Chapter 7

Conclusions and Future Directions

7.1 Summary of Contributions

As autonomous robotic systems are designed to take on more difficult tasks, the complex fault tolerant control systems need to be verified for safety. Model checking is a popular verification approach; several automatic symbolic model checkers are available. However, model checkers for hybrid automata with a continuous state space suffer from the inability to handle the state space explosion that ensues. One way to decrease the effect of the state space explosion is to impose some structure on the robotic control system that allows it to be verified more efficiently.

A goal-based fault tolerant control architecture based on MDS was analyzed in this work. Three methods of safety verification of the goal network control systems, two deterministic and one stochastic, were introduced. The first method was an automatic goal network conversion procedure that connected goal network executions to a hybrid automaton structure via a bisimulation when simple ordering on the goal network's time points was imposed. Then, existing symbolic model checkers, particularly PHAVer, were used to complete the verification. Since the hybrid automaton captured every possible executable set of goals in the goal network as a location, the number of locations was roughly exponential with the number of root goals in a goal network. The conversion procedure, written in Mathematica, was able to convert large goal networks given enough time, however, PHAVer was not always able to verify the converted automaton without state space reductions and abstractions. In general, the number of passive state variables that control failure transitions in the goal network is the limiting factor in PHAVer verification as the state space explosion depends strongly on that. The efficiency of the conversion algorithm also depends strongly on the number of passive state variables because the number of failure transitions from a location grows exponentially with the number of passive state variables constrained in that location.

These complexity issues drove the creation of new design for verification and verification algorithms. When the goal network has state-based transitions, which occurs when each possible passive state satisfies the passive constraints in some set of goal tactics for each group of goals, the invariants of the locations of the converted hybrid automaton contain all the information needed to find every possible transition between the locations. Therefore, the transitions do not need to be found explicitly and the reachability of the locations in a group depends only on the reachability of the states of the passive state variables constrained. This allows the locations only of the converted hybrid automaton to be searched for ones whose invariants and rate conditions satisfy the unsafe conditions. If the passive states constrained in these unsafe locations are reachable from the initial conditions on those state variables, then the unsafe locations are reachable. If one or more of the passive state variables constrained are continuous, rate-driven dependent state variables, such as power or temperature, a path through the discrete sets of states of those state variables from the initial condition to the unsafe condition must be found to prove reachability. However, this path search is simplified by the reachability properties of the automaton due to the state-based transitions requirement. Two software algorithms were introduced, the design for verification tool, SBT Checker, and the verification software, InVeriant. Though these software algorithms were developed for the verification of goal networks, they also can efficiently verify a class of hybrid systems.

The SBT Checker and InVeriant software combination proved to be a more efficient and effective verification method for goal networks. First, the provable modularity of the state-based transitions property allows for distributed design of goal networks. Designers can use the SBT Checker tool to create goal trees that have state-based transitions. The design can be iterative as the software provides nearly instantaneous feedback about which state constraints are missing from the goal tree. The goal network that is the combination of these goal trees is guaranteed to have state-based transitions as long as the controlled constraints are consistent. The state-based transitions and consistent controlled constraints requirements are very useful checks because, instead of being restrictive, they are good design practices that ensure that the tactics in the control system cover every possible modeled passive state. The consistent controlled constraints requirement is checked in the InVeriant algorithm when the locations are created during the goal network conversion. Since the transitions are not created and since most passive state models are not incorporated into the automaton being verified, the complexity issues that result from the number of passive state variables do not affect this algorithm. This allows for larger systems to be verified quickly and effectively. The output of the InVeriant software gives not only the unsafe locations but the set(s) of goals that are

common to them, which aids in the redesign of faulty control systems. For certain unsafe locations, InVeriant will also find an appropriate path to prove reachability.

