Formal Specification and Synthesis of Mission Plans for Unmanned Aerial Vehicles

Laura R. Humphrey
Air Force Research Laboratory
Control Science Center of Excellence
WPAFB, OH 45433

Eric M. Wolff
Control and Dynamical Systems
California Institute of Technology
Pasadena, CA 91125

Ufuk Topcu
Electrical and Systems Engineering
University of Pennsylvania
Philadelphia, PA 19104

Abstract

As unmanned aerial vehicle (UAV) missions become increasingly complex, automated tools are needed to assist UAV operators in mission planning and execution. Formal methods such as model checking have recently been used for similar purposes, e.g. motion planning for ground robots. Here, we review applications of formal methods to problems in robot control and multi-agent planning and discuss how related techniques can be expanded to serve as the foundation for improved human-automation UAV intelligence, surveillance, and reconnaissance (ISR) mission planning systems.

Introduction

Unmanned aerial vehicle (UAV) intelligence, surveillance, and reconnaissance (ISR) missions are becoming increasingly complex. Whereas in the past, several operators worked together to plan missions for a single UAV, a single operator is now expected to plan missions for multiple UAVs. This paradigm shift has been enabled by the increased use of autonomy, e.g. autopilots that can execute low-level tasks such as flying between waypoints, loitering over targets, surveilling roads, and monitoring patrol routes. On one hand, this increased use of autonomy allows the operator to more easily manage multiple UAVs at a higher level of abstraction. On the other, interactions and timings between high-level tasks – especially across multiple UAVs – can be difficult for the operator to anticipate and track.

In many ways, these issues are similar to those experienced by system designers. Interactions between various software and hardware components can be difficult to reason about, leading to design errors that are hard to understand and fix. Formal methods – mathematically based languages, techniques, and tools for specifying and verifying hardware and software systems – have been developed to assist designers in analyzing such errors. One such method is *model checking*. Model checking requires two inputs: 1) a set of system requirements that are formalized into mathematically based specifications and 2) a high-level system design that can be represented as a finite-state system model. The model is then checked or verified against the specifications using an automated tool called a *model checker*, and for each violated specification, the model checker outputs a counterexample that demonstrates how the violation occurs.

Copyright © 2014, Association for the Advancement of Artificial Intelligence (www.aaai.org). All rights reserved.

Given these similarities, it seems likely that formal methods such as model checking could also have useful applications in the UAV mission planning domain. For instance, if mission goals and constraints – e.g. tasks that must be coordinated, areas that must be surveilled in a particular order, regions that must be avoided – can be formalized into specifications, and if UAV behaviors can be represented as finite-state models, then model checking could be used as a tool to help operators verify mission plans and reason about errors using counterexamples, as depicted in Figure 1. Alternatively, operators could develop specifications that automated routines would then use to synthesize feasible or optimal mission plans. Both paradigms would provide useful mechanisms for improved human-automation interaction, though here we focus on automated synthesis of mission plans.

In what follows, we first review the use of model checking techniques in robot control and multi-agent planning. Then, we describe how similar techniques can be used to provide an improved framework for human-automation UAV mission planning. Finally, we conclude with a discussion of areas of interest for future work.

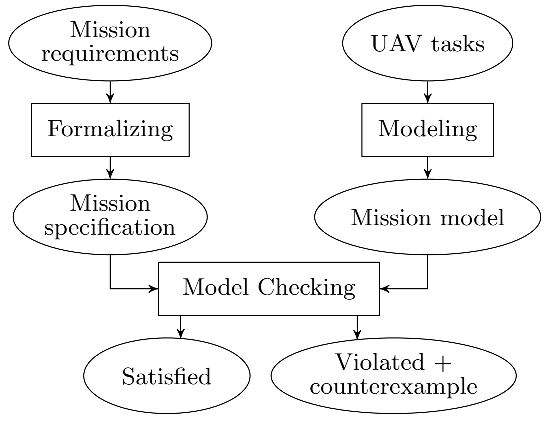

Figure 1: The model checking approach as applied to UAV mission planning. Modified from (Baier and Katoen 2008).

