Chapter 6

Significant Goal Network Verification Examples

6.1 Introduction

The three verification methods introduced so far are applied to two examples in this chapter. The first example is a goal network for a complex rover system that is based on autonomous robotic systems such as the DARPA Grand (or Urban) Challenge vehicles. This example has twelve state variables and twenty-one controlled goals, so is not too large for the conversion procedure and PHAVer verification method. It is difficult to apply the failure probability calculation method without some model reduction techniques, however. This example is described in Section 6.2. The second example, described in Section 6.3, is a goal network for an example mission to Titan, a moon of Saturn. The autonomous aerobot probe must explore the lower atmosphere of Titan and map its surface while staying safe and performing other tasks. This example is sizable; over 500 locations are found using the conversion procedure. The number of passively constrained state variables after some model reduction techniques is nine, but that proves to be too much for the conversion procedure because of the number of failure transitions. The conversion procedure is able to convert the goal network after applying a restricting assumption, but even then, PHAVer is not able to verify the system (though resorting to overapproximation and other abstraction techniques may have helped). So, this example is redesigned and verified using the novel verification method described in Chapter 4. The failure probability calculation must be approximated for this example following the Markov Chain Monte Carlo technique outlined in Chapter 5.

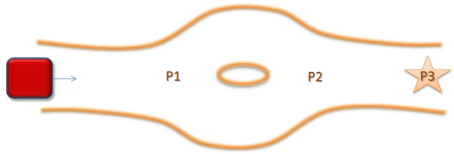

Figure 6.1: A depiction of the example task where the rover must traverse a path and reach the end point, which is marked with a star

6.2 Complex Rover Example

This example is based on a possible set of commands for autonomous rovers such as Mars exploration robots or DARPA Grand Challenge vehicles. The problem size is small but the system has enough complexity to begin to test the capabilities of the conversion software and the verification method.

6.2.1 Goal Network Design

This example involves an autonomous rover whose ultimate goal is to follow a given path to a specified end point, shown in Figure 6.1. This rover has three main sensor systems: GPS, LADAR, and an inertial measurement unit (IMU). The path choice and speed limit along the chosen path is dependent on the combined health of these sensors. Each sensor degrades or fails in a specified way. The GPS can experience periods of reduced accuracy (satellite dropouts) or failure (electrical or structural signal interference), and these can both be modeled as recoverable stochastic events. The health of the LADAR depends on the location of the sun in the sky. If the sun is shining directly into the LADAR, its measurements cannot be used. Some degradation of the LADAR's capabilities occur at less direct sun angles, as well. Finally, the health of the IMU is dependent on the temperature of the device. If the temperature of the IMU is too low, a heater can be used to heat the sensor. If the IMU temperature gets too high, the unit must be powered off. The state effects diagram listing all the state variables important to the system is shown in Figure 6.2. State variables, measurements, and commands are shown and the arrows between these indicate modeled effects on the accepting state variables or measurements.

The state variable types for each state variable can be found in Table 6.1. The goal network for this task is shown in Figure 6.3 and the individual goal trees are shown in Figures 6.4–6.6. The first goal tree describes the path the robot will take, the second constrains the speed limits that will apply to the robot, and the third describes the IMU temperature management method.

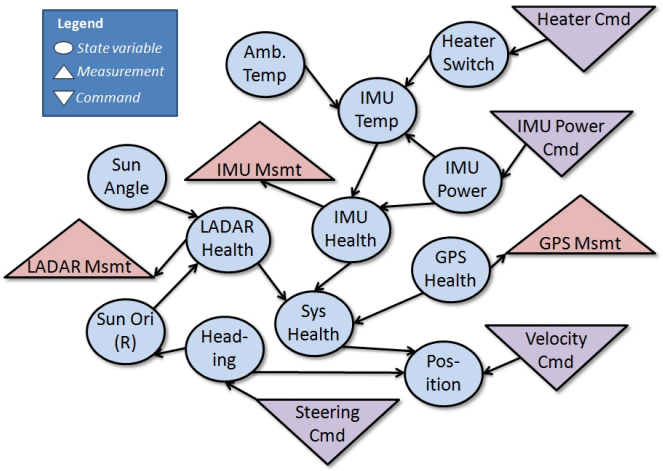

Figure 6.2: State effects diagram for the complex rover example

Table 6.1: State Variable Data

State Variable	Abbreviation	Type
Position	x	Controllable
Heading	θ	Controllable
IMU Power	ps	Controllable
Heater Switch	hs	Controllable
IMU Health	IH	Dependent
Rel. Sun Orientation	SO	Dependent
LADAR Health	LH	Dependent
System Health	SH	Dependent
IMU Temperature	IT	Dependent
GPS Health	GH	Uncontrollable
Ambient Temp.	AT	Uncontrollable
Sun Angle	SA	Uncontrollable

