The Workshops of the Thirtieth AAAI Conference on Artificial Intelligence
Planning for Hybrid Systems: Technical Report WS-16-12

An Architecture for Hybrid Planning and Execution

Robert P. Goldman, Dan Bryce, Michael J. S. Pelican, David J. Musliner
SIFT, LLC
Minneapolis, MN {rpgoldman,dbryce,mpelican,musliner}@sift.net
Kyungmin Bae
Carnegie-Mellon University

Abstract

This paper describes Hy-CIRCA, an architecture for verified, correct-by-construction planning and execution for hybrid systems, including non-linear continuous dynamics. Hy-CIRCA addresses the high computational complexity of such systems by first planning at an abstract level, and then progressively refining the original plan. Hy-CIRCA is an extension of our Playbook[1] approach, which aims to make it easy for users to exert supervisory control over multiple autonomous systems by "calling a play." The Playbook approach is implemented by combining (1) a human-machine interface for commanding and monitoring the autonomous systems; (2) a hierarchical planner for translating commands into executable plans; and (3) a smart executive to manage plan execution by coordinating the control systems of the individual autonomous agents, tracking plan execution, and triggering replanning when necessary. Hy-CIRCA integrates the dReal non-linear SMT solver, with enhanced versions of the SHOP2 HTN planner and the CIRCA Control Synthesis Module (CSM). Hy-CIRCA's planning process has 5 steps: (1) Using SHOP2, compute an approximate mission plan. While computing this plan, compute a hybrid automaton model of the plan, featuring more expressive continuous dynamics. (2) Using dReal, solve this hybrid model, establishing the correctness of the plan, and computing values for its continuous parameters. To execute the plan, (3) extract from the plan specifications for closed-loop, hard real-time supervisory controllers for the agents that must execute the plan. (4) Based upon these specifications, use the CIRCA CSM to plan the controllers. To ensure correct execution, (5) verify the CSM-generated controllers with dReal.

Introduction

In this paper we describe Hy-CIRCA, an architecture for verified, correct-by-construction planning and execution for hybrid systems, including non-linear continuous dynamics (see Figure 2). Hy-CIRCA addresses the high computational complexity of such systems by first planning at an abstract level, and then progressively refining the original plan.

[1]Playbook® is a registered trademark of SIFT, LLC.

Hy-CIRCA is an extension of our Playbook approach for controlling multiple autonomous agents to cover hybrid discrete/continuous planning and control, with non-linear continuous dynamics. The Playbook approach aims to make it easy for users to exert supervisory control over multiple autonomous systems by "calling a play." (Miller et al. 2004) The Playbook approach is implemented by combining (1) a human-machine interface for commanding and monitoring the autonomous systems; (2) a hierarchical planner for translating commands into executable plans; and (3) a smart executive to manage plan execution by coordinating the control systems of the individual autonomous agents, tracking plan execution, and triggering replanning when necessary (see Figure 1).

Our architecture integrates the dReal non-linear SMT solver(Gao, Kong, and Clarke 2013) with enhanced versions of the SHOP2 (Nau et al. 2003) planner and the CIRCA Control Synthesis Module (CSM) (Musliner, Durfee, and Shin 1993; Goldman, Musliner, and Pelican 2002). The planning process in Hy-CIRCA proceeds in 5 steps: (1) Using SHOP2, we compute an approximate mission plan. While computing this plan, we also compute a hybrid automaton model of the plan, featuring more expressive continuous dynamics. (2) Using dReal, we solve this hybrid model, establishing the correctness of the plan, and refining it by computing values for its continuous parameters. This mission plan uses projection to handle resources, and find paths to the goal state. However, it is still an open-loop plan. In order to execute this plan, we (3) extract from the plan specifications for closed-loop, hard real-time supervisory controllers for the agents that must execute the plan. (4) Based upon these specifications, we use the CIRCA CSM to plan the controllers. The CSM uses an abstract model of the continuous dynamics (similar to the temporal abstractions in PDDL 2.1 (Fox and Long 2003)), so to ensure correct execution, (5) we verify the CSM-generated controllers, using a higher-fidelity non-linear hybrid model, with dReal.

We have constructed a proof-of-concept implementation of key parts of Hy-CIRCA, and tested it on a demonstration problem involving multi-agent firefighting using uninhabited aerial vehicles (UAVs). Our work on Hy-CIRCA is ongoing. In this paper, we first describe this aerial firefighting scenario and how Hy-CIRCA meets its challenges. Then we present the three component technologies, describe how

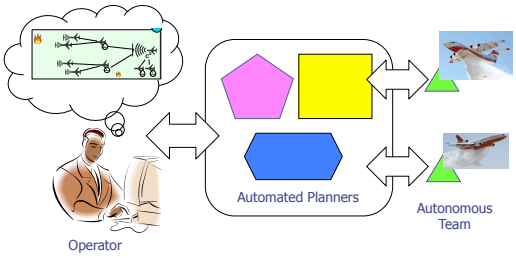

Figure 1: The Playbook architecture for supervisory control.

we have integrated them. Finally, we review related work, and present our conclusions and plans for future work.

Our contributions include: a method of using the SHOP2 planner to perform hybrid automaton model construction; a logical formalization of the SHOP2-generated plan, as a hybrid system, for use in the dReal SMT solver; techniques for extracting controller specifications from an HTN plan; and techniques for verifying the CIRCA closed-loop controllers, as supervisory controllers for hybrid systems.

Scenario of Use

The following motivating use case illustrates how the Hy-CIRCA architecture can be used to provide hybrid planning and control of multiple-platform packages of autonomous systems. In this multi-UAV firefighting scenario (see Figure 3) there is a fleet of UAVs made up of *waterbombers* that drop water or retardant on a fire, *spotters* that localize the fires and transmit targeting coordinates to the waterbombers, *C2 aircraft* that coordinate operations and have long-range sensors that detect likely fires, and *tankers* that provide in-air refueling.