Review

Several areas of research have focused on the application of model checking techniques to robot control and multi-agent planning. For instance, *linear temporal logic* (LTL) – a propositional temporal logic often used to formally express specifications in model checking – has been used as a task specification language for ground robots and UAVs (Finucane, Jing, and Kress-Gazit 2010; Tabuada and Pappas 2006). Given specifications in LTL, one can either use these techniques to verify and provide feedback to an operator that an existing plan satisfies the specifications (Humphrey 2012) or to automatically synthesize a plan that satsifies the specifications. Since tasks in these domains often involve the physical movement of agents, plan synthesis also usually entails path planning. In path planning, it is often desirable to optimize the routes taken by the agents, and one might also consider the avoidance of obstacles or inhospitable areas. The former can be addressed by posing the problem as a mixed integer linear program (MILP) (Karaman and Frazzoli 2008; Wolff and Murray 2013), or if uncertainties in the system or environment exist, as a Markov Decision Process (MDP) (Lahijanian, Andersson, and Belta 2012; Wolff, Topcu, and Murray 2012). For the latter, if certain areas are blocked or inhospitable, one can synthesize plans that minimize the number of violated specifications (Fainekos 2011; Tumova et al. 2013). If continous dynamics of the agents must be considered, then hybrid systems approaches are needed (Wongpiromsarn, Topcu, and Murray 2010; Fainekos, Kress-Gazit, and Pappas 2005). In addition, other lines of research have focused on the synthesis of control protocols that can react to environmental and adversarial disturbances (Liu et al. 2011) and account for unmodeled behaviors in the agents (Topcu et al. 2012).

UAV Mission Planning

We motivate use of these types of model checking techniques for UAV ISR mission planning through discussion of a simple mission scenario. In this scenario, we assume the primary goal is for an operator to use multiple UAVs to monitor a road network, with the constraint that certain areas of the map, i.e. “restricted operating zones” or ROZs, should be avoided. Figure 2 shows a sample map, where waypoints are labelled with numbers, roads connecting waypoints are drawn in black, and ROZs are represented by red polygons.

Since standard model checking techniques work on finite state systems, we must discretize movements of the UAVs. Here, we discretize roads based on the maximum speed achieveable by all UAVs (road points). For many applications, it is also desirable to specify the angle of approach to a waypoint, e.g. for surveilling targets using different look angles. Thus, we also discretize areas around waypoints at 45° angles based on the minimum speed of all UAVs (angle points). For this scenario, UAVs are assumed to move only between waypoints and associated angle points, angle points closest to road points at the beginning and end of a road, and neighboring road points. In some cases, angle points for different waypoints are close enough that no connecting road points are necessary, and in other cases, angle points are close enough that they are aliased to a single point. Here, a maximum speed of 90 km/h, a minimum speed of 30 km/h, and a discrete time step of 60 s are assumed, resulting in 1.5 km between road points and .5 km between angle points and waypoints. Vehicle accelerations are neglected. For details on methods to handle vehicle dynamics in a more realistic manner, see (Wongpiromsarn, Topcu, and Murray 2010; Fainekos, Kress-Gazit, and Pappas 2005).

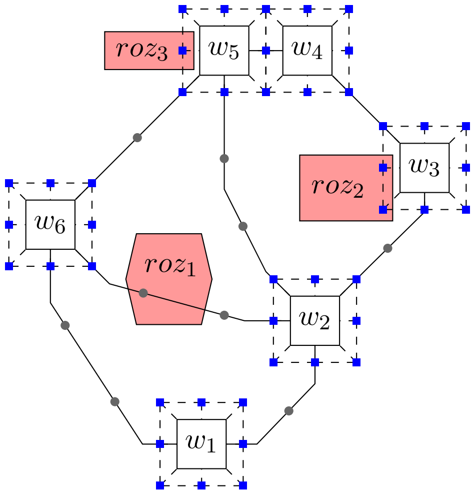

Figure 2: A portion of an example road network.

Given this scenario, we assume a human operator would like to synthesize paths for the UAVs based on a set of mission specifications, preferrably in real-time. In such a situation, there are at least two important aspects that should be considered, including but not limited to: expressing the mission specifications and synthesizing a solution.