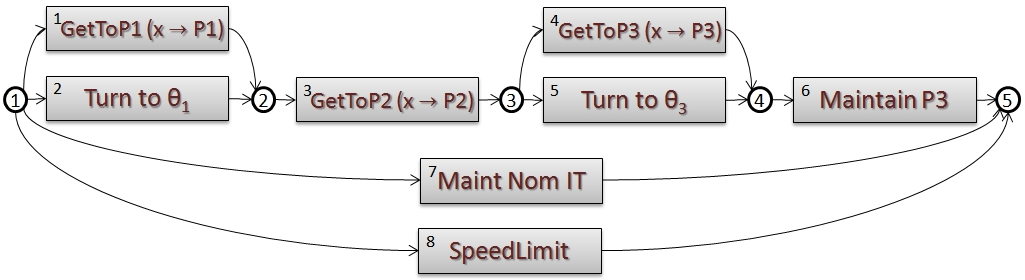

Figure 6.3: Goal network for the complex rover example

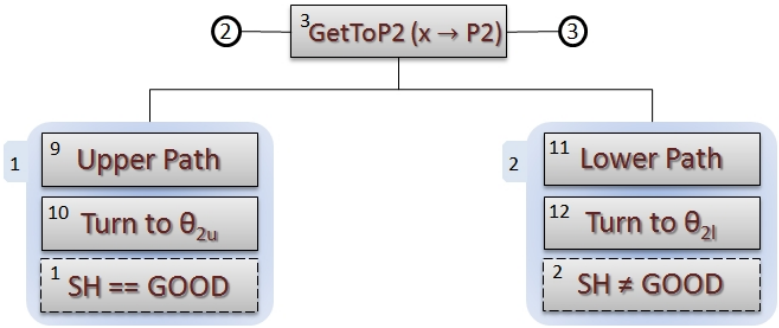

Figure 6.4: Path goal tree

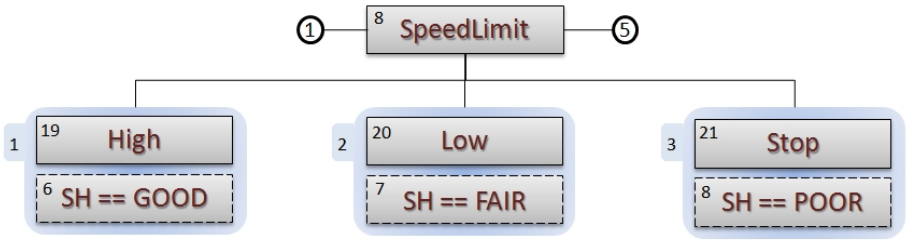

Figure 6.5: Speed limit goal tree

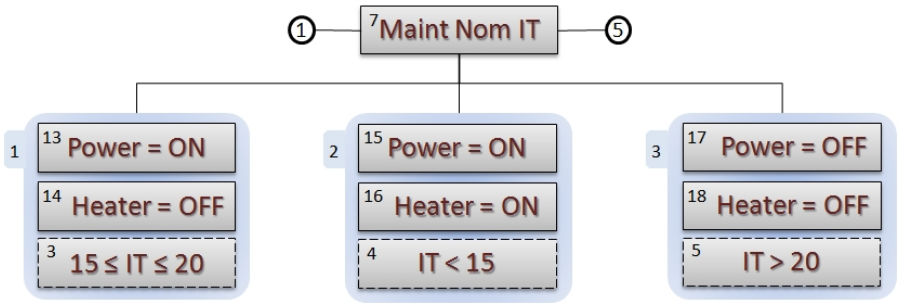

Figure 6.6: IMU Temperature goal tree

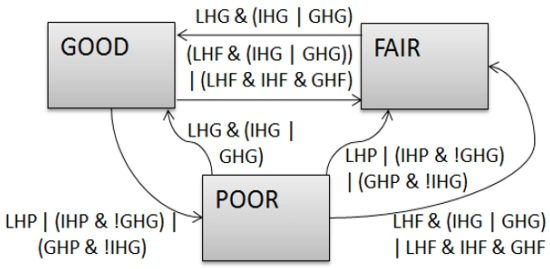

Figure 6.7: A finite state machine that describes the model of the `SystemHealth` derived state variable. IHG = `IMUHealth` is `GOOD`, GHF = `GPSHealth` is `FAIR`, LHP = `LADARHealth` is `POOR`, etc.

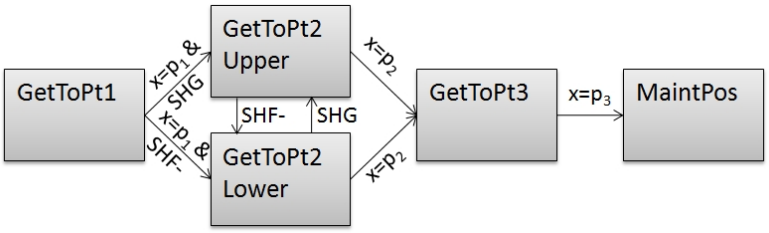

Figure 6.8: The path automaton. One of three automata that are composed into the control system for the rover. These four sets of locations represent groups 1–4.

6.2.2 Conversion and Verification

The goal network control program for this example consists of twenty-one controlled goals, including eight root goals, and twelve state variables. The conversion software found thirty-eight locations (including the Success location) in four groups in the goals automaton. Figures 6.8 and 6.9 show the hybrid systems whose composition creates the goals automaton. In addition to this, there are eight other automata that describe the state models of the dependent and uncontrollable state variables. One of these, the `SystemHealth` state variable model, is shown in Figure 6.7. In all, the composition of the nine automata creates a discrete state space with over 200,000 states. The unsafe set for this problem consists of the following conditions:

1. The rover is not stopped ($\dot{x} \neq 0$) when the IMU is off (`ps` $==$ `OFF`) and the GPS is degraded (`GH` $\neq$ `GOOD`).

2. The rover moves forward ($\dot{x} \neq 0$) when the sun is pointing directly into the LADAR unit (`LH` $==$ `POOR`).

While the conversion software took less than five seconds to generate the PHAVer code for this system, PHAVer was not able to handle the large state space without resorting to overapproximation.

