Automatic Satellite Telemetry Analysis for SSA using Artificial Intelligence Techniques

Richard Stottler
Stottler Henke Associates, Inc., San Mateo, CA 94402
Jay Mao
Stottler Henke Associates, Inc., San Mateo, CA 94402

ABSTRACT

In April 2016, General Hyten, commander of Air Force Space Command, announced the Space Enterprise Vision (SEV) (http://www.af.mil/News/Article-Display/Article/719941/hyten-announces-space-enterprise-vision/). The SEV addresses increasing threats to space-related systems. The vision includes an integrated approach across all mission areas (communications, positioning, navigation and timing, missile warning, and weather data) and emphasizes improved access to data across the entire enterprise and the ability to protect space-related assets and capabilities. "The future space enterprise will maintain our nation's ability to deliver critical space effects throughout all phases of conflict," Hyten said. Satellite telemetry is going to become available to a new audience. While that telemetry information should be valuable for achieving Space Situational Awareness (SSA), these new satellite telemetry data consumers will not know how to utilize it.

We were tasked[1] with applying AI techniques to build an infrastructure to process satellite telemetry into higher abstraction level symbolic space situational awareness and to initially populate that infrastructure with useful data analysis methods. We are working with two organizations, Montana State University (MSU) and the Air Force Academy, both of whom control satellites and therefore currently analyze satellite telemetry to assess the health and circumstances of their satellites. The design which has resulted from our knowledge elicitation and cognitive task analysis is a hybrid approach which combines symbolic processing techniques of Case-Based Reasoning (CBR) and Behavior Transition Networks (BTNs) with current Machine Learning approaches. BTNs are used to represent the process and associated formulas to check telemetry values against anticipated problems and issues. CBR is used to represent and retrieve BTNs that represent an investigative process that should be applied to the telemetry in certain circumstances. Machine Learning is used to learn normal patterns of telemetry, learn pre-mission simulated telemetry patterns that represent known problems, and detect both pre-trained known and unknown abnormalities in real-time. The operational system is currently being implemented and applied to real satellite telemetry data. This paper presents the design, examples, and results of the first version as well as planned future work.

1. INTRODUCTION

There are several benefits to processing satellite telemetry automatically. Perhaps most important is to quickly determine if there is a problem with the spacecraft that needs to be corrected. Automated processes running on the ground can more quickly determine this than humans examining the same telemetry. More dramatically, automated processes running onboard the spacecraft can automatically examine sensor data onboard the spacecraft that may not be transmitted to the ground for some time or may never be. This processing can also provide a high-level symbolic description of the spacecraft's state for use by higher-level decision-makers, as compared to the thousands of primarily numeric data that the raw telemetry presents. For example, several detailed quantitative telemetry points may indicate that a single event upset (SEU) or latchup has occurred in a microelectronic component. Specific types of SEUs can cause high power draws and associated rapid heating, so cycling the power quickly can prevent thermal damage. The high-level spacecraft state is that one of its components suffered an SEU, it is quickly recovering from it without damage, and this type of event is caused by cosmic rays or space weather, i.e., is caused by natural phenomena. Contrast this example to one where several detailed quantitative telemetry points indicate a rapid surge in power output from one or a few Photovoltaic (PV) cells followed quickly by their total failure. The spacecraft state is that PV cells have been recently destroyed permanently, implying reduced power-generating capability, and that the cause may be lasing, i.e., a manmade phenomena. Additional benefits are the manpower savings from having the telemetry automatically analyzed and having that analysis occur, even when the human operators are not

[1] This material is based upon work supported by the United States Air Force under Contract No. FA9453-17-C-0418

present (e.g., outside of normal hours for the satellite operators). It is also possible to extend the analysis BTNs to take needed actions such as sending reconfiguration commands to the spacecraft, for true lights-out operations.

We are taking a hybrid approach to this application, to use knowledge of what is easily known about the spacecraft as predefined behaviors to confirm expected relationships in telemetry values, but also to use Machine Learning to learn normal and abnormal behaviors of the satellite, to cover any gaps in the encoded knowledge, i.e., make easy use of data that is readily available—telemetry from simulated scenarios and in-space operations.