Fault tolerant systems designed with the concept of state-based transitions are very dependent on the quality of the state estimators for the passive state variables. Methods to calculate the failure probability of a system due to its estimation uncertainty are discussed. These methods are very useful to the design of a system as the result is a measure of how much the tactics depend on faulty estimators or hard to measure state variables, but they are severely limited by complexity issues. However, there are many ways to abstract the problem and there is even an automatic computational algorithm. In many cases, this sort of analysis is ignored, even though it is so important for autonomous systems whose reliance on estimated state values based on sensor measurements is an absolute.

Finally, two significant example goal networks were presented. The complex rover and Titan aerobot examples were verified versus unsafe sets using the conversion/PHAVer method and the SBT Checker/InVeriant method, respectively. These examples tested and improved the conversion procedure and the Titan aerobot example drove the design of the novel verification procedure due to the inability of PHAVer to verify the problem using simple abstractions and model reduction techniques.

This dissertation presents a significant study of the verification of goal-based control programs. The conversion of goal networks to hybrid systems allows for model checking techniques to be applied. A design for verification tool was developed and has great potential for the design and verification of real-world goal-based control systems and linear hybrid automata. The ensuing verification method is efficient enough to handle large goal networks and hybrid systems. Finally, the estimation uncertainty analysis is a novel concept that may lead to useful estimator or goal network control system design techniques.

7.2 Future Directions

There are several directions in which this work can continue. First, the restrictions on time points imposed nearly immediately on the goal network's structure can cause interesting problems in cases where going back or redoing a part of a tactic is required. There may be ways to loosen the time point restrictions imposed while maintaining the important group structure of the goal networks. An important benefit of MDS is its use of projections, which is not included in the goal networks used

for verification. It may be possible to place projection constraints with the rate conditions in each location upon conversion and it may be possible to reason about those constraints in the search for unsafe locations.

The concept of state-based transitions applies to goals networks as well as hybrid automata. When the dynamics in a set of hybrid automata are sufficiently simple, it is possible to use the SBT Checker and InVeriant verification method with the hybrid system directly. In fact, the class of hybrid systems upon which this method could be applied is broader than the class that is bisimilar to goal network control systems. This extra capability of the verification method seems very useful and should be explored more fully.

The two deterministic verification methods include software that is written in Mathematica. Though its kernel is fairly fast, it may be more efficient to convert the software into Java. While the conversion procedure has been tested fairly extensively, the SBT Checker and InVeriant could benefit greatly from more testing, especially on more complex problems. A test of the entire process from design to verification of a control system for a real-world system would be extremely useful.

The failure probability due to state estimation uncertainty is a very important tool in the analysis of a system, and it seems that one should be able to discover very specific feedback on how to redesign a system based on this analysis. Some possible ways to redesign a system include installing better sensors, designing more accurate estimators, or reducing the control dependence on particular state variables. Currently, there is no process or set of guidelines available to aid the analyst in making these design determinations, though this seems to be possible.

The verification emphasis of this work is on safety. Liveness properties, however, are also very important to check. Spin is a model checker that can verify liveness properties of discrete automata. A goal network with unimportant or no continuous state variables can be converted into an automaton specified in Promela code. It can then be verified versus an unsafe Buchi automaton using Spin. However, since Spin is not a symbolic model checker, the state space complexity issues can be even more restrictive. More work could be done to discover a better abstraction of goal networks so that Spin is more effective in their verification. Another benefit of Spin is its non-deterministic execution model, which could make it a good fit for verifying multiple robot systems.

The overall goal of this work and others on verification of autonomous control systems is to find an effective way to design these systems so that they work in all foreseeable situations (and some that are not). It is the author's belief that verification work must begin during the requirements stage of the system design and continue throughout the creation of the autonomous system. Therefore, the

tools and concepts of model checking must be integrated into the system architecture and design; at the same time, there must be enough flexibility to allow for complex behaviors to emerge in the autonomous systems. While the goal structure imposed in this work does not have the necessary flexibility, some important concepts, particularly the modularity of state-based control transitions, have been discovered and will be influential in future work towards the design of robust, fault-tolerant autonomous control systems.