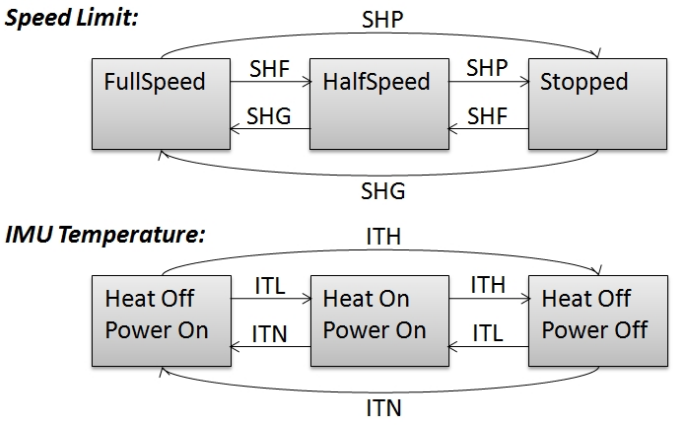

Figure 6.9: The speed limit and IMU temperature automata; all locations in the two automata span groups 1–3

So, several reduction techniques were employed. The first reductions were in the uncontrollable and dependent state variable automata. The `SunAngle` and `RelativeSunOrientation` state variables, which were modeled with stochastic transitions, were removed and the `LADARHealth` state variable's model became stochastic. Since these two state variables were not used elsewhere in the goal network, this simplification did not affect the quality of the model. Next, the `AmbientTemperature` state variable was removed as was the `IMUTemperature` state variable's dependence on it, simplifying the model. In the last model reduction, the `SystemHealth` state variable was removed in favor of using the three sensor health state variables in its place. These reductions made the discrete state space a more manageable 3726 states. The final reduction was to verify the goals automaton group by group, which is possible because the unsafe set did not constrain the progress of the `Position` or `Orientation` state variables, though the initial condition problem must be handled carefully.

The goals automaton could be verified after making some corrections. The verification software found reachable states in the automata that entered the unsafe set, so the goals automaton had to be corrected to ensure that the unsafe set was not entered. The transitions into the locations where the `IMUPower` is off (`ip` $==$ `OFF`) and the speed is not zero ($\dot{x} \neq 0$) must also have a condition that the `GPSHealth` is `GOOD` or `FAIR` (`GH` $\neq$ `POOR`)) to satisfy the unsafe set. These changes were added, verified, and then translated back into the goal network by adding a new tactic in the speed limit goal tree, which can be found in Figure 6.10. This makes the control program conservative (more states than necessary are constrained to have zero rate) but verifiable with respect to the given unsafe set.

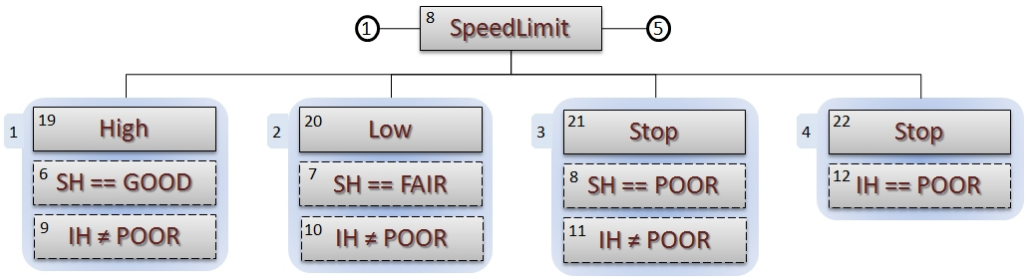

Figure 6.10: Redesigned speed limit goal tree

6.2.3 Uncertainty Analysis

The goal network verified in the previous section can now be analyzed for safety in the presence of state estimation uncertainty. The unsafe set specified in the previous section continues to be the set of conditions that the uncertainty analysis will use. Since the first three groups, V_1, V_2, and V_3, are essentially the same set of locations repeated, only V_1 will be analyzed. The velocity in V_4 is constrained to be zero, so there is no way to enter the unsafe set based on the uncertain state variables; the failure probability for this group is $W_s(4) = 0$. The system has four uncertain state variables (`IMUTemperature`, `GPSHealth`, `LADARHealth`, and `IMUHealth`), each with three possible state values ($\{$`GOOD`, `FAIR`, `POOR`$\}$ for the health state variables and $\{$`LOW`, `NOMINAL`, `HIGH`$\}$ for `IMUTemperature`). Since that translates into $3^8 = 6561$ complete system states, simplification is necessary. Instead, if the `SystemHealth` state variable with three states replaces the two of the sensor health state variables and a two state `IMUHealth` is used ($\{$`POOR`, `NOTPOOR`$\}$), the number of complete system states reduces to $2^2 * 3^4 = 324$.

Another observation is that the `IMUHealth` depends on the `IMUTemperature`. Since there is a model that controls what the `IMUHealth` is estimated to be given the `IMUTemperature`, the estimated `IMUHealth` is known given the `IMUTemperature`. However, the actual `IMUHealth` may not always be known given the actual `IMUTemperature` due to modeling errors. In certain cases, such as when the estimated `IMUTemperature` causes the `IMUPower` to be turned OFF, that the actual `IMUHealth` state is known given the estimated `IMUTemperature`. This dependence of an uncertain state variable on another causes the number of complete system states to be further reduced. Dependencies between two state variables is handled by creating a new composition state variable that consists of all the possible actual and estimated states that the two variables can take given the dependencies. For this problem, the new state variable is called `TI` and has 18 states, which are made up of estimated and actual values of `IMUTemperature` and `IMUHealth`.

