supports reasoning about level of satisfaction. Adaptability is also treated. While the soft agent framework provides an executable model, the ensembles system model is denotational.

A closely related area is work on Collective Adaptive Systems (CAS) [16]. CAS consist of a large number of spatially distributed heterogeneous entities with decentralized control and varying degrees of complex autonomous behavior that may be competing for shared resources, even when collaborating to reach common goals. CARMA (Collective Adaptive Resource-sharing Markovian Agents) [25] is a language and tool set for modeling CAS. CARMA complements the soft agent framework, focusing on quantitative analysis by simulation, and abstracting from details of agent behavior specification.

XTune [19,20,21] is a framework for designing, analyzing, testing and deploying cross-layer optimization policies. The goal is to find robustly optimal solutions to constraints, possibly conflicting, representing concerns/objectives of different system components. The framework supports analysis of design tradeoffs by a combination of formal methods, simulation, and testing in deployed systems. Formal specifications are analyzed using statistical model checking and statistical quantitative analysis, to determine the impact of resource management policies for achieving desired end-to-end timing/QoS properties. XTune has been applied to the adaptive provisioning of resource-limited distributed real-time systems using a multi-mode multimedia case study.

The Real-Time Maude tool [29,30] has been integrated into the Ptolemy II Discrete Event (DE) modeling system[3]. Real-Time Maude models are automatically synthesized from Ptolemy II design models, enabling Real-Time Maude verification of the synthesized model within Ptolemy II. This enables a model-engineering process that combines the convenience of Ptolemy II DE modeling and simulation with formal verification in Real-Time Maude. Formal verification of Ptolemy II models has been illustrated with several case studies [3].

In our previous work [15,35,36] we considered an ideal (symbolic) model without uncertainties. We introduced and studied a number of purely symbolic problems, such the complexity of the Reachability and Survivability Problems.

Recently [4], we also studied using Ptolemy instead of the soft agent model in conjunction with SITL for analysing flight strategies for visiting a set of points. We did not consider in that work the use of statistical model checking algorithms, but only analyse systems using simulations.

5 Conclusions

This paper introduces $\mathcal{SA}^2$ a framework for analysing drone flight strategies. Drones are specified in the soft agent framework, allowing the drone physics to be modeled logically or using a simulation engine such as ArduPilot/SITL (or simply SITL). $\mathcal{SA}^2$ allows users to carry out a number of analyses, such as symbolic model checking, simulation, and statistical model checking to explore behavior of different flight strategies, identify limitations and improve designs.

There are a number of directions for future work that we are investigating. Until now our system only supports SITL copters. It is possible, however, to support other types of drones, such as SITL airplanes or rovers. It is possible in principle to use other simulators to model other types of vehicles. We plan to carry out experiments using more powerful hardware to analyse larger scenarios using more realistic flight strategies. We are also interested in analyzing scenarios that require communication and co-operation/coordination among different drones, for example to carry out multi-step tasks or for fault tolerance. Furthermore, we would like to go a step further and investigate how our analyses correspond to the results obtained by using drones on the field. Finally, we are also investigating malicious environments with faulty sensors or where intruders try to trick drones, *e.g.*, to think they visited some point, but they actually did not.

Acknowledgments: Nigam was partially supported by Capes and CNPq. This work has been partially developed under contracting of Diehl Aerospace GmbH and Airbus Defense GmbH. Talcott and Mason were partially supported by ONR grant N00014-15-1-2202. Nigam and Talcott were partially supported by Capes Science without Borders grant 88881.030357/2013-01.

References

1. Arduplane, arducopter, ardurover. https://github.com/ArduPilot/ardupilot.
2. Ascens: Autonomic service-component ensembles. http://www.ascens-ist.eu.
3. K. Bae, P. C. Ölveczky, T. H. Feng, E. A. Lee, and S. Tripakis. Verifying hierarchical ptolemy II discrete-event models using real-time maude. *Sci. Comput. Program.*, 77(12), 2012.
4. J. Barros, A. Brito, T. Oliveira, and V. Nigam. A framework for the analysis of UAV strategies using co-simulation. In *SBESC*, 2016.
5. S. Bistarelli, U. Montanari, and F. Rossi. Semiring-based constraint satisfaction and optimization. *J. ACM*, 44(2):201–236, 1997.
6. Why BNSF railway is using drones to inspect thousands of miles of rail lines. http://fortune.com/2015/05/29/bnsf-drone-program/.
7. M. Clavel, F. Durán, S. Eker, P. Lincoln, N. Martí-Oliet, J. Meseguer, and C. Talcott. *All About Maude: A High-Performance Logical Framework*. 2007.
8. Y. G. Dantas, M. O. O. Lemos, I. E. Fonseca, and V. Nigam. Formal specification and verification of a selective defense for tdos attacks. In *WRLA*, 2016.
9. Y. G. Dantas, V. Nigam, and I. E. Fonseca. A selective defense for application layer ddos attacks. In *JISIC*, 2014.
10. J. Das, G. Cross, A. M. C. Qu, P. Tokekar, Y. Mulgaonkar, and V. Kumar. Devices, systems, and methods for automated monitoring enabling precision agriculture. In *CASE*, 2015.
11. Autonomous taxi drones. https://www.forbes.com/sites/parmyolson/2017/02/14/dubai-autonomous-taxi-drones-ehang/#54543d934702.