The Traditional Artificial Intelligence Community has historically promoted declarative knowledge. For applications such as this, that would involve representing the schematic knowledge of the satellite's subsystems as a model and using Model-Based Reasoning (MBR) techniques to process the telemetry against this model. There are many advantages to this approach (and we, ourselves, are applying it in another satellite application) but one disadvantage is that a fairly complete and accurate model of at least one subsystem must be developed and debugged before any useful results can be attained. Meanwhile the traditional software community tends to write software that simply executes a series of software language statements, i.e., follows a prescribed sequence of actions that are not readily intelligible to non-programmers. The Behavior Transition Networks proposed here represent a compromise between the two. The BTNs also prescribe actions but are readily understandable by non-programmers due to their graphic flow-chart-like representations, which also graphically describe the logic flow in a declarative knowledge format. We have found they are easily editable by non-programmers such as the satellite operators or designers.

2. OVERVIEW

As mentioned above, the telemetry processing application described here takes as input a stream of satellite telemetry and produces as output a high-level description of the state of the satellite along with an accompanying detailed analysis (e.g., specific subsystem or components behaving abnormally, the type of problem being experienced, and the most likely possible causes). Two different parallel processes come into play. The first is that, in advance, BTNs are created that define the desired logic to examine specific data from the telemetry stream and examine its relationship with other data in that stream. Specific BTNs may be generally applicable to any situation (e.g., that the total sensed current into a node should equal the total sensed current out of a node, or otherwise something is wrong), or may be specific to particular circumstances (such as the spacecraft being in a specific state or mode, or one or more of its subsystems being in a specific state or mode, whether specific equipment is operating or not (especially if it requires a large amount of power), the spacecraft's attitude and which sides are illuminated by the sun, external or internal warnings or concerns), or may exist to investigate a specific possibly abnormal situation. Having BTNs be specific to a certain type of situation or investigation tends to make them simpler and easier to understand, since many special cases that won't happen in the specific situation do not have to be addressed. The circumstances that affect a BTN's applicability may partly depend on information generated by earlier executing BTNs. And when appropriate, a specific BTN can be easily initiated from another (e.g., One BTN may determine that an anomaly exists in one part of the power generation system and immediately invokes the BTN which investigates that type of anomaly (through examining other telemetry values)).

A circumstances-specific BTN will include information about its applicability, i.e., the situations in which it should be applied. Case-Based Reasoning (CBR) techniques are used to determine which BTNs are applicable at any given time and executed. During real-time execution, as the satellite telemetry becomes available, all of the generally applicable BTNs are executed to examine it as well as the specifically applicable ones, as determined by CBR.

The second parallel process also begins in advance of the real-time use. Machine Learning techniques are used to learn relationships between telemetry values during normal and anticipated abnormal situations. During real-time execution, the satellite telemetry is processed by the ML-derived models to confirm normal operations, categorize known abnormal operations, or determine that an unknown abnormal situation is occurring.

Using this hybrid approach makes best use of ALL available information, both knowledge of how to examine telemetry that the satellite designers and operators are aware of *and* the telemetry data readily available from existing simulations and actual on-orbit operations.

3. BEHAVIOR TRANSITION NETWORKS

Behavior Transition Networks (BTNs) have proven themselves adept at real-time tactical decision-making and investigative processes in a variety of domains, and many of our real-time high-level reasoning systems make use of them since they execute extremely rapidly and readily and naturally represent and mimic high-level human reasoning processes (in dynamic and time-constrained environments). A BTN is a kind of finite state machine (hence its computational efficiency) where states are behaviors and state transitions represent the conditions under which an agent will stop executing one behavior and start executing another behavior. For this effort, BTNs are primarily used to represent the investigative procedures followed to process telemetry in order to detect and characterize spacecraft anomalies in different types of situations. BTNs are declarative and graphical for easy understanding and are easy for non-programmer Subject Matter Experts (SMEs) to understand and edit themselves (as shown in Fig. 1), but are also immediately, efficiently executable.