Mission Specifications

We use finite transition systems to model the behavior of the UAVs, which we have found to be a useful level of abstraction for high-level mission planning. Given UAV behaviors modeled as transition systems, we can use LTL to specify a wide range of properties relevant to UAV missions.

Definition 1 (Transition System) *A finite transition system is a tuple $TS = (S, R, s_0, AP, L)$ consisting of (i) a finite set of states S, (ii) a transition relation $R \subseteq S \times S$, (iii) an initial state $s_0 \in S$, (iv) a set of atomic propositions AP, (v) and a labeling function $L : S \rightarrow 2^{AP}$.*

We assume the transition system is non-blocking, so for each state $s \in S$, there exists a state $s' \in S$ such that $(s, s') \in R$. Then, a *path* or *run* of the transition system is an infinite sequence of its states, $\sigma = s_0s_1s_2\ldots$ where $s_i \in S$ is the state of the system at index i, and $(s_i, s_{i+1}) \in R$ for $i = 0, 1, \ldots$. A *trace* is an infinite sequence of labels $L(\sigma) = L(s_0)L(s_1)L(s_2)\ldots$ where $\sigma = s_0s_1s_2\ldots$ is a path of the transition system.

Definition 2 (The Grammar of LTL) *LTL formulae over a set AP of atomic propositions are formed according to* *the grammar:*

$$\varphi ::= \mathit{true} \mid a \mid \varphi_1 \land \varphi_2 \mid \neg\varphi \mid \bigcirc\varphi \mid \varphi_1 \mathsf{U}\, \varphi_2$$

where $a \in AP$.

LTL formulae include the standard and derived propositional logic operators – e.g. $\land$ "and," $\neg$ "not," $\lor$ "or," and " $\rightarrow$ "implies," – as well as the temporal operators $\bigcirc$ "next," and U "until." In addition, two common derived temporal operators are $\Diamond$ "eventually" and $\Box$ "always," where $\Diamond a \equiv \text{true}\,\mathsf{U}\,a$ and $\Box a \equiv \neg\Diamond\neg a$. Some model checkers support past time operators like $\bigcirc^{-1}$ "previously." Past research has identified a number of useful LTL specification patterns for agent task planning (Kress-Gazit, Fainekos, and Pappas 2009). These and some additional patterns relevant in the UAV domain are given in Table 1.

Table 1: LTL specification patterns for UAV missions, where p, q, and r are atomic propositions.

Property Name	**Formula**
Safety	$\Box\neg p$
Reachability	$\Diamond p$
Coverage	$\Diamond p \land \Diamond q \land \Diamond r$
Recurrent Coverage	$\Box\Diamond p \land \Box\Diamond q \land \Box\Diamond r$
Sequencing	$\Diamond(p \land \Diamond(q \land \Diamond r))$
Avoidance	$\neg q \,\mathsf{U}\, p$
Avoidance with Reachability	$(\neg q \,\mathsf{U}\, p) \land \Diamond q$
Sequencing with Avoidance	$\neg q \,\mathsf{U}\, p \land \neg r \,\mathsf{U}\, q \land \Diamond r$
Previously	$\bigcirc^{-1} p$
Never After	$\Box(p \rightarrow \bigcirc\Box\neg q)$

For this scenario, let us label waypoints as w_i, where i is the waypoint number; angle points as $a_{i,j}$, where i is the waypoint number and $j = \{1, \ldots, 8\}$ indicates the angle of approach, with 1 at 0° from the horizontal, 2 at 45°, etc; and road points as $r_{i,j,k}$, where i and j with $i < j$ indicate the waypoints at the ends of the road, and $k \geq 1$ numbers the road points starting from waypoint i. Denote the current UAV positions by uav_i for $i = \{1, \ldots, m\}$, where typically $m \approx 4$ for UAV missions. We form atomic propositions about a UAV's location with expressions of the form $uav_i = p_j$, where p_j is a valid waypoint, angle point, or road point. For simplicity in referencing multiple UAVs, we define $(uav = p_j) \equiv (uav_1 = p_j \lor \ldots \lor uav_m = p_j)$. And for simplicity in referencing ROZs, we define $roz_i \equiv (uav_i = z_1 \lor \ldots \lor uav_i = z_n)$, where $\{z_1, \ldots, z_n\}$ is the set of all points that fall inside a ROZ.

