From Safety to Guilty & from Liveness to Niceness

Stefan Mitsch*, Jan-David Quesel, André Platzer
Computer Science Department, Carnegie Mellon University, 5000 Forbes Ave, Pittsburgh, PA 15213
Email: {smitsch,jquesel,aplatzer}@cs.cmu.edu

***Abstract*—Robots are solving challenging tasks that we want them to be able to perform (*liveness*), but we also do not want them to endanger their surroundings (*safety*). *Formal methods* provide ways of proving such correctness properties, but have the habit of only saying "yes" when the answer is "yes" (*soundness*). More often than not, formal methods say "no": They find out that the system is neither safe nor live, because there are "unexpected" circumstances in which the robot just *cannot* do what we expect it to. Inspecting those unexpected circumstances is informative, and identifies constraints on reasonable behavior of the environment. This ultimately leads from safety to the question of who is guilty depending on whose action caused the safety violation. It also leads from liveness to the question of what behavior of the environment is nice enough so that the robot can finish its task.**

I. Formal Methods for Robotics

Robots often interact with a dynamically changing environment and in close proximity to humans or critical infrastructure. Thus, safety is key. But we also want a robot to complete some useful task or achieve a particular goal (liveness).

Formal verification methods help to exhaustively analyze a robot and its control algorithms for correctness. This paper is based on our experience with formal verification of safety and liveness properties of autonomous robotic ground vehicles [2]. The overall challenge arises, because robots not only execute discrete (control) algorithms, but they also interact with the real world through sensors and actuators. For verification purposes, thus, we need to take the discrete control algorithms and the continuous physical behavior of both our own robot and the environment into account. *Hybrid systems* are a suitable mathematical model to describe systems with interacting discrete and continuous behavior. We focus on theorem proving for hybrid systems as verification method, and use *differential dynamic logic* [4] implemented in the KeYmaera prover [5] and the modeling tool Sphinx [3] to illustrate the challenges that arise from analyzing safety and liveness questions.

II. Safety: Always Stay Safe

An important and useful consequence when applying formal verification to robotics and other systems is that we are forced to define unambiguously what we mean by safety.

a) What is Safe?: Often, pure safety conditions are unachievable (e.g., "never collide"), because they can no longer be guaranteed if other agents in the environment actively try to ruin safety. What we usually mean instead is safety w.r.t. some *norm* or global system invariant (i.e., our own robot behaves such that it stays safe *if* the environment is reasonable). If everybody follows such a norm—i.e., nobody shows *abnormal* behavior guilty of a violation—then everyone is safe.

b) Whom to Blame?: For safety verification of the dynamic window collision avoidance algorithm in the presence of moving obstacles [2], we used *passive safety* and *passive-friendly safety* [1] as the requisite safety conditions.

For passive safety, robots follow the norm of avoiding collision by just stopping. In case of a collision, we blame the agent that was not stopped. But this norm falls short of "correctly" attributing blame when some of the agents can only react slowly and our own robot should know better (e.g., move quickly to a stop right in front of a train).

And still there are situations where these safety norms may result in rather conservative or sometimes even surprising behavior, because the safety conditions lack a notion of orientation of agents towards each other. For example, our own robot would try to stop even if an obstacle approaches the robot from behind, which makes collision avoidance maneuvers for obstacles even more difficult. Such orientation relationships and other spatial configurations involving multiple agents pose an interesting challenge for formal verification techniques.

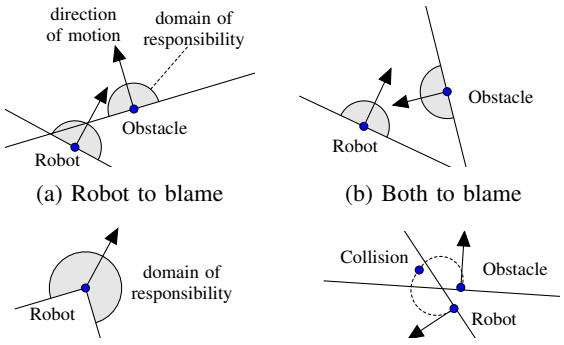
c) The Role of Spatial Configuration in Blame: In order to formulate norms we typically impose certain spatial structure (e.g., traffic lanes or 4-way stop intersections).

For instance, consider highway traffic, where rear-ending another car would certainly be deemed unsafe. Yet, if one car cuts in front of another, who is to blame for a resulting crash? Similarly, if a car turns from a parking lot onto a street, traffic laws clearly capture the liability in case of an accident, so drivers behave accordingly. Traffic participants rely on these assumptions. Otherwise, cars would need to slow down significantly at each driveway since a collision might be imminent. In order for such norms to work safely, the assumptions have to be acyclic and mutually consistent.

Even though traffic rules are well-known, tested for centuries, and can use structure manifested in real-world (e.g., lanes or road signs as area delimiters) they are already complicated. Assigning blame in unstructured two- or three-dimensional space gets even more complicated. For example, let us consider a simple orientation notion as in Fig. 1. Here, an agent is to blame if it collides with an object that was in its domain of responsibility when the agent initiated a maneuver (different notions of domain of responsibility are possible, e.g., along the sensor range of a Lidar, cf. Fig. 1c). This rule may fail if both agents can make fast sudden turns, as in Fig. 1d. In summary, we argue that finding an appropriate spatial structure is crucial to formalize and verify useful safety conditions.