BTNs are typically hierarchical and made up of several components. Basic, atomic behaviors (those that cannot be described (or decomposed) in terms of other behaviors) are called actions. Unlike state machines, BTNs provide the following constructs: they (i) are hierarchical, (ii) have variables (local memory), (iii) have access to blackboards (shared memory), (iv) are polymorphic, and (v) can execute arbitrary planning or action-oriented code. We use the Open Source SimBionic technology to create and execute these BTNs. SimBionic's graphical BTN representation lets SMEs define problem-solving behaviors by "drawing" them, producing a flowchart-like graphical description. Specifically, problem-solving actions are represented as rectangles, behaviors as boldfaced rectangles, conditions as ovals, and transition connectors as lines. SMEs can attach as many variable assignments, complex expressions, and explanatory comments as they like to any of these elements.

BTNs are used to create intelligent behaviors by dividing the behavior hierarchically into tasks connected with transitions. The current task executes until one of its outgoing transitions becomes true, then transitions to the task indicated by the true transition's arrow. Tasks with a heavy outline are themselves BTNs that are further expanded. Different Tasks and BTNs can communicate with each other using a variety of means (blackboards, parameter passing, etc.). Generally, small groups of BTNs cooperate to fulfill some role, and many execute in parallel.

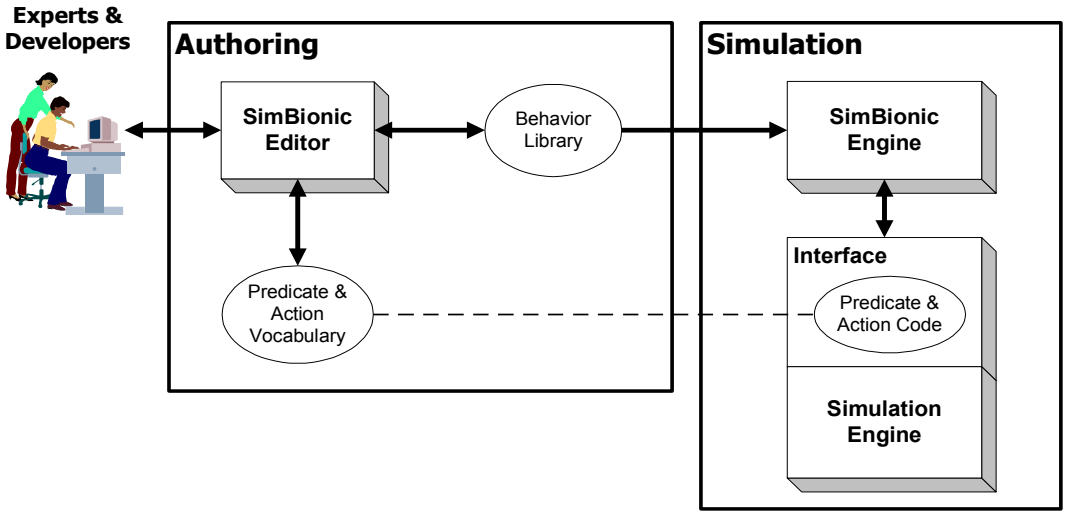

Fig. 1. High-Level SimBionic Architecture

4. CASE-BASED REASONING

When reasoning about a new situation, people extensively employ associations to previous experiences. Case-Based Reasoning (CBR) is the field of artificial intelligence (AI) that deals with the method of solving a current problem by retrieving the solution to a previous similar problem and altering that solution to meet the current needs. Case-Based Reasoning is a knowledge representation and control methodology based upon previous experiences and

patterns of previous experiences. These past solution methods provide expertise for use in new situations or problems. Each case, in the extreme, can represent a different problem-solving methodology and type of reasoning so that each type of problem the CBR system encounters can be theoretically solved in a different way. For this effort, the cases are the circumstances-specific BTNs (which are themselves the telemetry investigative steps appropriate to the specific circumstances).

Stottler Henke is a pioneer in the development and application of CBR. We were among the first to employ separate, explicit, fuzzy definitions of similarity used to calculate a quantitative similarity score between a target case and stored cases in the case base. The features may contain qualitative (symbolic) or quantitative values. Stottler Henke developed rapid retrieval algorithms that could quickly retrieve the most similar case from very large case bases (thousands or millions of cases) [1]. Based on our past CBR experience, we believe that automated satellite telemetry processing is a natural application of CBR.