Given these definitions, there are a number of specifications that might interest an operator. For instance, the operator might want to establish a repeating patrol route for one UAV (uav_1) using a "recurrent sequencing" specification while using another UAV (uav_2) to surveil a particular set of points using a "coverage" specification, e.g.

$$\Box\Diamond uav_1 = w_3 \land \Box\Diamond uav_1 = w_4 \land \Box\Diamond uav_1 = w_5 \land$$
$$\Diamond uav_2 = w_1 \land \Diamond uav_2 = w_2 \land \Diamond uav_2 = w_6$$

Or, the operator might want to see a given set of points according to a "sequencing" specification using any UAV, e.g.

$$\Diamond\left(uav = w_1 \land \Diamond\left(uav = w_2 \land \Diamond uav = w_3\right)\right) \tag{1}$$

Note (1) does not prohibit any waypoints from being visited before other waypoints; it only requires that certain waypoints be visited after other waypoints. If a UAV were to visit w_2, w_1, w_2 again, then w_3, this would satisfy (1). If the operator wanted a stricter ordering, he could use a "sequencing with avoidance" specification, e.g.

$$\Diamond uav = w_1 \land \Diamond uav = w_2 \land \Diamond uav = w_3 \land$$
$$\neg w_2 \,\mathsf{U}\, w_1 \land \neg w_3 \,\mathsf{U}\, w_2 \tag{2}$$

Note (2) does not prohibit waypoints from being visited more than once. To prevent a waypoint from being repeated, the operator could add a "never after" specification, e.g.

$$\Box(uav = w_1 \rightarrow \bigcirc\Box\neg uav = w_1)$$

The operator can also control the angle of approach to a waypoint in any of these specification patterns using "previously" specifications, e.g.

$$\Diamond(uav = w_1 \land \bigcirc^{-1} a_{1,1})$$

In many scenarios, it is useful to have two or more UAVs approach the same point at the same time from different look angles. If the UAVs are flying at the same altitude, it may also be necessary to ensure they maintain spatial separation. A corresponding "coordination with mutual exclusion" specification would take the form

$$\neg(uav_1 = uav_2 \lor uav_1 = uav_3 \lor uav_1 = uav_4 \lor$$
$$uav_2 = uav_3 \lor uav_2 = uav_4 \lor uav_3 = uav_4) \,\mathsf{U}$$
$$(uav_1 = w_1 \land uav_2 = w_1 \land uav_3 = w_1 \land uav_4 = w_1 \land$$
$$\bigcirc^{-1} uav_1 = a_{1,1} \land \bigcirc^{-1} uav_2 = a_{1,3} \land$$
$$\bigcirc^{-1} uav_3 = a_{1,5} \land \bigcirc^{-1} uav_4 = a_{1,7}) \tag{3}$$

For any of the aforementioned specifications, the operator may also want to add a "safety" specification so that all UAVs avoid all ROZs, e.g.

$$\Box\neg(roz_1 \lor \ldots \lor roz_4)$$

Synthesis of Feasible Mission Plans

Once the operator develops a set of mission specifications, model checking techniques can be used to synthesize a feasible solution. Normally, model checkers are used to ensure a system TS satisfies a set of specifications φ, written

$$TS \models \varphi$$

If not, the model checker returns a trace $L(\sigma) \in traces(TS)$ that does not satisfy the specifications, i.e. $L(\sigma) \not\models \varphi$. If we negate the original specifications, the model checker will return a counterexample such that

$$L(\sigma) \not\models \neg\varphi \rightarrow L(\sigma) \models \varphi$$

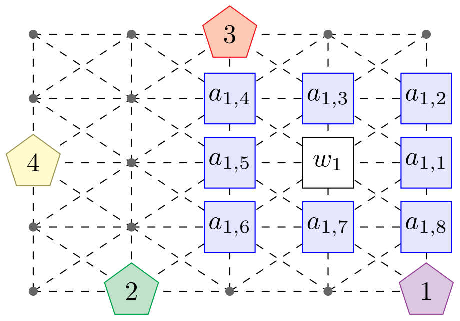

Figure 3: An example 5x5 grid problem, where pentagons represent UAVs and dashed lines represent allowed transitions at each time step.