* Currently on leave from Johannes Kepler University, Linz, Austria

In Calin Belta and Hadas Kress-Gazit, editors, 5th Workshop on Formal Methods for Robotics and Automation, 2014.



(a) Robot to blame

(b) Both to blame

(c) Sensor-dependent domain of responsibility

(d) Collision, but both not considered guilty

Figure 1: Orientation scenarios

d) The Role of Temporal Configuration in Blame: Blame cannot always be evaluated at a single instant of time. Consider the following two cases: (1) a car cuts in front of another one, which leads to a collision and (2) a car blindly rear-ends the car in front because the driver fell asleep. These two scenarios are indistinguishable if we only consider the moment of collision. However, knowing the course of events leading to the collision enables us to assign blame properly. Thus, in case (1) the car that was rear-ended was to blame, whereas it was behaving correctly in case (2).

Hence we conjecture that the system model needs to keep a record of certain situations in order to avoid situations where it is to blame in case a collision is imminent.

e) Unique Guilty Party: If for every collision scenario there is a unique guilty party then the local optimizing behavior of robots will ensure global safety. We say, a single robot behaves correctly if it is never *causing* a collision rather than never being involved in one. Hence if all robots behave correctly according to this definition, there will not be collisions when they are avoidable from the initial configuration. Overall, this could also have interesting consequences for cases where a robot is not even involved in the ultimate safety violation but still to blame for it. Again, think of a car cutting in front of another. The second car, who is not able to brake in time, might instead decide to avoid collision by performing a lane change as well. However, this might cause a collision between the second car and a third car, which does not involve the accident perpetrator itself: the first car. A further open question is how to avoid safety violations caused by uncoordinated actions of multiple robots that each on their own cannot identify themselves as contributing to a safety violation.

III. Liveness: Ultimately Get to the Goal

Liveness, such as whether a robot ultimately reaches its goal for any behavior and layout of the environment, is an even harder question than safety, because there are many ways that environmental conditions or behavior prevent the robot from achieving its goals. For example, an obstacle that is allowed to move arbitrarily could always block the robot's path. To guarantee liveness, one has to characterize *all necessary conditions* that allow the robot to reach its goal, which are often prohibitively many.

The change in perspective from safety to guilty corresponds to the change from liveness to niceness, i.e., the study of whether a robot is guaranteed to achieve its goal assuming the environment is reasonably nice or even supportive. Yet, while global safety may follow from the conjunction of local safety constraints, the pattern falls short for liveness properties in most cases. If the goal of one robot is to indefinitely occupy a certain space and that of another is to reach this very space there is no possible way for both to satisfy their requirements. Unfortunately, these conflicting situations are not always as easily identifiable. Therefore, if there are situations where we cannot get to the goal, not even in a nice environment, another common criterion in robotics is that of having invested *reasonable effort* of trying to get to the goal.

Another challenge in liveness is how to measure *progress*. In the presence of obstacles a robot sometimes needs to move away from the goal in order to ultimately get to the goal. But how far is a robot allowed to deviate on the detour?

We argue that spatial and temporal contracts again help ensure liveness properties. For example, if n robots operate on n disjoint connected sub-spaces of a plane, each one of them can reach every position within its own sub-space as long as all robots respect the sub-space boundaries. More complex temporal contracts are necessary for cases where the spatial regions intersect, so that every robot can make progress, just at different times. For example, we could allow boundary violation for a limited time per robot. Then eventually, each robot will be able to operate within its sub-space undisturbed.

IV. Conclusion

Safety and liveness are important properties for robotic systems. Their specification is by no means trivial. For traffic scenarios, the existing laws provide helpful advice which corner cases need to be considered for safety. Still, lack of structure in general two- or three-dimensional space makes it difficult already to capture formally who is at fault in case of a collision, so that robots know how they need to act in order to avoid being guilty of a safety violation. Even worse, for liveness properties the possibilities of failing are almost endless, which makes it highly non-trivial to list all necessary conditions to ensure progress towards a specific goal.

Acknowledgments: This material is based upon work supported by the NSF (NSF CAREER Award CNS-1054246, NSF EXPEDITION CNS-0926181, CNS-1035800, CNS-0931985), by DARPA (FA8750-12-2-0291), by the US DOT UTC T-SET (DTRT12GUTC11), and by the EU REA (FP7 Marie Curie Actions PIOF-GA-2012-328378).

References

[1] Kristijan Maček, Dizan Alejandro Vasquez Govea, Thierry Fraichard, and Roland Siegwart. Towards Safe Vehicle Navigation in Dynamic Urban Scenarios. *Automatika*, 50(3–4):184–194, 2009.

[2] Stefan Mitsch, Khalil Ghorbal, and André Platzer. On Provably Safe Obstacle Avoidance for Autonomous Robotic Ground Vehicles. In *Robotics: Science and Systems*, 2013.

[3] Stefan Mitsch, Grant O. Passmore, and André Platzer. Collaborative Verification-Driven Engineering of Hybrid Systems. *Math. in Comp. Sci.*, 2014. doi: 10.1007/s11786-014-0176-y.

[4] André Platzer. Logics of Dynamical Systems. In *LICS*, pages 13–24. IEEE, 2012. doi: 10.1109/LICS.2012.13.

[5] André Platzer and Jan-David Quesel. KeYmaera: A Hybrid Theorem Prover for Hybrid Systems. In Alessandro Armando, Peter Baumgartner, and Gilles Dowek, editors, *IJCAR*, volume 5195 of *LNCS*. Springer, 2008.