5. DESIGN

The architecture for the automatic satellite telemetry analysis module is shown below. Analysis is instigated by any of a number of initiating events, which could be as simple as new telemetry being received (e.g., a new satellite communications pass has started), or more complex, such as external indications of problems with the spacecraft or SSA concerns (e.g., communications issues with the spacecraft). As telemetry arrives, it is placed in memory for high-speed access. The events and telemetry are used to derive an initial situation (e.g., assumed normal, in eclipse, specific attitude and sunny/dark sides, specific important equipment on or off, executing or not specific important tasks (e.g., communications or thrusting), in distress, specific obvious subsystem faults, emergency, under attack, etc.).

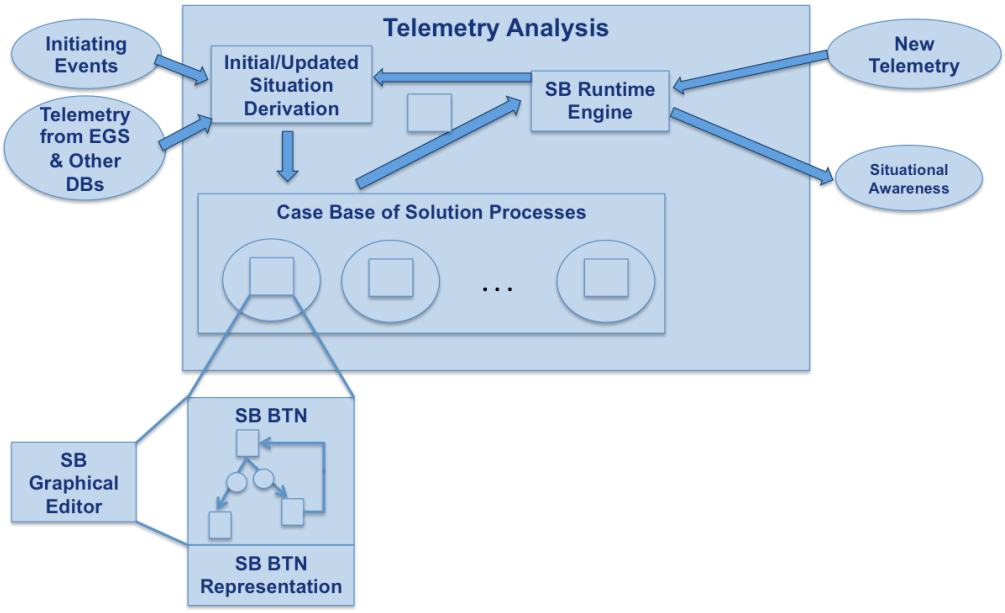

Fig. 2. Automatic Satellite Telemetry Analysis Architecture

Generally applicable BTNs begin executing on the telemetry and, based on the evolving circumstances, appropriate BTNs are also retrieved and begin executing on the telemetry data. Different BTNs process telemetry according to their defined flow chart/FSM-like logic, by accessing specific values of specific mnemonics as described by either their actions (e.g., to perform calculations based on the values) or by the predicates, to compare telemetry values or functions of telemetry values with other telemetry values (e.g., to validate a relationship such as the value of one telemetry mnemonic being between the values of two other specific ones, for example, or being near the average of the other two). Typically, many BTNs execute in parallel.

The actions of the BTNs may create several types of outputs, including outputs to indicate the high-level state, to indicate an intermediate state, to call other BTNs, or to simply complete execution providing no output at all (e.g., a BTN may confirm that an expected relationship holds). Typically, intermediate and high-level states output by the BTN are fed back to update the situation, which may prompt the initiating of other now-applicable BTNs.

A specific BTN may primarily process telemetry values from one instant of time, or it may include past data. BTNs can issue commands to download more and different telemetry and can cue additional BTNs to examine specific future telemetry values. The BTNs may process any of the types of telemetry values. Typical mnemonics include temperature sensors, electrical current and voltage sensors, Photovoltaic (PV) system sensed values, sensors monitoring batteries, the state or mode of equipment, (e.g., a Hall Effect Thruster being on or off is important because it uses a lot of electrical power), momentum wheel sensors, etc.

As mentioned, one of the advantages of the BTN approach is that it is very incremental. Literally, one BTN can be quickly built to check for one specific relationship in the telemetry and it will immediately work. Then a second can be added and so on, incrementally building up the capability with incremental effort. Obviously, the most critical and important and/or the ones that save the most manpower are normally developed first.