12. M. Hölzl, A. Rauschmayer, and M. Wirsing. Engineering of software-intensive systems. In *Software-Intensive Systems and New Computing Paradigms*, 2008.
13. M. Hölzl and M. Wirsing. Towards a system model for ensembles. In *Formal Modeling: Actors, Open Systems, Biological Systems*, 2011.
14. The JSBSim flight dynamics model. http://www.jsbsim.org.
15. M. Kanovich, T. B. Kirigin, V. Nigam, A. Scedrov, and C. Talcott. Timed multiset rewriting and the verification of time-sensitive distributed systems. In *FORMATS*, 2016.
16. S. Kernbach, T. Schmickl, and J. Timmis. Collective adaptive systems: Challenges beyond evolvability. In *Fundamentals of Collective Adaptive Systems*. European Commission, 2009.
17. Networked cyber physical systems. http://ncps.csl.sri.com.
18. M. Kim, M.-O. Stehr, J. Kim, and S. Ha. An application framework for loosely coupled networked cyber-physical systems. In *EUC*, 2010.
19. M. Kim, M.-O. Stehr, C. Talcott, N. Dutt, and N. Venkatasubramanian. Combining formal verification with observed system execution behavior to tune system parameters. In *FMOODS* 2007.
20. M. Kim, M.-O. Stehr, C. Talcott, N. Dutt, and N. Venkatasubramanian. A probabilistic formal analysis approach to cross-layer optimization in distributed embedded systems. In *FMOODS* 2007.
21. M. Kim, M.-O. Stehr, C. Talcott, N. Dutt, and N. Venkatasubramanian. XTune: A formal methodology for cross-layer tuning of mobile embedded systems. *Transactions on Embedded Computing Systems*, 2011.
22. Knightscope. http://www.knightscope.com.
23. R. Lassaigne and S. Peyronnet. Probabilistic verification and approximation schemes. *Annals of Pure and Applied Logic*, 152(1–3):122–131, 2008.
24. Liquid robotics. http://liquidr.com.
25. M. Loreti and J. Hillston. Modelling and analysis of collective adaptive systems with carma and its tools. In *SFM 2016*.
26. I. A. Mason and C. L. Talcott. IOP: The InterOperability Platform & IMaude: An interactive extension of Maude. In *WRLA'2004*.
27. MAVLink micro air vehicle marshalling / communication library. https://github.com/ArduPilot/mavlink.git.
28. V. Nigam, C. Talcott, and A. A. Urquiza. Towards the automated verification of cyber-physical security protocols: Bounding the number of timed intruders. In ESORICS, 2016.
29. P. C. Ölveczky and J. Meseguer. Abstraction and completeness for Real-Time Maude. In WRLA, 2007.
30. P. C. Ölveczky and J. Meseguer. Semantics and pragmatics of Real-Time Maude. *Higher-Order and Symbolic Computation*, 20(1–2):161–196, 2007.
31. Inventory robotics. http://www.pinc.com/inventory-robotics-cycle-counting-drones.
32. R. P. . F. T. Rocco De Nicola, Michele Loreti. A formal approach to autonomic systems programming: The SCEL language. *ACM TAAS*, 9(2), 2014.
33. K. Sen, M. Viswanathan, and G. A. Agha. VESTA: A statistical model-checker and analyzer for probabilistic systems. In *QEST 2005*.
34. SITL. http://python.dronekit.io/about/index.html. 2016.
35. C. Talcott, V. Nigam, F. Arbab, and T. Kappé. Formal specification and analysis of robust adaptive distributed cyber-physical systems. In SFM, 2016.
36. C. L. Talcott, F. Arbab, and M. Yadav. Soft agents: Exploring soft constraints to model robust adaptive distributed cyber-physical agent systems. In *Software, Services, and Systems - Essays Dedicated to Martin Wirsing*, pages 273–290, 2015.
37. Drone swarms: The buzz of the future. https://www.vlab.org/events/drone-swarms/.