The mission's objective is to extinguish a fire which has just been spotted by the long-range low-accuracy sensors on a C2 aircraft. An airborne fire manager on the C2 aircraft uses his Playbook interface to call the *extinguish* play, tailoring the play to the situation by incorporating information about the location of the suspected fire, the assets available to the team, and their current state/location. The mission plan (see Figure 4) first requires that a spotter go to the vicinity of the suspected fire to confirm its existence. After the fire's existence has been confirmed, it can be extinguished by dropping a load of retardant or water from a waterbomber. But in order for the waterbomber to correctly target the fire, it must get a stream of location information from a spotter. After the waterbomber has attempted to extinguish the fire, it returns home, and the spotter must confirm that the fire has been extinguished.

Initially, Hy-CIRCA invokes the HTN planner SHOP2 to generate a mission plan. SHOP2 generates a temporal mission plan in which the spotter is vectored to the fire area to confirm the fire's existence. SHOP2 also plans a route for the waterbomber to reach the area of the fire to extinguish the fire. SHOP2 chooses a take-off time for the waterbomber that attempts to get it to the fire area shortly after the spotter has arrived. This will avoid fuel waste that would occur in a plan where the waterbomber arrives first and must loiter, waiting for the spotter. But we don't want the waterbomber to arrive too late: even though the spotter uses less fuel than the waterbomber, we still don't want to waste its fuel, or keep it from other uses. Note that this means that our problem has *required concurrency* (Cushing et al. 2007).

SHOP2's plan is only approximate, because of limits in its computations about real continuous quantities such as fuel (in particular, its inability to solve systems of simultaneous equations). But SHOP2 can more efficiently solve the discrete sequencing problems than can dReal, and it can invoke special solvers to synthesize waypoint sequences.

Once the initial plan has been computed, Hy-CIRCA uses the higher fidelity reasoning offered by dReal to refine it. As a side-effect of computing its plan, Hy-CIRCA's version of SHOP2 builds a hybrid systems SMT problem. Hy-CIRCA uses dReal to solve this problem where a solution is a satisfying trajectory through the high-dimensional hybrid space. By solving that problem, dReal will synthesize continuous parameters for the mission plan. In our tests, dReal solved a system of non-linear constraints to choose the fuel and flame retardant loads for the waterbomber, and refined the mission schedule based on better models of flight than those used by SHOP2.

The mission plan guides the operation of the autonomous systems, but is not sufficient for their closed-loop control. Hy-CIRCA uses an enhanced version of the CIRCA Control Synthesis Module (CSM) to automatically synthesize controllers for the platforms (waterbombers, spotters, etc.) in the mission. These controllers will constitute the smart executives for those platforms.

The first step of controller synthesis process is to generate controller specifications from the mission plan. These specifications include both temporal logic invariants and goals. For our current Hy-CIRCA proof of concept, we extracted the specifications for the waterbomber aircraft in the mission. These specifications include control operations used in flight, representations of known disturbances, temporal logic representations of the goals ("eventually the fire should be extinguished"), temporal logic invariants ("the waterbomber must release its load within k time units from receiving a target message from the spotter"), and signal temporal logic (STL) (Maler and Nickovic 2004) hybrid invariants ("the vehicle must always maintain a fuel reserve of at least n gallons").

The CIRCA CSM uses these inputs to synthesize a closed-loop, real-time discrete outer-loop controller for each platform. Note that these controllers will perform coordinated actions, directed by the mission plan that has been translated into temporal logic specifications. For example, the spotter will repeatedly transmit targeting information until the waterbomber has dropped its load. Similarly, the waterbomber will wait in the vicinity of the fire until it has received the targeting information, and is constrained to drop its load before the targeting information becomes stale.

The CIRCA CSM reasons about continuous processes only as approximated by temporal durations. For example, its reasoning about whether it can reach the target quickly enough is based on whether it can initiate the motion soon enough, and is verified in terms of time bounds on its flight

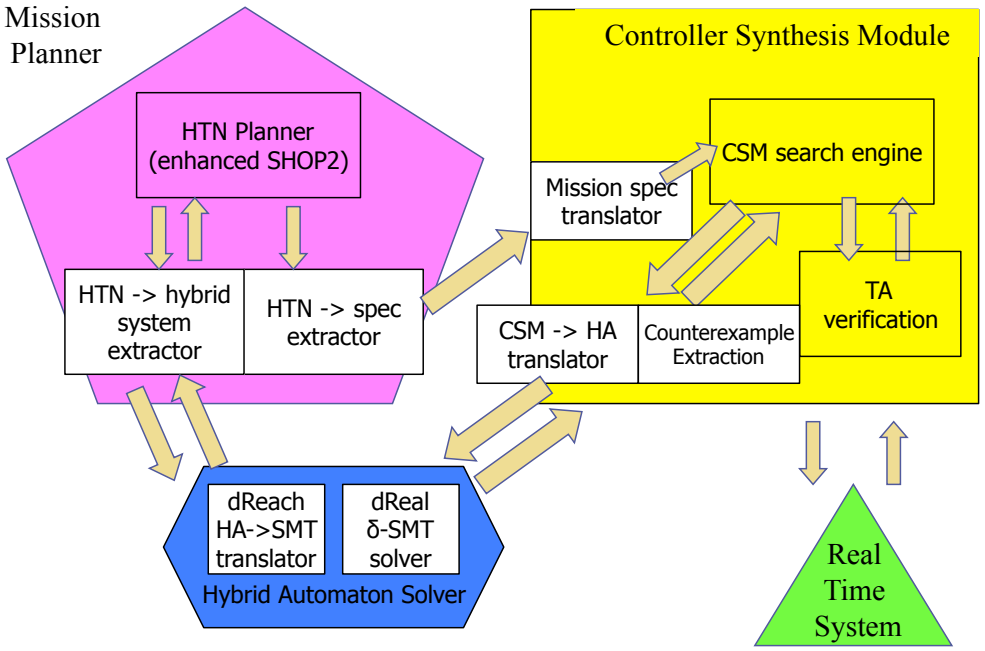

Figure 2: Hy-CIRCA will synthesize hybrid controllers to satisfy mission specifications expressed in high-level temporal logic through two-way interactions between the Mission Planner, the Control Synthesis Module, and the Hybrid Automaton Solver.