Shown below is a simple, general BTN which is used by several others. It simply checks that a telemetry value is within prescribed limits.

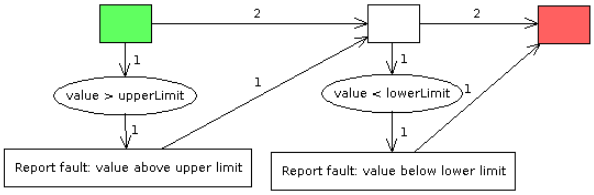

Fig. 3. Simple General Limit-Checking BTN

This simple limit-checking BTN is used to check the overall condition of the 3.3V Power Bus, shown below in Fig. 4, by checking that its voltage, temperature, and current fall within prescribed normal operations limits.

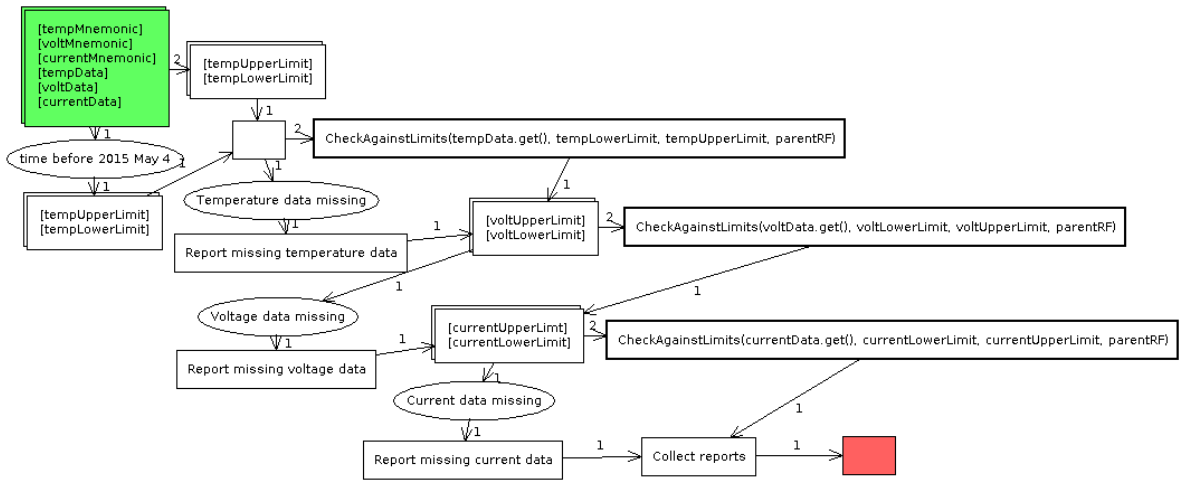

Fig. 4. 3.3V Power Bus Condition Checking BTN

In this particular satellite, the temperature of the Global Positioning System (GPS) and the Communications equipment may rise or fall together, depending on the overall temperature of the spacecraft (which is affected by sunlight and equipment state), and under normal circumstances GPS is slightly warmer, unless a serious problem is

potentially initiating in the Communications system. The BTN shown below in Fig. 5 checks that the relative temperatures are as expected.