With the nine possible states for the estimated and actual values of the `SystemHealth` state variable, the new total number of complete system states is $18 * 9 = 162$.

The group V_1 is a non-uniform completion group with three different contribution values, 1, $\frac{1}{2}$, and 0, that correspond to the Full Speed, Half Speed, and Stopped speed limit tactics, respectively. The completion time for V_1 is assumed to be $c_1 = 5$. Based on the unsafe set specification and the composed hybrid automaton control system, the nominal set, Ξ_1, has 120 complete system states, the Safing set, F_1, is empty, and the unsafe set, Ω_1 contains the remaining 42 complete system states. The state transitions of the two remaining uncertain state variables are modeled as stationary Markov processes; in this case the models were chosen to have what was considered to be realistic values, but in practice, these models would be chosen based on simulations or tests of the hardware. For this example, several different estimator uncertainty values were chosen and the failure probability was calculated for each. Using these models, uncertainty values, and the sets of complete system states, the appropriate vectors and matrices were calculated. The initial failure probability, a_1, is the sum of the initial probabilities of all 42 unsafe complete system states. There are three initial probability vectors W_1^i corresponding to the three contribution values; the dimensions of W_1^1 is 1×6, W_1^2 is 1×12 and W_1^3 is 1×102. There are nine Q_1 matrices between the β groups and three $W_{u,1}$ vectors, with the same dimensions as the W_1 vectors.

Since this case has a set of locations that has zero contribution value, there are an infinite number of failure paths. However, by using the power series equation,

$$\sum_{x=0}^{\infty} Q^x = (I - Q)^{-1}, \tag{6.1}$$

each failure path can be accounted for in the failure probability calculation. The failure probability was calculated for several values of estimation uncertainty and the results are shown in Fig. 6.11.

6.3 Titan Aerobot Example Mission

Titan is the largest moon of Saturn and is remarkable for its dense atmosphere that has an estimated composition of 95% nitrogen, 3% methane, and 2% argon. The surface pressure on Titan is about 1.5 bars, which is 1.5 times the surface pressure on Earth. The thick atmosphere and the methane haze make surface observation difficult; however, near infrared observations and pictures from the Huygens probe suggest that interesting terrain is present and is made of solid rock and frozen water

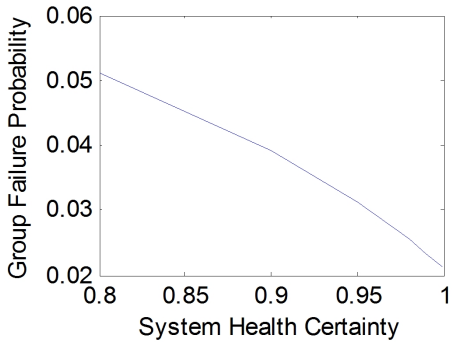

Figure 6.11: Group failure probability vs. SH estimation certainty

ice littered with liquid methane and ethane bodies. Cryovolcanism has been conjectured, as has a methane and ethane cycle like the water cycle present on Earth [82].

A proposed mission to Titan consists of a satellite of Titan that would release an aerobot probe to the lower atmosphere. This lighter-than-air vehicle would use wind currents to explore the moon by taking advantage of Titan's unique atmosphere. The probe would have the capability to fly to points while simultaneously mapping Titan's surface; it would also be able to stationkeep. In addition to wind profiling, surface and atmospheric observations, and atmospheric composition testing, the probe would also have the capability to collect samples from the surface without landing [1].

Because the Saturn system is far away from Earth, there is a significant light-time delay of about 2.6 hours round-trip [82]. This means long communication latencies between the aerobot and Earth. Having an autonomous vehicle that can execute a relatively long mission plan without human interference is important. This autonomous control system must be able to function without human intervention and be able to identify and handle many types of faults and failures in a safe manner. The verification of the fault-tolerant control plans against sets of unsafe conditions will be extremely important and useful for a Titan exploration mission.

6.3.1 Problem Statement

A simplified model of the Titan aerobot was used as an example for the conversion and verification procedure. The aerobot used in this example has a mission to fly to a specific area while maintaining at least 10% power, position awareness, and appropriate safe altitudes, and while being aware of spontaneous science observation opportunities. The example aerobot has several sensors, including two cameras, a laser range finder (LRF), a radar, a hygrometer, and a motion sensor. The cameras and laser range finder allow the aerobot to localize and map the surface of Titan while maintaining

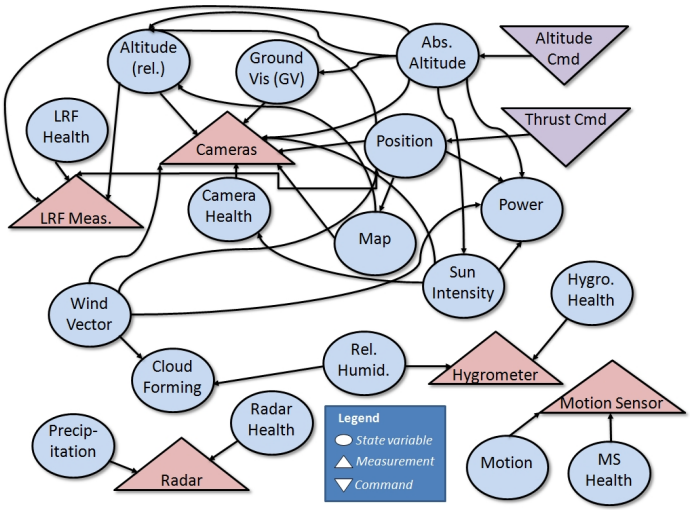

Figure 6.12: State effects diagram for the aerobot example

a safe altitude above Titan's surface features. The radar, hygrometer, and motion sensor are used to detect spontaneous science events such as cloud formation, precipitation, and cryovolcanism. Figure 6.12 gives the state effects diagram for this example problem. The state effects diagram lists all pertinent state variables, commands, and measurements that are used to control the system. The arrows between the different bodies in the diagram indicate that the originating body has a modeled effect on the accepting body.