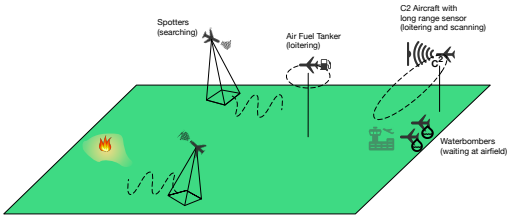

Figure 3: Multiple-UAV firefighting scenario.

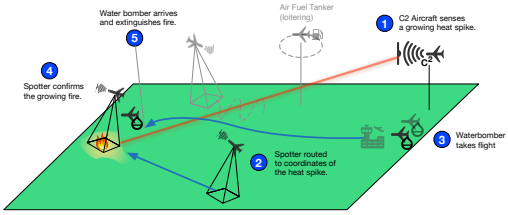

Figure 4: Plan to extinguish a fire.

processes.

For more accurate reasoning about the control of continuous processes, Hy-CIRCA re-checks the controller using dReal. We developed a technique for translating the CIRCA controllers and STL invariants into a hybrid automaton representation that can be checked by dReach and dReal. We tested this algorithm by translating the waterbomber controller into a hybrid automaton, the STL invariants into a separate automaton, forming the product, and then checking reachability. The reachability computation was done using dReach's translation from automaton to dReal SMT formulas. For example, in one of our tests, we checked the invariant that the waterbomber would always have an adequate fuel reserve.

If Hy-CIRCA finds that an invariant cannot be verified, it will use a process of culprit extraction to translate the counterexample into information that the CIRCA CSM can use to guide backtracking and repair the synthesized controller. We have developed a method for performing this culprit extraction. We tested the approach on an example where the waterbomber's controller originally generated a plan that involved monitoring a fuel level warning, which is signaled when a threshold fuel level is reached (like the fuel level warning in a car). The original synthesized controller would go into a special fuel conserving flight mode and return to base if the low fuel warning was triggered. In some missions, where the flight radius is lower, this controller will be verified to work correctly.

However, if the flight radius is longer, then dReal will detect that a failing state can be reached if the fuel warning

goes off when the aircraft is beyond some distance d from base. From this counterexample, Hy-CIRCA can extract a culprit that indicates that the controller using the special return-to-base flight mode is not safe. This will cause the CSM to backtrack and choose the (more expensive) recovery action of in-air refueling. The resulting revised controller then will be verified by dReal.

In the following section, we describe the existing three components that we integrated into Hy-CIRCA. After that we explain the techniques we have developed to enable them to work together.

Background

The three components on which we have built Hy-CIRCA include the SHOP2 HTN planner, the CIRCA intelligent real-time control framework and the dReach/dReal tools for hybrid verification. In this section, we briefly review each of these systems.

SHOP2

Hy-CIRCA's top-level Mission Planner (MP) interacts with the operator and the lower-level planners to direct the transformation of an incompletely specified play into a set of executable controllers. We use the SHOP2 HTN planner (Nau et al. 2005; 2003) as the foundation of the Hy-CIRCA MP. SHOP2 is a modern HTN planner with a relatively clean implementation that performed well in the IPC knowledge-based planning track. SHOP2 is available under a generous open-source license, and is maintained at SourceForge (by SIFT). Like other HTN planners, and unlike first-principles planners, SHOP2 searches top-down from a task or set of tasks, rather than chaining together primitive actions. SHOP2 and other HTN planners decompose complex tasks into more primitive sub-tasks (methods), thus building a plan tree that terminates at leaves corresponding to primitive actions (operators).However, unlike other HTN planners (Erol, Hendler, and Nau 1994, *e.g.*), SHOP2 operates forward in time, building its plan from left to right and always having a well-formed sequential plan prefix. This avoids the complications of partial-order planning and permits the use of complex operator models, which are not limited to STRIPS or PDDL expressivity. For our purposes in Hy-CIRCA, this method of operation has the advantage of providing an easy way to capture standard operating procedures (SOPs) and trajectory constraints, and interaction with external solvers (*e.g.*, a geometric route planner), that would not be possible if limited to PDDL expressive power.

The Hy-CIRCA MP produces a plan with a coordinated set of tasks for the autonomous vehicles. Constraints on these tasks include temporal deadlines, resource limits, and synchronization requirements. For example, in an aerial firefighting domain the system needs different controllers for mission phases such as takeoff, ingress, coordinated monitor and extinguish, and landing. Those controllers would be responsible for accomplishing different types of goals and handling different types of failures in each phase. See the example in Figure 5.

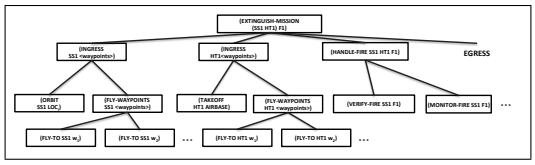

Figure 5: Example SHOP2 plan for the firefighting scenario. The plan shows tasks for the spotter aircraft (SS1), and the waterbomber (HT1). Note the incorporation of waypoint-flying operators ("fly-to").