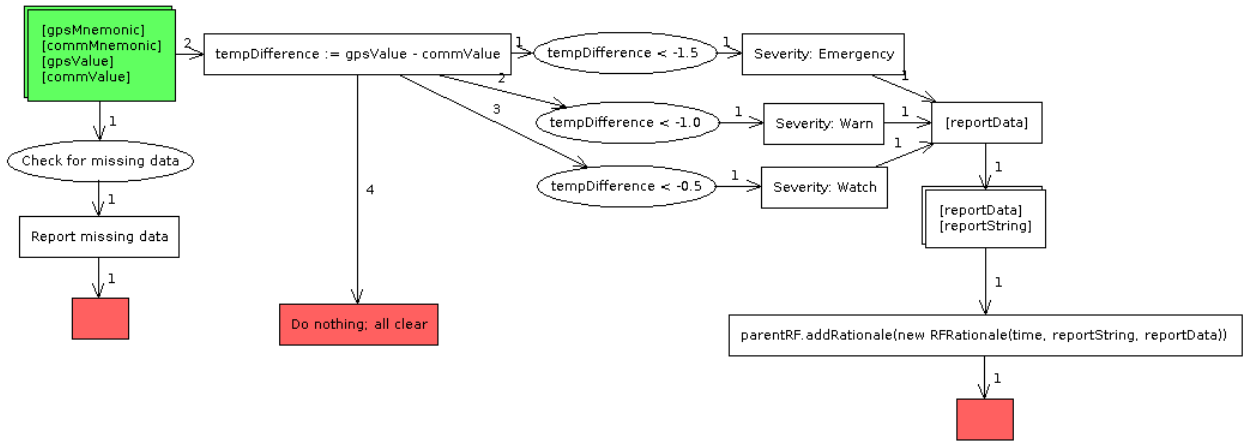

Fig. 5. Relationship Between Different Equipment

The above BTNs all performed checks within one type of sensor, but it is also possible to check for conditions that involve multiple sensor types. Within this spacecraft, the battery charging is indicated by a negative battery current flow. Normally during battery charging operations, the battery voltage sensor zeroes out. The BTN shown below in Fig. 6 checks that the battery and current and voltage sensors are behaving as expected in relationship to each other and this specific expected behavior.

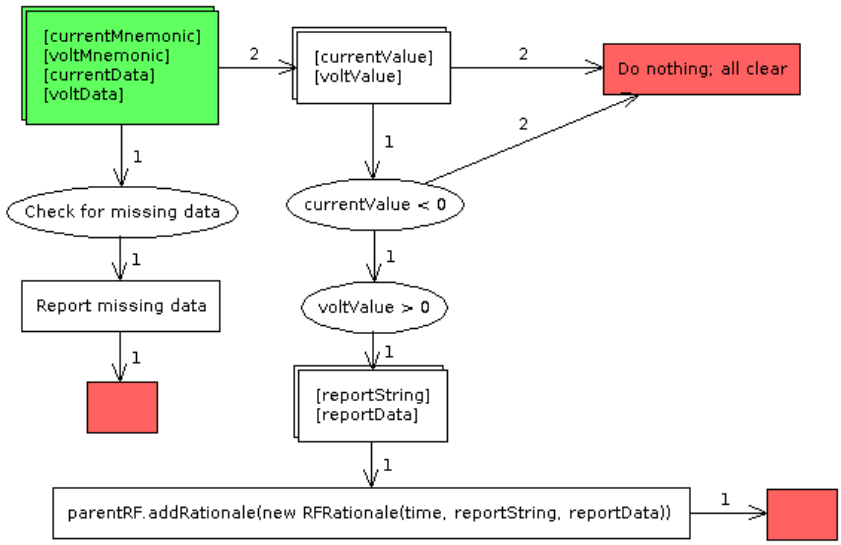

Fig. 6. Different Sensor Types

Finally, shown below, BTNs can be used to check very high-level expected subsystem behavior, including several different components and values from multiple types of sensor. Specifically, all of the power (Power = Voltage x Current) being provided to the Power Busses ought to equal all of the Power Being Consumed by the Power Busses within some reasonable error range. (Otherwise, some sort of parasitic power drain or leakage is occurring.)