Table 2: Time in seconds to compute feasible solutions with 4 UAVs on a 4x4 grid and a 5x5 grid.

Mut-ex	BDD 4x4	SAT 4x4	BDD 5x5	SAT 5x5
2	0.102 s	0.144 s	0.554 s	0.209 s
3	4.32 s	0.087 s	51.1 s	0.244 s
4	10.5 s	0.076 s	492 s	0.277 s

This provides a feasible solution for the specifications.

Here, will explore this synthesis procedure using the model checker NuSMV (Cimatti et al. 2002). NuSMV implements two classes of algorithms for model checking of LTL specifications: one that uses satisfiability solvers to search for counterexamples (SAT-based), and one that uses binary decision diagrams (BDD-based). Consider a "coordination with mutual exclusion" specification similar to the one given in (3), but with a varying number of mutual exclusion terms of the form $\Box\neg(uav_i = uav_j)$ and over a grid of the type shown in Figure 3. Table 2 shows the amount of time needed to synthesize a feasible solution with 4 UAVs over a 4-by-4 grid and a 5-by-5 grid using the BDD-based algorithm and the SAT-based algorithm. These results show that for this type of specification, the SAT-based algorithm is faster and less sensitive to the number of UAVs. It also performs better with more mutual exclusion constraints in the specification.

In contrast, consider a simpler "coordination" specification without the "mutual exclusion" constraints, e.g. because the UAVs are altitude deconflicted. This has the form

$$\begin{aligned}\Diamond(uav_1 = w_1 \wedge uav_2 = w_1 \wedge uav_3 = w_1 \wedge uav_4 = w_1 \wedge \\ \bigcirc^{-1} uav_1 = a_{1,1} \wedge \bigcirc^{-1} uav_2 = a_{1,3} \wedge \\ \bigcirc^{-1} uav_3 = a_{1,5} \wedge \bigcirc^{-1} uav_4 = a_{1,7})\end{aligned}$$

Figure 4 shows the amount of time needed to compute solutions for this simpler specification for a varying number of UAVs and varying grid size using the BDD-based algorithm, and Figure 5 shows the amount of time needed by the SAT-based algorithm. Interestingly, the SAT-based algorithm is now much more sensitive to the number of UAVs and is also slower than the BDD-based algorithm. This demonstrates the importance of the algorithm used to synthesize the mission plans, with performance potentially varying depending on the types of specifications being synthesized.

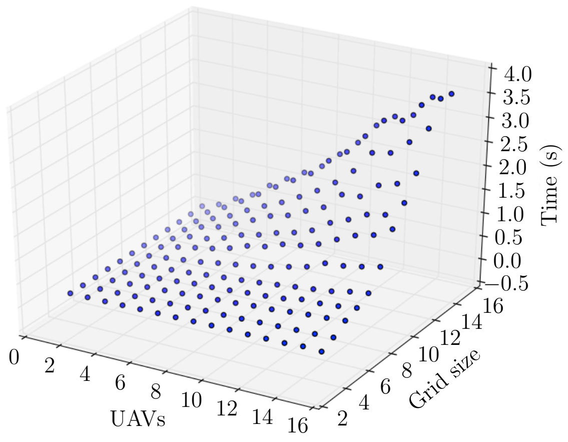

Figure 4: Timing results for the "coordination" specification using a BDD-based algorithm.

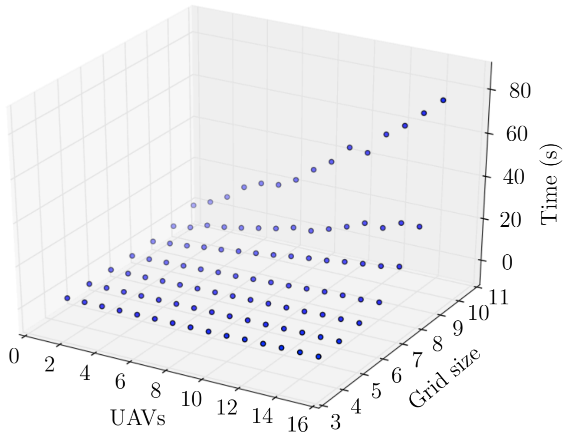

Figure 5: Timing results for the "coordination with mutual exclusion" specification a SAT-based algorithm.