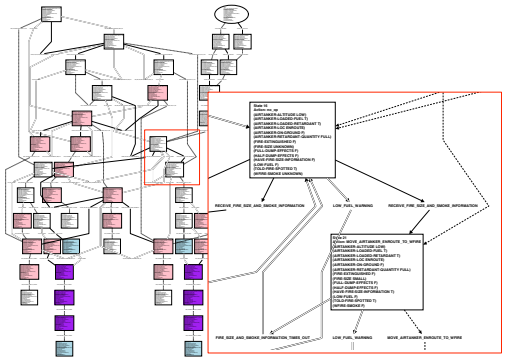

Figure 6: This CSM-generated controller for a fire fighting water bomber handles communications from a spotter aircraft and possible low fuel warnings.

CIRCA

In order to operate the vehicles in the plan, the Hy-CIRCA CSM generates a *hard real-time, closed-loop controller* for each platform for each mission phase. These mission phases correspond to one or more subtasks of the overall mission plan. These closed-loop controllers handle operations of lower-level control behaviors of the platforms, coordinate different behaviors, coordinate behaviors across platforms and handle disturbances. See Figure 6 for an example controller.

The Hy-CIRCA CSM receives problem descriptions from the MP that As with most planners, a CIRCA problem description specifies an initial state of the world, a set of goal states that the planner attempts to reach, a distinguished failure state that the planner must avoid, and a set of actions that move the world from one state to another. Unlike most planning systems, CIRCA's actions include uncontrollable actions, used to model adversaries, environmental disturbances, system failures, etc. Also unlike most planning systems, CIRCA's actions timing characteristics that specify how fast the transitions can occur.

The CSM attempts to create a goal-achieving *safe* plan in which no failure states are reachable. To do so, the CSM searches for an assignment of controllable actions to each anticipated reachable state, such that the resulting state space is closed and drives the system towards its goals while avoiding failure states. If the CSM encounters a transition to a failure state, it chooses an action that can "preempt" that transition by violating its preconditions, and which can be

guaranteed to occur before the failure transition. For example, consider a case where the CSM is synthesizing a controller for a UAV that may encounter a heat-seeking surface-to-air missile (SAM), and the UAV is constrained to fly at an altitude that ensures it will take at least k seconds for the SAM to reach the UAV. The CSM's problem specification may also show that the UAV can drop flares, and that these flares will break a heat-seeking SAM lock in l seconds. The CSM will attempt to synthesize a controller that will guarantee to drop flares at most $k - l$ seconds after SAM launch.[2] If the CSM is unable synthesize a controller that will guarantee to preempt the SAM attack, it would use verifier counterexamples to backtrack in the synthesis process (Goldman, Pelican, and Musliner 2004), and attempt to avoid states in which the SAM threat was active (*e.g.*, by flying at a higher altitude).

The CSM uses timed automaton (Alur 1998) verification techniques to check its plans and ensure that failure is not reachable. CIRCA uses a PDDL-style domain description language to provide a compact, implicit representation for its controller synthesis problems. It exploits this compact representation, using "on-the-fly" verification techniques, to control the state space explosion (Goldman, Musliner, and Pelican 2002). A second key to the CSM's efficiency is its **counterexample-guided backjumping**: it uses counterexamples generated by the verifier to guide backtracking, eliminating from consideration substantial portions of the synthesis search space that are provably barren (Goldman, Pelican, and Musliner 2004). Finally, CIRCA uses a plan-graph heuristic to guide planning. Unlike conventional planners, CIRCA's heuristic must balance goal achievement against safety preservation.

dReal and dReach

Hy-CIRCA reasons about continuous resources over time by applying the dReach model checker and the associated dReal SMT solver. dReach formulates model checking problems as SMT, and dReal determines whether they are unsatisfiable or δ-satisfiable.

dReal (Gao, Avigad, and Clarke 2012; Gao et al. 2010) is based upon the rigorous foundation of *computable analysis* (Brattka, Hertling, and Weihrauch 2008), the study of computability and complexity questions of continuous functions over the real numbers. The approach investigates the notion of numerical *perturbations* on logic formulas. The traditional *decision problem* asks whether a logic formula is "true" or "false." Instead, we ask the question of whether a formula is "true", or "its perturbed form is false." This minor change permits imprecision (in the form of a real parameter $\delta > 0$) in the decision algorithms for logic formulas, while controlling this imprecision to be one-sided and bounded. By doing this, we replace the conventional decision problem with the *δ-decision problem*, and study *δ-complete* algorithms. δ-completeness is extensively investigated in (Gao, Avigad, and Clarke 2012) and the key results

[2]For simplicity, we assume here that the UAV is able to detect SAM launch instantly. In a real application, CIRCA would *not* make this assumption.

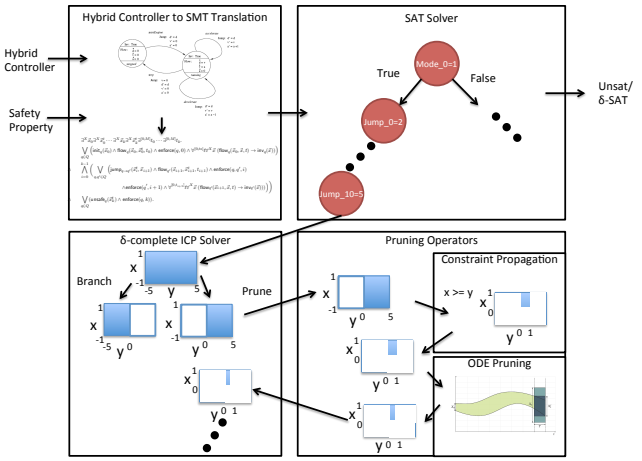

Figure 7: dReach translates a hybrid automaton model to a logical formula, which can be solved dReal's SMT methods. dReal combines SAT and interval constraint propagation into a δ-complete solver.

state that, while the logic theories over the reals with any interesting functions are either highly intractable or undecidable, the δ-decision versions of the problems avoid undecidability and have a much lower complexity.