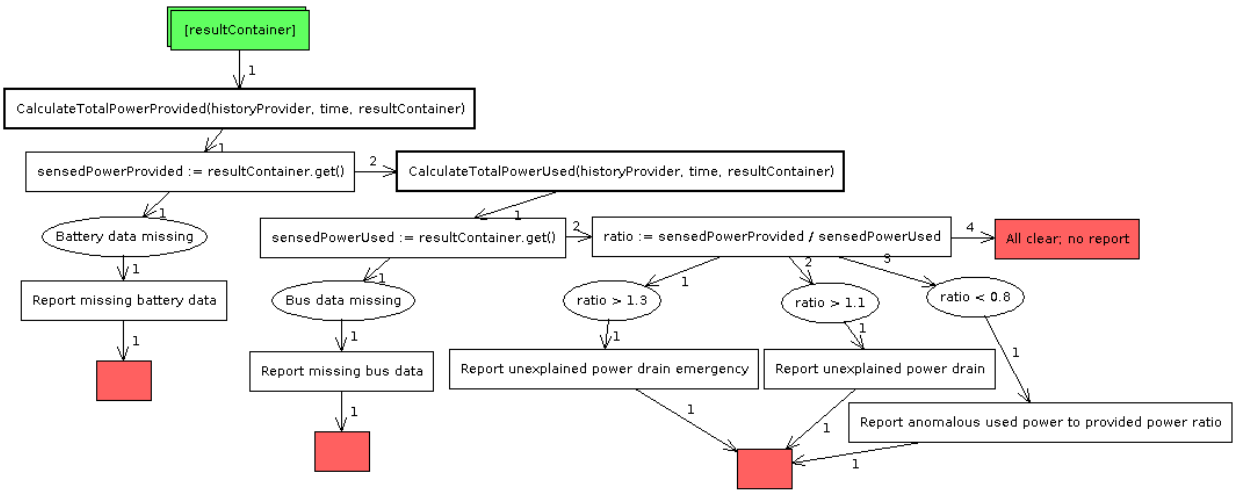

Fig. 7. Total Battery Power Checked Against Total Bus Power Consumed

6. MACHINE LEARNING

Because satellites are expensive to place into space, making sure they are designed and assembled correctly is very important; for this reason, simulators are generally used before launch. Normally there will be two kinds, software-only simulators and hardware in the loop simulators (where a second set of the satellite's hardware components is assembled in the lab). Both types of simulators provide provide a good mechanism for simulating various types of normal and abnormal situations and generating simulated telemetry data for them. These can be used as a ready supply of data for various supervised learning techniques. In the case of cubesats, the number of telemetry mnemonics is small enough that this is generally all that is needed. In the case of larger satellites, which may have thousands of telemetry mnemonics, a mechanism may be needed to determine the most important telemetry mnemonic and associated values that indicate important spacecraft states. For these, an Automatic Fault Recovery Configuration file may provide the most important fault recovery procedures and implicitly indicate what telemetry mnemonics/values are considered important/significant. The simulator can be run to learn normal and abnormal satellite behaviors, associating thousands of telemetry values with these dozens of critical telemetry values.

The vast majority of these relationships exist entirely within one snapshot of data. However, there are some relationships which inherently span across time. These tend to fall into two categories—those that represent short-term behavior and those that represent long-term changes in components. An example of the former is charging and discharging the batteries through electrical loads and PV panels, respectively. The charge in the battery is most closely related to its previous charge (as described by previous telemetry) and the charging and discharging currents that have occurred since then. Another example is warming and cooling (as evidenced by temperature sensor value changes) of different locations in the satellite based on component use within the satellite and external conditions (e.g., the side(s) facing the sun heating up). Similar to the battery case, the temperature at a sensor currently is most closely related to its previous value plus/minus sources of heating/cooling. These short-term behaviors can be learned by including a few previous telemetry values during machine learning (including deep learning) training.

Long-term changes in the components are exemplified by PV Panels and Batteries, both of which degrade over time and use, partly based on sheer age but also partly based on the conditions they experience while in use. Some of these are very predictable and can be simulated in advance. But it will also be important to continue to relearn updated behaviors from telemetry from in-space operations over the entire life cycle of the satellite.

Predicting and identifying faults can be cast as a classification problem that can be modeled using a variety of techniques. Supervised machine learning techniques would be the first line of investigation. These techniques require both positive and negative training examples to produce accurate models. Their success will therefore depend on the availability of such data through the use of the simulator described above. A variety of supervised learning techniques such as artificial Neural Networks (ANN), Support Vector Machines (SVM), and Gradient Boosting Decision Trees (GBDT) are available. The Support Vector Machine (SVM) is a classifier that performs

categorization by constructing hyperplanes in a multidimensional space that separates cases of different class labels. SVM training algorithms build a model that assigns new examples into one category or the other, making it a non-probabilistic binary linear classifier. In addition to performing linear classification, SVMs can efficiently perform a non-linear classification using what is called the kernel trick, implicitly mapping their inputs into high-dimensional feature spaces. We are using this approach to build fault prediction models using both telemetry data snapshots and temporal data. The latter can be modeled as features, along with the snapshot data (e.g., battery charge at times T1, T2, and T3 can be modeled as three separate features).