Synthesis of Optimal Mission Plans

Standard model checking techniques only synthesize feasible solutions. Techniques to generate mission plans that are optimal with respect to various cost functions have recently been developed. These techniques are based on dynamic programming (Smith et al. 2011; Wolff, Topcu, and Murray 2012) and mixed-integer linear programming (Karaman and Frazzoli 2008; Wolff and Murray 2013). Such methods are highly relevant for UAV mission planning, which often requires fast execution or minimization of resources like fuel. An important question for synthesis of human-automation mission plans is the cost versus quality tradeoff between feasible and optimal solutions. In our experience, operators will not tolerate automated routines that are too slow, and they will reject solutions that are poor. Thus, we use the map in Figure 2 to run a small number of benchmarks to begin to gauge these tradeoffs.

For these benchmarks, we assume four UAVs with starting positions $uav_1 = a_{1,1}$, $uav_2 = a_{4,2}$, $uav_3 = a_{5,7}$, and $uav_4 = a_{6,7}$. We then generate formulas of varying length based on conjunctions of a random mix of "coverage" and "recurrent coverage" terms of the form $\Diamond w$ and $\Box\Diamond w$, where "formula length" refers to the total number of coverage and recurrent coverage terms in the formula. For each formula length, we examine three cases: a "vehicle agnostic" case, a "vehicle specific" case, and a "mutual exclusion" case. For the vehicle agnostic cases, terms take the form $\Diamond \bigvee_{i=1}^{4}(uav_i = w_j)$ or $\Box\Diamond \bigvee_{i=1}^{4}(uav_i = w_j)$, where each point w_j is chosen randomly without replacement from the set of all map points, excluding UAV starting positions. For the vehicle specific cases, we take the same formulas but randomly choose a specific UAV to associate with each term, i.e. terms take the form $\Diamond(uav_i = w_j)$ or $\Box\Diamond(uav_i = w_j)$. The mutual exclusion cases add three random but unique constraints of the form $\Box\neg(uav_i = uav_j)$ with $i \neq j$ to the vehicle agnostic cases. Mission cost is measured according to the longest path taken by any UAV, where the length of a UAV's path is the number of transitions it takes through the road network, neglecting "loiters" or self-transitions. Paths that satisfy recurrent coverage terms end with a series of transitions that must be infinitely repeated. We can thus think of a path as a concatenation of a prefix path and an infinitely-repeated suffix path. Total path length is the length of the prefix plus the length of the suffix, which is only counted once.

For synthesizing feasible results, we use NuSMV's BDD- and SAT-based algorithms, and for optimal results, we solve an equivalent integer programming (IP) problem using CPLEX (Wolff and Murray 2013). The SAT and optimal solvers both require a bound on total solution lengths, which was set to 40. The optimal solver also requires a maximum prefix length, which was set to 13. These values were sufficient for most trials. Figure 6 shows benchmark solution costs and computation times, based on a median of 10 trials run on a standard desktop. For BDD- and SAT-based synthesis, trials exceeding 600 s were terminated, and for optimal synthesis, trials exceeding 60 s were termined. Trials over these limits were rarely observed to return a result. Results are reported for formula lengths in which at least 5 solutions were returned within the time limit. For longer formula lengths, timing results are set to 600 s to provide a visualization of the drastic increase in computation time.

As these results show, the SAT solver tends to be the fastest, and it provides solutions with costs substantially lower than the BDD solver. An exception is the vehicle specific case, in which the BDD solver is substantially faster than the SAT solver, though solution costs are still much higher. The SAT solver also often provides solutions with slightly lower cost than the optimal solver, most likely because the prefix length of the optimal solver is set to a suboptimal value. The optimal solver returns solutions with low cost but can only generate solutions for relatively short formulas in a reasonable amount of time. Note that feasible rather than optimal results can also be synthesized using an IP formulation, and cost and computation times were found to be comparable to those of the SAT solver (results not shown). Thus, though more thorough testing and analysis is required, it appears a SAT-based solver or possibly a feasible IP-based solver could provide a good tradeoff between computation time and solution quality.

