Safe CPS from Unsafe Controllers

Usama Mehmood
Stony Brook University
Stony Brook, USA

Stanley Bak
Stony Brook University
Stony Brook, USA

Scott A. Smolka
Stony Brook University
Stony Brook, USA

Scott D. Stoller
Stony Brook University
Stony Brook, USA

ABSTRACT

Modern cyber-physical systems (CPS) interact with the physical world, hence their correctness is important. In this work, we build upon the Simplex Architecture, where control authority may switch from an unverified and potentially unsafe *advanced controller* to a verified-safe *baseline controller* in order to maintain system safety. We take the approach further by lifting the requirement that the baseline controller must be verified or even correct, instead also treating it as a black-box component. This change is important; there are many types of powerful control techniques—model predictive control and neural network controllers—that often work well in practice, but are unlikely to be formally proven correct due to complexity. We prove such an architecture maintains safety, and present case studies where model-predictive control provides safety for multi-robot coordination, and unverified neural networks provably prevent collisions for groups of F-16 aircraft.

KEYWORDS

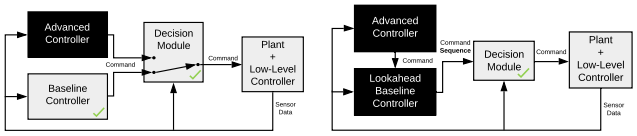
Simplex Architecture, Formal Verification, Cyber-Physical Systems

1 INTRODUCTION

Modern cyber-physical systems (CPS) are found in vital domains such as transportation, autonomy, health-care, energy, agriculture, and defense. As these systems perform complex functions, they require complex designs. Since CPS interact with the physical world, correctness is especially important, but formal analysis can be difficult for complex systems.

In the design of such CPS, powerful techniques such as model-predictive control and deep reinforcement learning are increasingly being considered instead of traditional high-level control design. Such trends exacerbate the safety verification problem as classical verification strategies are poorly suited for these new designs.

One approach for dynamically providing safety for systems with complex and unverified components is *runtime assurance* [2], where the state of the plant is monitored at runtime to detect possible imminent violations of formal properties. If necessary, corrective measures are taken to avoid the violations. A well-known runtime assurance technique is the Simplex Architecture [6], which has been applied to a wide range of systems. The Simplex Architecture, shown in Figure 1(a), guarantees safety of the plant by deploying a verified safe *baseline controller* (BC) in conjunction with the performance oriented *advanced controller* (AC). The *decision module* (DM) periodically monitors the state of the system and switches the control from AC to BC if a safety violation is imminent.



(a) Traditional Simplex Architecture **(b) Black-Box Simplex Architecture**

Figure 1: The Black-Box Simplex Architecture guarantees safety despite a black-box AC and a black-box BC.

The successful application of the original Simplex Architecture requires creating a provably safe BC, a difficult task for many systems. Further, many classes of controllers, such as those designed using model-predictive control, rapidly-exploring random trees, or neural-network controllers, may work well in practice, but are difficult to verify and therefore cannot be used as BCs. **The main contribution of this work is to overcome this limitation.** We propose the *Black-Box Simplex Architecture*, a variant of the traditional Simplex Architecture that can guarantee the safety of the system despite an unverified and even incorrect BC, which is treated as a black box.

In the Black-Box Simplex Architecture, shown in Figure 1(b), the BC tries to produce a *sequence* of commands that begins with the AC's current command and brings the plant to a state where maintaining the safety property is easy. If the DM is able to verify the safety of the proposed command sequence, it is stored for potential use as a backup plan at the next time step and the AC controls the plant at the current time step. Otherwise, safe command sequence stored in the previous time step takes over the control until another safe command sequence is produced.

We demonstrate the effectiveness of the proposed approach by successfully applying it for collision avoidance on a multi-robot coordination system and on groups of F-16 aircraft. Further, we formally establish that the Black-Box Simplex Architecture guarantees safety of the system. In this paper, we will briefly discuss the results of the case studies and introduce the Black-Box Simplex. For further details on the case studies and the safety theorem we refer the readers to the full paper [5].

2 BLACK-BOX SIMPLEX

We consider discrete-time plant dynamics, modeled as a function $f(x_i, u_i, w_i) = x_{i+1}$ where $x_i \in \mathcal{X}$ is the system state, $u_i \in \mathcal{U}$ is a control input command, $w_i \in \mathcal{W}$ is an environmental disturbance, and $i \in \mathbb{Z}^+$ is the time step.

The Black-Box Simplex Architecture, shown in Figure 1(b), lifts the requirement that the BC is verified, allowing provable safety

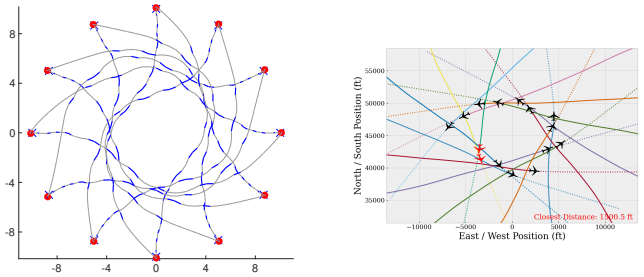


Figure 2: *Left*: 12 robots reach targets safely. The trajectory segments where stored BC commands are used are shown in blue. *Right*: 15 aircraft cross the center of the circle while maintaining the 1500 ft separation distance.

with both an unverified AC and an unverified BC. Apart from eliminating the need to establish safety of the BC, the Black-Box Simplex Architecture differs from the traditional Simplex Architecture in other important ways. First, the AC shares its command with the BC instead of passing it directly to the decision module. Second, the BC uses this command as the starting point of a *command sequence* intended to safely recover the system. Many control techniques naturally produce command sequences, such as model predictive control with a finite-step horizon or controllers derived from rapidly-exploring random trees (RRTs). If a model of the low-level controllers and plant is provided, a traditional single-output controller can be used to create a command sequence through repeated invocations and system simulation.