In bounded model checking, dReach can translate the safety property of a system to a logical formula, P, such that if dReal determines P is false, then the system is safe. Now, note that in δ-complete decision procedures, errors can only occur when the procedures decides a formula is *true*. Thus, when we conclude a system is safe, we are absolutely correct and no numerical errors can affect the answer. On the other hand, when we say a system is unsafe, due to the possible errors in the procedures, we can conclude that either the system is indeed unsafe, or that under some perturbations the system would *become* safe. In the latter case, since the perturbations are controlled by the parameter δ which can be arbitrarily small, a minor change in the environment would render it unsafe, and consequently the system should not pass the verification stage anyway, since it is not robust. In this way, we can now take full advantage of scalable numerical methods in our reasoning tools.

Integrated Solution

Our Hy-CIRCA architecture integrates the Playbook tasking model, the SHOP2 HTN planner, the closed-loop CIRCA Control Synthesis Module, and the dReal SMT solver for hybrid systems. In this paper we focus on the operations internal to the system, rather than the Playbook human-machine interface. The planning and controller synthesis process is summarized in Figure 2: initially tasks are decomposed by SHOP2 to build a mission plan (top left). While building the mission plan, SHOP2 also composes a hybrid automaton model for the plan. That hybrid automaton model is solved by dReal (lower left), to synthesize continuous parameters for the plan. Once the mission plan is

complete, with continuous parameters, it is translated into a set of specifications for the CIRCA CSM (upper right). The CSM builds a closed-loop plan/controller for each phase of the mission, which is guaranteed to be correct *with respect to a temporal abstraction*. After the controller is built, it is checked against a higher-fidelity hybrid automaton model, which includes continuous dynamics based on a hybrid automaton translation of the controller. We discuss each of these processes below.

Integrating SHOP2 and dReal

In Hy-CIRCA, SHOP2 provides the top-level mission structure, based on a simplified model of a domain's continuous dynamics. In order to build an accurate mission plan, we use dReal to refine this approximate plan. This two-step process helps us focus the more expensive process of hybrid, nonlinear planning.

Specifically, we construct a dReal hybrid automaton problem corresponding to the HTN plan. Solving this problem both establishes the correctness of the original SHOP2 plan, and identifies continous parameters for the plan, based on a fuller, non-linear model of the continuous dynamics. In our firefighting scenario, we used dReal to compute feasible payload configurations for the water tankers in terms of the proper amount of fuel and water. Figure 10 illustrates the result of the dReal analysis for one of the water tankers. The figure shows the values of several continuous variables throughout the plan, including the fuel and water levels as they are consumed by the flight and firefighting activities.

The SHOP2 plan handles resources and temporal aspects of the problem using the technique of *multiple-timeline preprocessing* (MTP) (Nau et al. 2003). In the MTP approach, each resource has a corresponding timeline, and in addition to tracking the state of the resource for use in preconditions, the planner also records the latest read-time and write-time for this state. This technique allows a limited form of partial-order temporal reasoning, and parallel execution, at a very modest cost in bookkeeping. Limited amounts of more complex temporal reasoning can be handled using axioms. For example, the plan library excerpt in Figure 9 shows that each vehicle has a consumable `fuel` resource tracked according to MTP. Another library excerpt (Figure 8) shows the use of an axiom to perform some special-purpose temporal reasoning: the method chooses a take off time for the water-bomber (`?ht`) based on approximate reasoning about its time-to-target given a set of waypoints given by `?wayin-ht`. SHOP2 will arrange for the heavy tanker to arrive on the target after the spotter has arrived and verified the location of the fire. This coordination involves *required concurrency* (Cushing et al. 2007), so cannot be done by MTP alone.

We build the dReal model concurrently with the construction of the SHOP2 plan. In the dReal model, we abstract away the particular temporal values of the SHOP2 plan, but retain its ordering constraints and add constraints that specify how each activity consumes resources and their resource preconditions. Activities in the SHOP2 plan are represented as temporal intervals with constraints corresponding to the orderings in the SHOP2 plan. Continuous quantities are represented as dReal continous variables, as are the start and end times of the intervals. We formulate this model as an SMT encoding.

```
((:method (team-extinguish-fire)...
  ;; task list
  (:ordered
    (!!assert-facts
     ((dreal :before-or-equal ?earliest-start-ss time)
      (dreal :declare-var (fuel ?ss) ?sensor-fuel)
      (dreal :declare-var (fuel ?ht) ?tanker-fuel)
    ;; coordinated ingress
    (ingress ?ss ?wayin-ss ?earliest-start-ss
             ?ingress-start-ss ?ingress-end-ss
             ?first-ss-opid ?last-ss-opid)
    (!!compute-tanker-takeoff-time ?ht ?wayin-ht
                                   ?ingress-end-ss
                                   ?earliest-start-ht)
    (ingress ?ht ?wayin-ht ?earliest-start-ht
             ?ingress-start-ht ?ingress-end-ht
             ?first-ht-opid ?last-ht-opid)...
```

Figure 8: This fragment of the top-level SHOP2 method for the firefighting domain shows annotations that create dReal variables, assert temporal constraints, and compute approximate deadlines.

```
(:operator (!takeoff ?opid ?p ?flight-alt ?earliest-start
                     ?start ?end)...
  ;; preconditions
  (fuel ?p ?fuel)
  (assign ?fuel-cost (takeoff-fuel-cost ?flight-alt))
  (assign ?fuel-remaining (- ?fuel ?fuel-cost))
  (call >= ?fuel-remaining 0)
  ;; timelines for fuel update
  (write-time (fuel ?p) ?t-write-fuel)
  (read-time (fuel ?p) ?t-read-fuel)  ...
  ;; deletes
  (fuel ?p ?fuel)
  ;; timelines for fuel update
  (write-time (fuel ?p) ?t-write-fuel)
  (read-time (fuel ?p) ?t-read-fuel)
  (read-timepoint (fuel ?p) ?tp-read-fuel) ...
  ;; adds
  (fuel ?p ?fuel-remaining)
  ;; timelines for fuel update
  (write-time (fuel ?p) ?end)
  (read-time (fuel ?p) ?end)
  ;; dreal constraints
  (dreal :before-or-equal ?tp-write-at (begin ?opid))
  (dreal :before-or-equal ?tp-write-fuel (begin ?opid))
```

Figure 9: These elements of the SHOP2 `takeoff` operator for platform `?p` compute approximate `?fuel-remaining` and manage the fuel resource timelines necessary to create constraints for dReal.