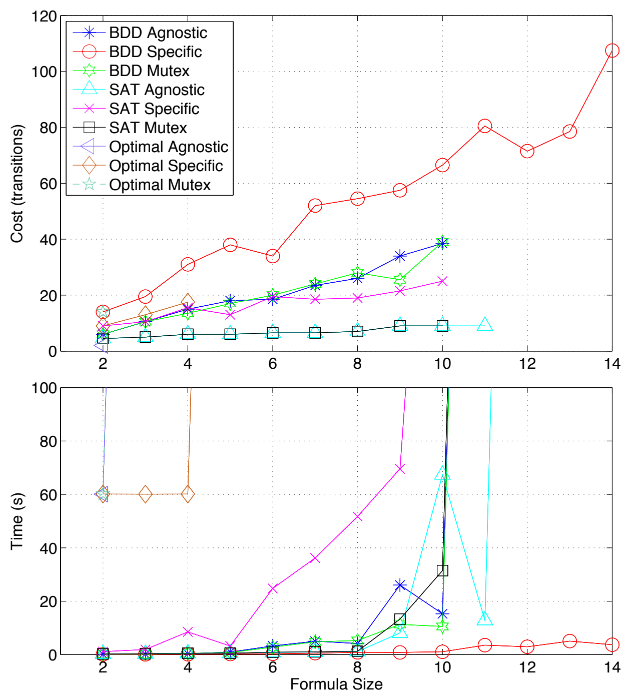

Figure 6: Median costs and computation times for the BDD, SAT, and optimal solvers.

Conclusions and Future Work

Use of formal specifications, synthesis, and verification for UAV missing planning – especially as a basis for human-automation interaction – is in its infancy, leading to a wide range of opportunities for potential research advances. We now discuss some of these as they relate to the background and results described in this paper.

Here, LTL was used as the sole specification language. However, LTL is only one of the dialects in the broader family of temporal logics, some of which may be more suitable for human-automation UAV mission planning. We anticipate the need, for example, to incorporate languages that capture hard timing constraints and probabilistic specifications. Similarly, the models we used to represent UAV behaviors are relatively high-level finite transition systems that abstract away most of the underlying vehicle dynamics, leaving the need to more faithfully capture these dynamics in future work. Another important aspect of UAV mission planning is uncertainty. Algorithmically, problems involving UAVs acting in a priori unknown and dynamic environments can be addressed with reactive synthesis. A related problem is distinguishing between adversarial, environental, and cooperative uncertainties, which is in line with the scenarios discussed in the previous sections. So is compositional synthesis of distributed control protocols, which could address issues related to computational complexity and implementation.

Note since UAV operators will not be familiar with LTL, another open question is how to best allow operators to write mission specifications. Past research has focused on identifying common specification patterns, which can be parameterized and given intuitive explanations (Dwyer, Avrunin, and Corbett 1999). More current research has focused on providing more intuitive "natural language" interfaces, e.g. for instantiating these types of patterns (Konrad and Cheng 2005; Finucane, Jing, and Kress-Gazit 2010) or using English words in place of mathematical operators and increasing flexiblity in the specification grammar (Rothwell et al. 2013). To our knowledge, no comprehensive human studies have been done on the usability of temporal logics for mission planning; however, we plan to perform preliminary studies in the UAV mission planning domain soon.

Last but not least, interesting problems arise in the integration of these capabilities in such a way to facilitate interaction between human operators and automation algorithms. Toward this end, examples of improvements to our current work would include the following: (i) creating a diverse set of feasible solutions – rather than a single solution – to the operator, (ii) formalizing a notion of feedback from the operator and using it to revise the mission specifications or the underlying cost function used for optimal synthesis, and (iii) displaying solutions to the operator in a form that is human-interpretable and human-actionable. Also, when optimal synthesis is too time consuming, synthesis of feasible solutions, e.g. using SAT solvers as in the previous section, may be an appropriate substitute.

Acknowledgments

Portions of this work were funded by AFOSR lab task #13RQ03COR and the AFRL SFFP program. Cleared for public release on 9 January 2014, case #88ABW-2014-0042.