Most of the models between state variables or between state variables and measurements or commands are fairly obvious. The aerobot is able to localize using the existing map, which is generated by the orbiting satellite and to which details are added by the aerobot. The map uncertainty is a measure of the scale of the surface feature information in an area; the uncertainty is high in areas only covered by satellite images and is low in areas imaged by the aerobot. The position uncertainty, then, is a measure of how well the aerobot can constrain its position relative to the existing map. Sunlight intensity and ground visibility are affected by the aerobot's absolute altitude, and these state variables affect the quality of the measurements taken by the cameras. Relative altitude is the height that the aerobot is above the ground and is the state that the LRF is measuring. The aerobot's position is controlled by the thrust and altitude commands and affected by the wind vector. Power is also affected by the wind vector because more or less control effort may be needed to drive the aerobot based on the direction and magnitude of the wind. The aerobot is assumed to have some regenerative power capability based on solar energy, so sunlight intensity also affects the percentage of power remaining on the probe. (However, with Titan located so far from the sun, solar energy

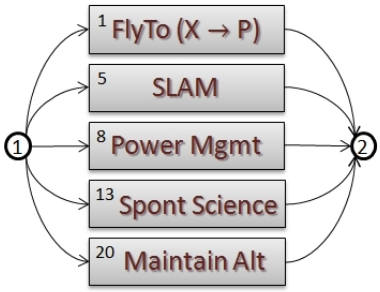

Figure 6.13: Goal network for Titan example

could at best be a back-up power system for an actual mission.) It is also assumed that altitude affects sunlight intensity, with more intensity near the top of the atmosphere.

A derived state variable is a non-physical state variable that depends only on other state variables. The `SpontaneousScience` state variable with four states (`NONE`, `MOTION`, `PRECIP`, `CF`) is a derived state variable that depends on seven state variables (`Motion`, `Precipitation`, `CloudForming`, `RelativeHumidity` and the accompanying sensor health state variables). The `SpontaneousScience` state variable prefers the rarer events; if motion is sensed, then that is the value of the state variable regardless of the presence of precipitation or cloud formation. The next preferred event is precipitation followed by cloud forming.

6.3.2 Goal Networks

The goal network for this example problem, shown in Figure 6.13, is based on the control of the position and altitude of the aerobot as it flies to a specified point. The `Position` state variable (`X`) is controlled via three modes: a "fly to" mode where the aerobot heads towards the constrained area; a "stationkeeping" mode where the aerobot maintains its current position; and a "float" mode where the aerobot drifts without controlling its position. There are also five control modes for the `Altitude` state variable (`Z`) that refer to different absolute altitudes; from lowest to highest, these altitudes are ground observation, detailed mapping, minimum en route, maximum terrain clearance, and service ceiling. The abbreviations used for the passive state variables can be found in Table 6.2.

The first concurrently executed goal tree, shown in Figure 6.14, involves the task of flying to a specified area. There are two tactics available for doing this that are chosen based on the relative wind vector. When the wind vector is favorable or small, the aerobot attempts to maintain a minimum velocity in the direction of the specified area. When the wind vector is large and unfavorable,

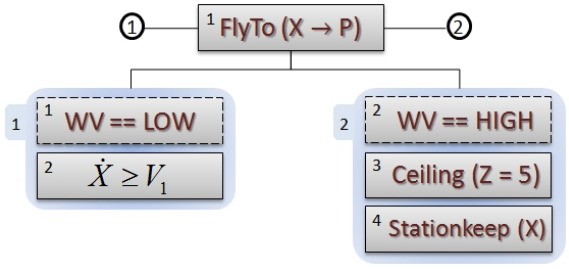

Figure 6.14: Goal tree for flying to a specified area

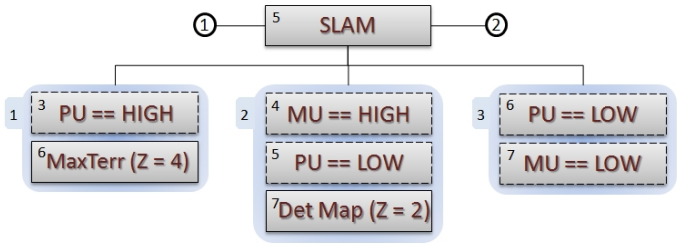

Figure 6.15: Goal tree for simultaneous localization and mapping

the aerobot instead attempts to stationkeep; in a more complex example, the aerobot would profile the wind to find a new altitude at which to fly.

How well the aerobot can constrain its position on the existing map contributes to the position uncertainty; when the uncertainty is high, the aerobot must ensure that it is at a safe altitude to avoid controlled flight into terrain. The second goal tree, shown in Figure 6.15, gives tactics that help to accomplish the task of simultaneous localization and mapping (SLAM). When position uncertainty is high, the aerobot ascends so as to clear all possible obstacles and to match its position with the less detailed satellite map. Execution continues as usual when the position and map uncertainty are low. If the position uncertainty is low and the map uncertainty is high, the aerobot flies at a lower altitude to achieve more detailed mapping.