The methods and operators in the SHOP2 plan library contain instructions to build the dReal model. For example, in Figure 8, we see the declaration of two fuel variables, and the addition of temporal constraints (using `:before-or-equal`), to capture the temporal structure of the plan.

Using this extended HTN plan library, Hy-CIRCA simultaneously computes a complete, approximate sequence of operators and a complete dReal model, building on the SMT encoding techniques we have previously developed (Bryce et al. 2015). dReal solves the resulting hybrid problem using continuous models, for example to compute fuel usage and vehicle velocity. By solving the hybrid problem, dReal both proves the feasibility of the plan proposed by SHOP2

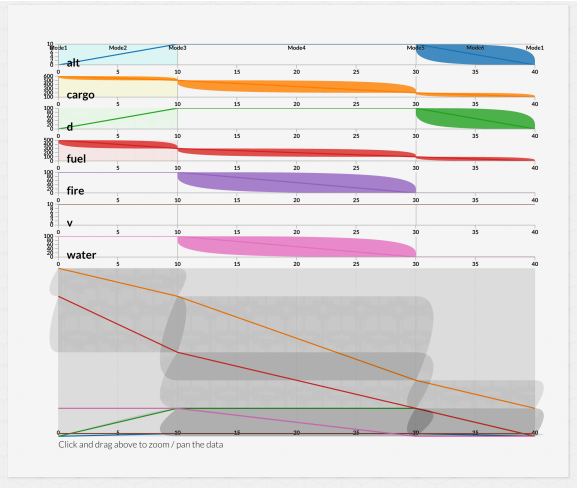

Figure 10: Plots of continuous parameters for firefighting plan generated by dReal.

and finds a consistent set of parameters for a plan solution. Figure 10 shows how dReal finds an envelope or multi-dimensional "pipe" of feasible values for the salient mission variables throughout a sample firefighting mission plan.

Planner/CSM integration

We have prototyped methods for implementing LTL mission specifications in terms of the CIRCA CSM's existing goal and failure representation. The CSM enforces safety conditions by means of *transitions to failure* that cause the CSM to fail (and backtrack) when a safety goal is violated. In order to accommodate more flexible temporal logic specifications, and the translation of SHOP2-generated mission plans into CSM problems, we have developed a translation from temporal logic invariants to existing CIRCA CSM constructs.

Recent work (Humphrey, Wolff, and Topcu 2014; De Giacomo, De Masellis, and Montali 2014) in AI planning and UAV control systems has argued for the use of temporal logic for the specification of mission goals and preferences. For example, temporal logic could be used to express properties such as "always when there is a fire reported at location w_i, a spotter will eventually visit location w_i." To support the real-time guarantees that CIRCA makes, Hy-CIRCA uses Metric Interval Temporal Logic, a temporal logic variant that adds bounds to deadlines and durations. For example, an operator will be able to require "always when there is a fire reported at location w_i, a spotter will *within 10 minutes* visit location w_i." Temporal logic specifications lend themselves to many relevant classes of mission specifications, such as deadlines, durations, and fairness conditions.

For example, in the egress phase of our fire fighting domain, the vehicles have two flight modes: (1) a nominal waypoint-following mode and (2) an emergency flight mode which conserves fuel by taking a direct path the airbase and other conservation measures. The mission director may require that whenever the warning triggers, the controlled vehicle must land within 20 minutes. This constraint can be encoded in classic LTL notation as:

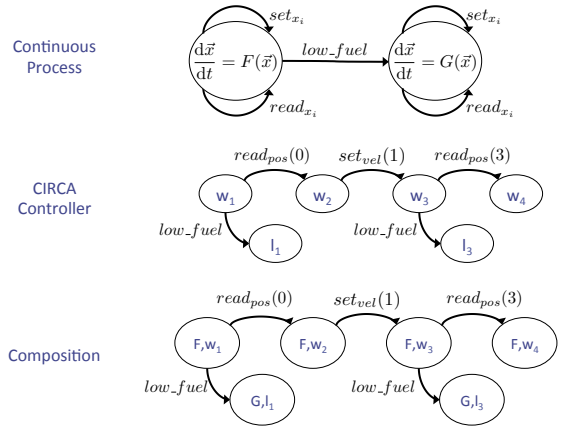

Figure 11: Forming hybrid automaton by means of synchronous composition.

$$\Box(low\text{-}fuel\text{-}warning \implies \Diamond_{\leq 1200}(loc = w_{landed}))$$

In Hy-CIRCA syntax, we encode the same constraint as:

```
(:always
  (:implies (low-fuel-warning t)
    (:bounded-eventually (landed t)
                         :bound 1200)))
```

Finally, our mission specification translator constructs a potential failure in CIRCA's domain language to enforce the specification:

```
(DEF-TEMPORAL <GENSYMNAME>
   :PRECONDS ((LANDED F)
              (LOW-FUEL-WARNING T))
   :POSTCONDS ((FAILURE T))
   :MIN-DELAY 1200)
```

The presence of this failure transition will cause the CSM to build a plan which avoids the failure condition in the timed discrete model with which it plans. If this planning completes successfully, dReach will test the resulting plan to ensure that the invariant(s) hold in the higher-fidelity nonlinear hybrid model, as we explain in the following section.