References

Baier, C., and Katoen, J. 2008. *Principles of Model Checking*. The MIT Press.

Cimatti, A.; Clarke, E.; Giunchiglia, E.; Giunchiglia, F.; Pistore, M.; Roveri, M.; Sebastiani, R.; and Tacchella, A. 2002. NuSMV 2: An opensource tool for symbolic model checking. In *Computer Aided Vision*, 241–268.

Dwyer, M.; Avrunin, G.; and Corbett, J. 1999. Patterns in property specifications for finite-state verification. In *Proc. 21st Int. Conf. on Software Eng. (ICSE'99)*, 411–420.

Fainekos, G. E.; Kress-Gazit, H.; and Pappas, G. J. 2005. Hybrid controllers for path planning: A temporal logic approach. In *Proc. 4th IEEE Conf. on Decision and Control*, 4885–4890.

Fainekos, G. E. 2011. Revising temporal logic specifications for motion planning. In *Proc. 2011 IEEE Int. Conf. on Robotics and Automation (ICRA'11)*.

Finucane, C.; Jing, G.; and Kress-Gazit, H. 2010. LTLMoP: Experimenting with language, temporal logic and robot control. In *IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS'10)*, 1988–1993.

Humphrey, L. 2012. Model checking UAV mission plans. In *Proc. AIAA Modeling and Simulation Technologies Conf.*

Karaman, S., and Frazzoli, E. 2008. Vehicle routing with linear temporal logic specifications: Applications to multi-UAV mission planning. In *Proc. AIAA Conf. on Guidance, Navigation, and Control*.

Konrad, S., and Cheng, B. 2005. Facilitating the construction of specification pattern-based properties. In *Proc. IEEE Int. Requirements Eng. Conf. (RE'05)*.

Kress-Gazit, H.; Fainekos, G. E.; and Pappas, G. 2009. Temporal-logic-based reactive mission and motion planning. *IEEE Trans. on Robotics* 25(6):1370–1381.

Lahijanian, M.; Andersson, S. B.; and Belta, C. 2012. Temporal logic motion planning and control with probabilistic satisfaction guarantees. *IEEE Trans. on Robotics* 28(2):396–409.

Liu, J.; Ozay, N.; Topcu, U.; and Murray, R. 2011. Synthesis of switching protocols from temporal logic specifications. Technical report, California Institute of Technology.

Rothwell, C.; Eggert, A.; Patzek, M.; Bearden, G.; Calhoun, G.; and Humphrey, L. 2013. Human-computer interface concepts for verifiable mission specification, planning, and management. In *Proc. AIAA Infotech@Aerospace Conf.*

Smith, S. L.; Tumova, J.; Belta, C.; and Rus, D. 2011. Optimal path planning for surveillance with temporal-logic constraints. *The Int. J. of Robotics Research* 30(14):1695–1708.

Tabuada, P., and Pappas, G. J. 2006. Linear time logic control of discrete-time linear systems. *IEEE Trans. on Automatic Control* 51(12):1862–1877.

Topcu, U.; Ozay, N.; Liu, J.; and Murray, R. 2012. On synthesizing robust discrete controllers under modeling uncertainty. In *Proc. 15th Int. Conf. on Hybrid Systems: Computation and Control (HSCC'12)*.

Tumova, J.; Hall, G.; Karaman, S.; Frazzoli, E.; and Rus, D. 2013. Least-violating control strategy synthesis with safety rules. In *Proc. 16th Int. Conf. on Hybrid Systems: Computation and Control*. ACM.

Wolff, E., and Murray, R. 2013. Optimal control of nonlinear systems with temporal logic specifications. In *Proc. of the 16th Int. Symp. on Robotics Research (ISRR'13)*.

Wolff, E.; Topcu, U.; and Murray, R. 2012. Robust control of uncertain Markov decision processes with temporal logic specifications. In *Proc. 51st IEEE Annu. Conf. on Decision and Control*, 3372–3379.

Wongpiromsarn, T.; Topcu, U.; and Murray, R. 2010. Receding horizon temporal logic planning. *IEEE Trans. on Automatic Control* 57(11):2817–2830.