The third task for the aerobot is power management, which is controlled in the goal tree shown in Figure 6.16. Overall, the aerobot must maintain at least 10% power; if it does not, the aerobot safes to floating until the power increases. While the power value is above 10%, there are several tactics to ensure that the power level does not drop to the safing level. When the power drops below 50%, the aerobot climbs to increase the sunlight intensity that it is receiving. If the power dips below 30%, the aerobot discontinues its trek to the specified point and instead stationkeeps to preserve power.

Spontaneous science observation is an important part of the Titan aerobot's mission, so the goal

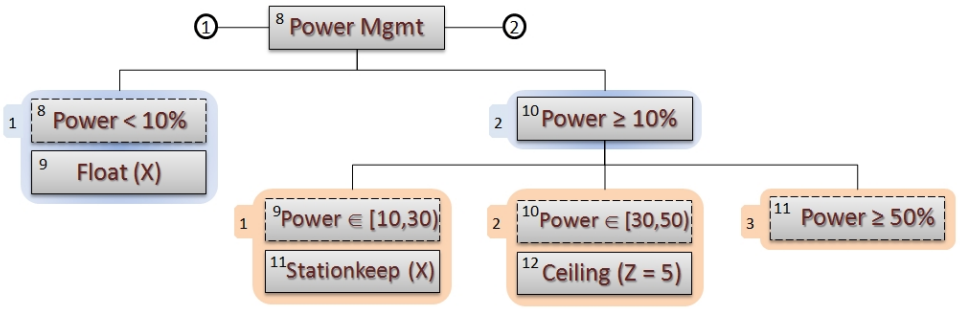

Figure 6.16: Goal tree for power management

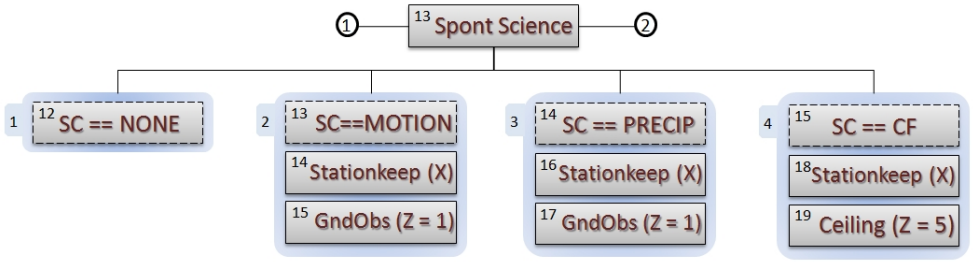

Figure 6.17: Goal tree for observing spontaneous science

tree shown in Figure 6.17 deals with this task. When no motion, precipitation, or cloud formation is detected, the aerobot continues on with its current task. However, if motion on the surface (such as cryovolcanism) or precipitation is detected, the aerobot descends and stationkeeps to observe it. Likewise, if cloud formation is detected, the aerobot ascends to observe it.

Several other factors also affect the altitude at which the aerobot flies, such as the health of the position sensors (the cameras and the LRF), the ground visibility, and sunlight intensity. These conditions make up the five tactics of the final goal tree controlling the altitude of the aerobot; this is shown in Figure 6.18.

Each of the goal trees presented are executed concurrently in the aerobot's goal network. It is assumed that when the aerobot has positive control (i.e., the position and map uncertainties are low, the wind vector is low, etc.), the low-level position controllers successfully avoid terrain at the low altitudes. When two goals constraining altitude are merged (executed concurrently), the higher altitude is taken; likewise, when two goals constraining position are merged, float constraints take priority, and then stationkeeping constraints are preferred over fly to constraints.

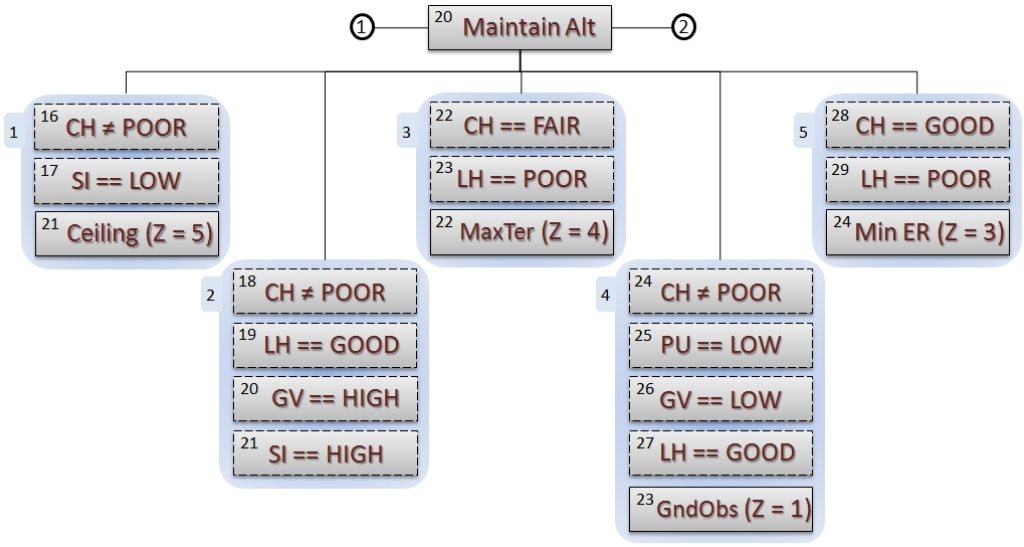

Figure 6.18: Goal tree for controlling the altitude of the aerobot

6.3.3 Verification

The control system for the Titan aerobot mission designed in Section 6.3.2 was converted to a hybrid automaton with 544 locations and thousands of transitions (using a restrictive simplifying assumption that only a single passive state can change per time step) and an input file to the PHAVer symbolic model checker was automatically generated. The unsafe set for the goal network, $Z = \{\zeta_1, \zeta_2\}$, had two sets of constraints:

1. Power is less than 10%, $\zeta_1 = \{(\texttt{Power}, <, 10)\}$.
2. The altitude is lower than maximum terrain clearance while the ground visibility is low and the position uncertainty is high, $\zeta_2 = \{(\texttt{z}, <, 4), (\texttt{GV}, ==, \texttt{LOW}), (\texttt{PU}, ==, \texttt{HIGH})\}$.