Hybrid verification of CSM controllers with dReach

The final step in planning for Hy-CIRCA is to verify the correctness of the CSM-generated controllers, using dReal. Recall that during the generation of these controllers, their correctness is assured by incremental verification. However, this is correctness assessed against a model in which all continuous change is summarized in terms of durative action (a timed automaton model). In this final stage, we check the generated controllers against a higher-fidelity hybrid automaton model with nonlinear continuous dynamics.

In order to do so, we must translate the CSM controller into a hybrid automaton. We have developed a technique

for doing this that relies on the relationship between the CSM models and timed automata. So initially we translate the controller into a timed automaton model, and then we augment the simple temporal model of processes by hybrid models. Hy-CIRCA has ancillary models of the continuous processes in process in each of the states of the timed automaton, and these models are merged, using synchronous composition, to a full-fledged hybrid automaton (see Figure 11).

We also translate the STL invariants that the controller must honor into hybrid automata. Bae has developed a new translation of STL to hybrid automata that we use to perform this operation. The automata for the invariants are composed with the hybrid model of the controller. The resulting product machine is then checked using dReach, producing either confirmation or a counterexample.

The CIRCA CSM uses *verifier-directed backjumping* in its search (Goldman, Pelican, and Musliner 2004). Briefly, states of the timed automaton, and edges between them, that appear in a counterexample, are mapped to decisions made in the search. The decisions in question are either action choices, or temporal decisions (must one transition complete before another). Using this mapping, we may translate counterexamples to nogood clauses, and use these clauses to direct backjumping. Backjumping is essential to controlling the search in the CSM. We have designed a nogood extraction approach for the hybrid verification step, that parallels that used in the timed automaton verification step. We have tested it in examples from the firefighting domain, although it is not yet fully implemented and integrated.

Related Work

Prior work (Cimatti et al. 2003) has addressed closed-loop planning, where the objective is to find weak, strong, or strong-cyclic plans that achieve the goal despite non-determinism. CIRCA addresses a special form of non-determinism, namely temporal non-determinism that is both endogenous and exogenous. Our extension to CIRCA to reason about continuous resources is most similar to work on continuous state MDPs (Guestrin, Hauskrecht, and Kveton 2004), albeit without the decision theoretic aspects of the model.

Humphrey *et al.* (2014) have developed an approach that uses a model-checker to synthesize multi-vehicle plans for high-level mission specifications expressed in LTL. They have shown that many specification patterns common to UAV missions, such as recurrent surveillance coverage and no-fly zone avoidance can be represented in standard LTL. Using discretized representations of the continuous properties of the domain, such as vehicle dynamics, their algorithm finds mission plans in dozens of seconds. Our approach complements this work by using a top-down HTN planning approach to narrow the search space by composing mission high-level tasks and mission specifications and, simultaneously, extending the search space by incorporating the continuous aspects into the mission planning problem. In the context of Hy-CIRCA, the increased complexity of tackling the hybrid problem is balanced by the focus provided by the top-down guidance of the HTN planner.

Our work is also related to the recent surge of interest in PDDL+ planning (Fox and Long 2006) – temporal planning with continuous change and exogenous events and processes. Similar to CIRCA, many of these works are based upon either model checking hybrid automata (Bryce et al. 2015; Bogomolov et al. 2014; Della Penna et al. 2009) or heuristic search (Coles and Coles 2014; Coles et al. 2012). Our use of dReal to reason about hybrid planning as SMT is also related to TM-LPSAT (Shin and Davis 2005), which combines SAT and LPs.

Conclusions and Future Work

We have described our Hy-CIRCA framework for integrated planning, controller synthesis, and verification in hybrid domains. Hy-CIRCA decomposes complex mission planning into strategy planning with SHOP2, controller synthesis with CIRCA, and verification with dReach. Some of the most challenging and novel aspects of this framework are how the tools integrate to solve the overall problem. For instance, SHOP2 and dReal interact to solve planning and scheduling with numeric resources. SHOP2 and CIRCA jointly synthesize low-level controllers for high-level actions that must satisfy temporal properties. CIRCA and dReach cooperate to verify the controllers with respect to the underlying nonlinear continuous change.

As we continue to develop the Hy-CIRCA framework, the important remaining issues concern how to automatically divide the overall problem among the Hy-CIRCA components and how to effectively evaluate these decisions. We recognize that the division of decision making between the tools is an important area for future research, especially given that we have recently made advances in scaling dReal's discrete decision making capabilities (Bryce et al. 2015). We have designed algorithms for SHOP2/CSM and CSM/dReach integration, and tested on cases drawn from the firefighting scenario. We are about to begin a second phase of the project to complete the integration and extend our set of experiments. We are also working to extend the capabilities of the system in a number of areas. In order to handle the full generality of the temporal specifications handled by the Hy-CIRCA CSM, we are improving the native finite trace verification of temporal logic specifications in the CSM. The current timed automaton verifier in the CSM checks only safety conditions. We will extend it to handle liveness conditions, as well. While the basics of liveness checking are well understood (Clarke, Grumberg, and Peled 1999), our algorithms for extracting nogoods – used for backjumping – need extension in order to accommodate counterexamples to liveness goals.

Acknowledgments Thanks to the anonymous reviewers, and to Laura Humphrey, Sicun Gao, and Soonho Kong for helpful comments and advice. This material is based upon work supported by the U.S. Air Force (AFRL) under Contract No. FA9550-15-C-0030.. Any opinions, findings and conclusions, or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the U.S. Air Force (AFRL)..

References

Alur, R. 1998. Timed automata. In *NATO-ASI Summer School on Verification of Digital and Hybrid Systems*.

Bogomolov, S.; Magazzeni, D.; Podelski, A.; and Wehrle, M. 2014. Planning as model checking in hybrid domains. In Brodley, C. E., and Stone, P., eds., *Proceedings of the Twenty-Eighth AAAI Conference on Artificial Intelligence, July 27 -31, 2014, Québec City, Québec, Canada.*, 2228–2234. AAAI Press.