The decision module checks the BC's command sequence, possibly rejecting it if safety is not ensured. As long as the AC drives the system through states where the BC can recover, it continues to actuate the system. However, if the BC fails to compute a safe command sequence, due to the fault of either the unverified AC or the unverified BC, the decision module can still recover the system using the safe command sequence from the previous step.

The applicability of Black-Box Simplex depends on the feasibility of two system-specific steps: (i) constructing safe command sequences and (ii) proving their safety at runtime. For some systems, a safe command sequence can simply bring the system to a stop. An autonomous car, for example, could have safe command sequences that steer the car to the side of the road and then stop.

Proving safety of a given command sequence can also be challenging and depends on the system dynamics. For nondeterministic systems, this could involve performing reachability computations at runtime [1, 4]. Such techniques assume an accurate system model is available in order to compute reachable sets. In the Black-Simplex Architecture, although both controllers are unverified, we do not combine them into a single unverified controller for two reasons. First, the design of the safety controller is easier if it is kept simple and is not burdened with fulfilling all mission-specific goals. Second, it allows for the use of off-the-shelf controller strategies that are focused on either mission completion or safety.

3 CASE STUDIES

We show-case the results for two case studies: a multi-robot coordination system, and a mid-air collision avoidance system for groups of F-16 aircraft. In both case studies, in the initial state the agents are evenly-spaced on a circle and the AC's commands will cause a collision at the center of the circle. The safety property is that the aircraft and robots are to maintain a separation of a fixed value between them.

For the multi-robot coordination study, the robots are modelled as points with double-integrator dynamics. They are to reach a target located on the opposite side of the initial circle. Both the AC and the BC are designed using centralized model predictive control (MPC), which produces command sequences as part of the solution of an optimization problem: hence prone to failure as the optimal solutions are not guaranteed by the numerical non-linear methods. Nevertheless, as shown in Figure 2, the Black-Box Simplex is able to prevent all collisions while all robots are able to reach their targets.

For the multi-aircraft study, each aircraft is modeled with 16 state variables, including positional states, positional velocities, rotational states, rotational velocities, an engine thrust lag term, and integrator states for the low-level controllers. The model includes high-level autopilot logic for waypoint following, which is used in the AC. For the BC, we build upon the ACASXu system designed for unmanned aircraft [3] which issues horizontal turn advisories based on the relative positions of two aircraft, an *ownship* and an *intruder*. We adapt this system to the multi-aircraft case by having each aircraft run an instance of ACASXu against every other aircraft. To create command sequences, we advance the plant model and rerun ACASXu from the future state multiple times in a closed-loop fashion. The ACASXu system is neural network compression of a large look-up table and hence is unsafe. Our results show that the aircraft maintain the desired horizontal separation.

4 CONCLUSIONS

We have presented the Black-Box Simplex Architecture, a methodology for constructing safe CPS from unverified black-box high-level controllers. The main tradeoff present in Black-Box Simplex is that the decision module has increased complexity, and for the system to perform smoothly, it must be able to quickly verify command sequences. This itself is not an easy problem. With the proposed approach, however, we have reduced the difficult problem of proving high-level safety to a simpler problem of *performance optimization* of the decision module logic. Black-Box Simplex provides a feasible path for the verification of systems that are otherwise unverifiable in practice.

REFERENCES

[1] Stanley Bak, Taylor T. Johnson, Marco Caccamo, and Lui Sha. 2014. Real-Time Reachability for Verified Simplex Design. In *35th IEEE Real-Time Systems Symposium (RTSS 2014)*. IEEE Computer Society, Rome, Italy.

[2] Matthew Clark, Xenofon Koutsoukos, Joseph Porter, Ratnesh Kumar, George Pappas, Oleg Sokolsky, Insup Lee, and Lee Pike. 2013. *A study on run time assurance for complex cyber physical systems.* Technical Report. Air Force Research Laboratory, Aerospace Systems Directorate.

[3] Mykel J Kochenderfer and JP Chryssanthacopoulos. 2011. Robust airborne collision avoidance through dynamic programming. *Massachusetts Institute of Technology, Lincoln Laboratory, Project Report ATC-371* 130 (2011).

[4] Qin Lin, Xin Chen, Aman Khurana, and John Dolan. 2020. ReachFlow: An Online Safety Assurance Framework for Waypoint-Following of Self-driving Cars. In *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*.

[5] Usama Mehmood, Stanley Bak, Scott A. Smolka, and Scott D. Stoller. 2021. Safe CPS from Unsafe Controllers. arXiv:2102.12981 [cs.SE]

[6] L. Sha. 2001. Using Simplicity to Control Complexity. *IEEE Software* 18, 4 (2001), 20–28. https://doi.org/10.1109/MS.2001.936213