PHAVer was not able to handle the automaton along with the nine passive state variable model automata due to the large state space that results (over 2.5 million discrete states). The list of passive state variables and the number of discrete states in the model of each are given in Table 6.2.

To handle the large verification effort, the method introduced in Chapter 4 was used. The first step of that procedure is to ensure that the goal network has state-based transitions. Each root goal and its goal tree were run through the SBT Checker and the software found that the altitude controlling goal tree was missing a tactic with passive constraints as follows:

$$\texttt{LH} == \texttt{GOOD} \wedge \texttt{SI} == \texttt{HIGH} \wedge \texttt{GV} == \texttt{LOW} \wedge \texttt{PU} == \texttt{HIGH} \wedge \texttt{CH} \neq \texttt{POOR}.$$

Table 6.2: Passively Constrained State Variables

State Variable	Abbreviation	Number of States	Estimator Accuracy
Camera Health	CH	3	0.95
LRF Health	LH	2	0.95
Sun Intensity	SI	2	0.99
Ground Visibility	GV	2	0.99
Wind Vector	WV	2	0.99
Position Uncertainty	PU	2	0.9
Map Uncertainty	MU	2	0.9
Spontaneous Science	SC	4	0.8
Power		6	

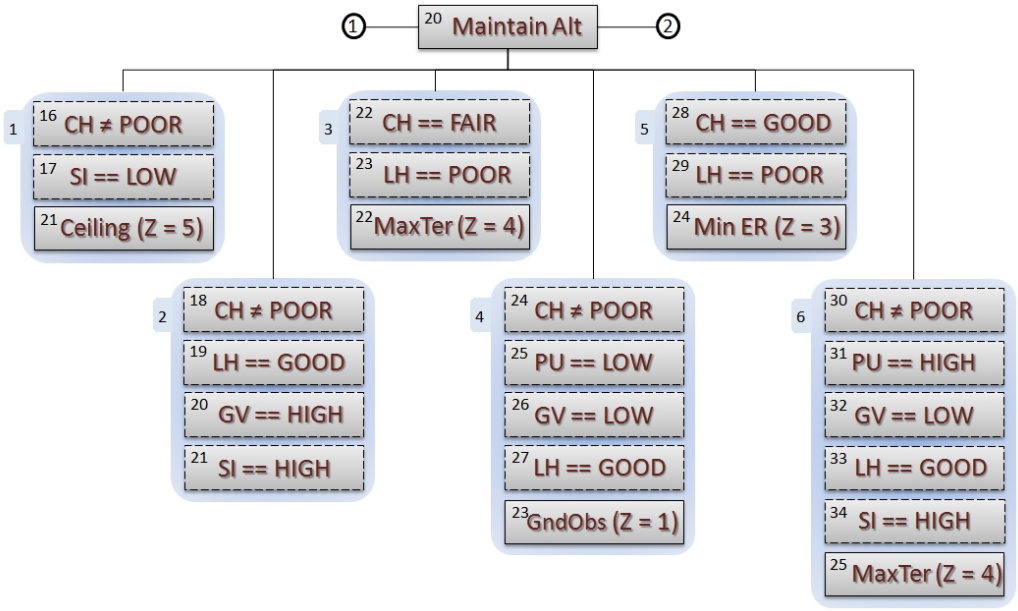

Figure 6.19: Redesigned altitude control goal tree

The unsafe set, ζ_2, dictates that the altitude must be constrained in this tactic to be either the maximum terrain clearance altitude or the service ceiling ($\mathtt{z} = 4$ or $\mathtt{z} = 5$, respectively). Since the sun intensity is high, the maximum terrain clearance altitude is the appropriate constraint. The new tactic in the redesigned altitude control goal tree is shown in Figure 6.19.

The goal network was then verified using the InVeriant software. More than 600 locations were generated and no inconsistent controlled constraints were found. The verifier composed the converted automaton with the `Power` state variable's model automaton for the first unsafe set, ζ_1. The verifier found locations in which the unsafe power constraint was possible when $10 \leq \mathtt{Power} < 30 \wedge \mathtt{WV} == \mathtt{HIGH} \wedge \mathtt{SI} == \mathtt{LOW}$. In order to most efficiently use the software, the unsafe power constraint, ζ_1, was converted into two equivalent constraints: $10 \leq \mathtt{Power} < 30 \wedge \frac{d}{dt}(\mathtt{Power}) > 2$.