Brattka, V.; Hertling, P.; and Weihrauch, K. 2008. A tutorial on computable analysis. In Cooper, S. B.; Löwe, B.; and Sorbi, A., eds., *New Computational Paradigms*, 425–491. Springer New York.

Bryce, D.; Gao, S.; Musliner, D. J.; and Goldman, R. P. 2015. Smt-based nonlinear PDDL+ planning. In Bonet, B., and Koenig, S., eds., *Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence, January 25-30, 2015, Austin, Texas, USA.*, 3247–3253. AAAI Press.

Cimatti, A.; Pistore, M.; Roveri, M.; and Traverso, P. 2003. Weak, strong, and strong cyclic planning via symbolic model checking. *Artificial Intelligence* 147(12):35 – 84. Planning with Uncertainty and Incomplete Information.

Clarke, E. M.; Grumberg, O.; and Peled, D. A. 1999. *Model Checking*. Cambridge, Massachusetts: The MIT Press.

Coles, A. J., and Coles, A. I. 2014. PDDL+ planning with events and linear processes. In Chien, S.; Do, M. B.; Fern, A.; and Ruml, W., eds., *Proceedings of the Twenty-Fourth International Conference on Automated Planning and Scheduling, ICAPS 2014, Portsmouth, New Hampshire, USA, June 21-26, 2014*. AAAI.

Coles, A.; Coles, A.; Fox, M.; and Long, D. 2012. Colin: Planning with continuous linear numeric change. *Journal of Artificial Intelligence Research* 44:1–96.

Cushing, W.; Kambhampati, S.; Mausam; and Weld, D. S. 2007. When is temporal planning really temporal? In Veloso, M. M., ed., *IJCAI 2007, Proceedings of the 20th International Joint Conference on Artificial Intelligence, Hyderabad, India, January 6-12, 2007*, 1852–1859.

De Giacomo, G.; De Masellis, R.; and Montali, M. 2014. Reasoning on ltl on finite traces: Insensitivity to infiniteness. In *Proceedings of the Twenty-Eighth AAAI Conference on Artificial Intelligence*.

Della Penna, G.; Magazzeni, D.; Mercorio, F.; and Intrigila, B. 2009. Upmurphi: A tool for universal planning on pddl+ problems. In *ICAPS*.

Erol, K.; Hendler, J.; and Nau, D. S. 1994. UMCP: A sound and complete procedure for hierarchical task network planning. In Hammond, K. J., ed., *Artificial Intelligence Planning Systems:Proceedings of the Second International Conference*, 249–254. Waltham, MA: Morgan Kaufmann.

Fox, M., and Long, D. 2003. PDDL2.1: An extension to PDDL for expressing temporal planning domains. *Journal of Artificial Intelligence Research* 20:61–124.

Fox, M., and Long, D. 2006. Modelling mixed discrete-continuous domains for planning. *J. Artif. Intell. Res.(JAIR)* 27:235–297.

Gao, S.; Avigad, J.; and Clarke, E. M. 2012. δ-complete decision procedures for satisfiability over the reals. In *IJCAR*.

Gao, S.; Ganai, M. K.; Ivancic, F.; Gupta, A.; Sankaranarayanan, S.; and Clarke, E. M. 2010. Integrating icp and lra solvers for deciding nonlinear real arithmetic problems. In *FMCAD*, 81–89.

Gao, S.; Kong, S.; and Clarke, E. M. 2013. dReal: An SMT solver for nonl. theories over the reals. In *CADE*.

Goldman, R. P.; Musliner, D. J.; and Pelican, M. J. 2002. Exploiting implicit representations in timed automaton verification for controller synthesis. In *Proceedings of the 2002 Hybrid Systems: Computation and Control Workshop*.

Goldman, R. P.; Pelican, M. J. S.; and Musliner, D. J. 2004. Guiding planner backjumping using verifier traces. In *International Conference on Automated Planning and Scheduling*.

Guestrin, C.; Hauskrecht, M.; and Kveton, B. 2004. Solving factored mdps with continuous and discrete variables. In *Proceedings of the 20th Conference on Uncertainty in Artificial Intelligence*, UAI '04, 235–242. Arlington, Virginia, United States: AUAI Press.

Humphrey, L. R.; Wolff, E. M.; and Topcu, U. 2014. Formal specification and synthesis of mission plans for unmanned aerial vehicles. In *Proceedings of the AAAI Spring Symposium: Formal Verification and Modeling in Human-Machine Systems*, 116–121.

Maler, O., and Nickovic, D. 2004. Monitoring temporal properties of continuous signals. In *Formal Techniques, Modelling and Analysis of Timed and Fault-Tolerant Systems*. Springer. 152–166.

Miller, C. A.; Goldman, R. P.; Funk, H. B.; Wu, P.; and Pate, B. 2004. A playbook approach to variable autonomy control: Application for control of multiple, heterogeneous unmanned air vehicles. In *American Helicopter Society 60th Annual Forum Proceedings*, 2146–2157. Alexandria, VA: American Helicopter Society.

Musliner, D. J.; Durfee, E. H.; and Shin, K. G. 1993. CIRCA: a cooperative intelligent real-time control architecture. *IEEE Transactions on Systems, Man and Cybernetics* 23(6):1561–1574.

Nau, D.; Au, T. C.; Ilghami, O.; Kuter, U.; Murdock, J. W.; Wu, D.; and Yaman, F. 2003. SHOP2: An HTN planning system. *Journal of Artificial Intelligence Research* 20:379–404.

Nau, D. S.; Au, T.-C.; Ilghami, O.; Kuter, U.; Munoz-Avila, H.; Murdock, J. W.; Wu, D.; and Yaman, F. 2005. Applications of SHOP and SHOP2. *IEEE Intelligent Systems* 20(2):34–41.

Shin, J., and Davis, E. 2005. Processes and continuous change in a sat-based planner. *Artif. Intell.* 166(1-2):194–253.