Deep Learning techniques can also be utilized for this problem [2]. Deep Learning is a recent machine learning approach that is showing considerable promise at effectively addressing some of the most challenging problems in areas such as speech recognition, natural language processing, and computer vision. There are many different architectures and algorithms for deep learning that have been highly successful in addressing challenging artificial intelligence problems. Unsupervised and supervised learning approaches can be combined to develop powerful classifiers. One common approach is to use Denoising Autoencoders in an unsupervised learning mode to construct high value features detectors. Denoising Auto-Encoders is a specific Deep Learning approach where the input data is intentionally corrupted with noise and a neural network is used to recover the original training data. The process of recovering from the corruption leads the network to learn statistical dependencies between the raw input features and leads to the identification of derived, high-value features. These can then be used with a supervised learning approach like SVM to build a classifier. Developing supervised classifiers using these derived features leads to a better model than using raw data alone. Another variation of this approach is to use auto-encoders to predict future values of sensors from current values. This additionally leads to the encoding of temporal features. It is also possible to use a combination of these approaches, which we are investigating. The chief disadvantages of Deep Learning approaches are their computational complexity, and the need for parameter and architectural tuning. It is important to experiment with different network architectures and parameters in order to find high-accuracy models for any given problem. Fortunately, third-party machine learning libraries make it feasible to conduct such experiments rapidly. Packages like WEKA [3] are used for experimenting with supervised learning approaches like SVM. Libraries like TensorFlow and Caffe are available for rapid prototyping of Deep Learning approaches.

As described above, ML is used to complement the BTN approach. Our first experiments worked with SVMs. Since the SVMs are intended to automatically fill in gaps in the knowledge represented by the BTNs, we needed to determine if they could learn relationships in the telemetry that provide indications of different high- and low-level states of the spacecraft. To determine if this was possible, we needed to start by determining if they could learn known relationships and states.

The temperature data on the 3.3V power bus falls into two regimes. The normal operating range for temperatures before 2015 May 4 is between 27 and 34 degrees. The normal range for temperatures after 2015 May 4 is between 1 and 18 degrees. The BTN displayed in Fig. 4 reports points in time with readings outside of these limits as thermal anomalies.

In order to see if known relationships and states could be learned, the same limits were used for assigning classifications to data points used to train and evaluate SVM models. Six categories are used; for each of the two regimes, there are two abnormal states—a category for points where the reported temperature is below the regime lower threshold and a category for points where the reported temperature is above the regime upper threshold, and the normal state where the reported temperature is in the normal range.

From each category, a sample of 100 points was selected. The training data set consisted of 90 points from each category; the other 10 points were used as a test set. Cross-validation was used when building models on the training data set to select hyperparameters. The trained classifier when trained achieves a true positive rate of 93% on the test set; the classifier has an 85% true positive rate for points in the normal categories and 97.5% true positive rate for points in the anomaly categories. The very high accuracy on the 4 abnormal states is important since these represent abnormalities that should be reported and investigated.

The entire dataset consists of 7991 points. 93% of the points are in the normal categories, and the remaining 7% are distributed roughly evenly among the four abnormal categories. The classifier achieves a true positive rate of 83% on the entire dataset; the rate for points in the normal categories is 82%, and the rate for points in the abnormal categories is 97%.

7. FUTURE WORK

We are currently simultaneously working with telemetry from MSU and the Air Force Academy and our incrementally created BTNs for both applications. The most immediate future work is to continue this process to have a more complete set, in order to check for easily known relationships and constraints on telemetry values. We are currently investigating several Machine Learning approaches, and this process will continue with the goal of determining which ones work best and/or can be usefully combined for our two different applications. Once the two applications have proven themselves, we will start to transition them to operational use on the ground as part of the ground control stations at the two locations. There are current plans to place a version of the software onboard a future MSU cubesat that will then be launched into space, demonstrating onboard, in-space, operational use of these techniques.

8. REFERENCES

1. Stottler R.H., Henke, A.L., and King, J.A., Rapid Retrieval Algorithms for Case-Based Reasoning, IJCAI'89 Proceedings of the 11th international joint conference on Artificial intelligence, Vol. 1, 233-237, 1989.
2. Goodfellow, I., Bengio, Y., and Courville, A., *Deep Learning*, MIT Press, Cambridge, Massachusetts, 2016.
3. Hall M. et al., The WEKA Data Mining Software: An Update, ACM SIGKDD Explorations Newsletter, Vol. 11, Issue 1, 10-18, 2009.