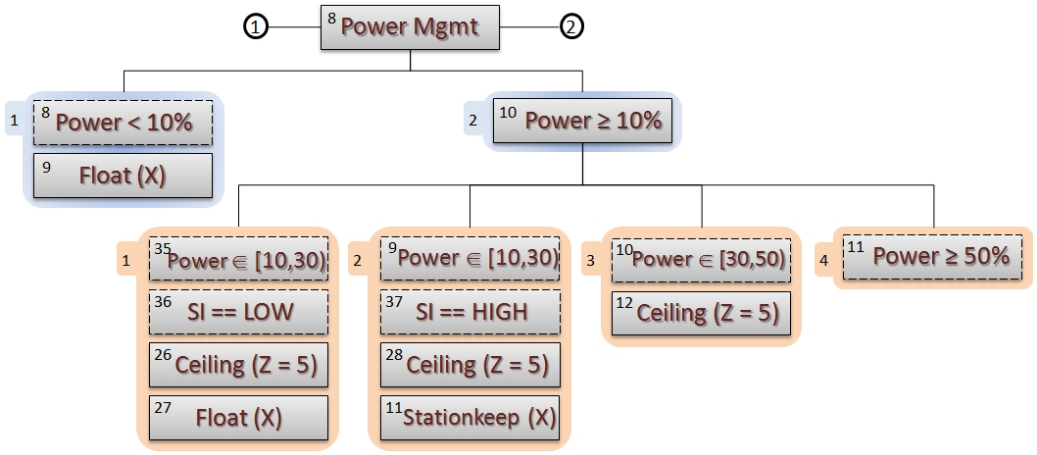

Figure 6.20: Redesigned power management goal tree

The constraint on the power rate is equivalent to a medium or high power use state; if these power rates are present in a location where the power falls into the given constraint, it is possible to achieve a power state that is less than 10%. Because the `Power` state variable is a continuous, rate-driven dependent state variable, a path from the initial condition ($\texttt{Power} = 100$) to the unsafe condition ($\texttt{Power} < 30$) must be found to prove that the unsafe locations are reachable. The software was able to find a path of three locations whose invariants included the appropriate power constraints ($\texttt{Power} \geq 50$, $30 \leq \texttt{Power} < 50$, and $10 \leq \texttt{Power} < 30$) and whose power rates were negative, proving that the unsafe conditions were reachable. The software also output the goals that were common to all the unsafe locations, which triggered the redesign of the power management goal tree to include a tactic that commands the aerobot to float at the service ceiling when the `Power` is less than 30% and the `SunIntensity` is `LOW`. This new power management goal tree is shown in Figure 6.20. Verification of the redesigned goal network confirmed its correctness.

The SBT checker and InVeriant verification software package is superior to the conversion software and PHAVer for this application. The conversion software was able to handle the large goal network to linear hybrid automaton conversion with the transition restriction, taking just under five hours on a 2.0 GHz Intel Core 2 Duo CPU with 4.0 GB of RAM. However, many verification attempts using PHAVer proved to be unfruitful. While this does not show that PHAVer could not complete the task, abstraction, model reduction, and overapproximation would be necessary. The SBT Checker, however, took nearly no time to run once the appropriate data were entered and the output of that software was useful to the design process, unlike the output of the conversion procedure. Then, the InVeriant software was able to convert the goal network into locations and

invariants in about fifteen minutes, followed by about two minutes of verification work. Whereas PHAVer outputs conditions on the state variables that allow at least one of the unsafe conditions to be true, InVeriant gives that information along with the unsafe constraint satisfied and the goals and tactics that are responsible for the failure.

6.3.4 Uncertainty Analysis

The uncertainty analysis was completed for the Titan aerobot mission. The large number of potentially uncertain state variables caused an explosion in the number of complete system states. Assuming that all state variables in Table 6.2 except Power are uncertain, the number of complete system states was almost 600,000. Only the second unsafe set condition was analyzed for simplicity.

The failure probability of the second unsafe set condition was calculated using the automatic complete system state sorting software. The fourth column of Table 6.2 gives the probability that the state variables' estimators are correct. Stationary Markov chains with no mixing time were assumed for the state propagation models for each of the uncertain state variables. After about 65 hours of computation on the computer described above, a stationary unsafe probability of $p_u = 0.00915 \pm 0.005$ was found. Since the problem can be estimated as a uniform completion problem, the failure probability of the goal network with respect to the unsafe condition chosen can be calculated as a function of the number of time steps the goal network executes:

$$W_s = p_u \times \sum_{x=0}^{c_k} p_n^x, \tag{6.2}$$

where $p_n = 0.99085 \pm 0.005$ is the stationary nominal probability. Given the estimator uncertainties, these failure probabilities are fairly low, which suggests a well-designed system. The failure probability as a function of completion time is shown in Figure 6.21; the failure probability approaches one as the completion time goes to infinity.

6.4 Conclusion

The example goal networks presented in this chapter were very useful in driving the design and improvement of the verification methods introduced in the previous chapters. The rover example in Section 6.2 validated the conversion software and pushed the capabilities of the failure probability calculation procedure. The Titan aerobot example goal network was significantly larger and more

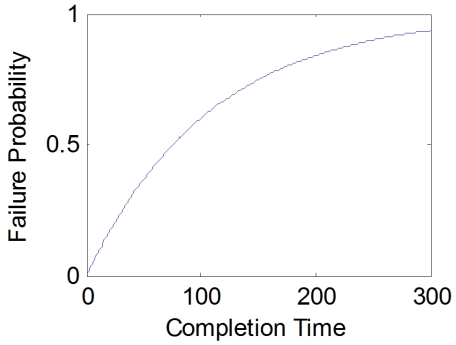

Figure 6.21: Failure probability of the Titan example as a function of completion time

complex and it drove the creation of the SBT Checker and InVeriant verification method due to the inability of the conversion procedure and PHAVer to verify the goal network. Both examples illustrate the techniques in this dissertation well.