Space Data Model Modernization for Proactive and Machine-Assisted Analytics

Alexandra Wright, Peter W. Boettcher, Anye Li, Erin L. Main
Massachusetts Institute of Technology, Lincoln Laboratory

1 ABSTRACT

Modern challenges and objectives for space domain awareness look different than those decades ago when people first began launching satellites. However, we still rely on data models developed for reactive catalog maintenance, whose purpose is to provide a latest orbital update on each satellite. In this paper, we present modernized space data models that redefine time and data representations to enable pro-active and machine-assisted decision making. A flat list of catalog updates is inadequate for this, as it does not represent a history of each satellite's behavior over time, instead offering a history of sensor collections. Furthermore, it does not provide the time constructs needed for representing multiple simultaneous current or future hypotheses, which is important when assessing or anticipating satellite actions that manifest as non-deterministic orbit maneuvers. Inability to represent such realistic satellite behaviors with clear mathematical constructs is a barrier to machine automation of assessment, detection, and prediction of orbital actions.

In our model, we abandon a flat catalog structure in favor of a state tree that constructs a genealogy of a satellite's related, consecutive orbital states and a method for alternately pruning and promoting branches of a satellite's state history that incoming data either refutes or supports, respectively. Branches in the state tree connect via orbital maneuvers, which when confirmed, are promoted to the main branch of the state's history. This same structure also supports owner/operator plans or hypothesized future actions, as they are also naturally expressed as a series of connected orbital states. By accommodating parallel genealogies from alternate sources, this method of describing orbits disambiguates valid start and stop times of orbital estimates, presents a representative history of a satellite's past orbits and actions, allows for analysis of multiple possible future scenarios, and enables machine automation of both labeling and generating orbit behaviors and detecting when data supports specific hypotheses generated.

We built and refined our data model by standing up multi-hypothesis tracking techniques over many years of measurement data on spacecraft and constructing tools for space operators that are fed by the results stored in our data model. This has allowed us to test, break, and improve our data model while creating functionality that demonstrates the instances where our constructs provide key scaffolding for richer data analytics. In addition to the novel predictive automation alluded to, we have found that a great strength of the flexible data model includes the incorporation of non-measured data beyond the labeling of genealogical state branches. Adding other contextual metadata on maneuvers and spacecraft themselves further assists in constructing new methods for previously challenging orbital analysis tasks, such as synthesizing historically observed satellite actions into inferred intent.

This paper also considers architectural impacts of our model if it were to scale to the broader space domain awareness community of military, commercial, and international partners. We propose an implementation of our construct that supports decentralized catalogs of orbital state data to support the challenge of collaboration with a diverse and distributed user base. We frame all of this with how the model we have built addresses core needs of this work's sponsor, the United States Space Force, as well as the broader community of space domain participants.

2 INTRODUCTION

The United States (US) "space catalog" was started after the first satellites were launched, beginning with the Union of Soviet Socialist Republics' (USSR) Sputnik and its rocket body in October of 1957. By 1961, the US Air Force had stood up a network of space surveillance sensors that were tracking over 100 human-made objects in orbit [1, p. 18] and documenting them in a "catalog" of satellites. Modern space surveillance and control systems are still cataloging space objects today, which is useful, but the catalog itself has inherited limitations of the original design. While this design was appropriate and efficient for its job decades ago, today's new space challenges warrant

This material is based upon work supported by the Department of the Air Force under Air Force Contract No. FA8702-15-D-0001. Any opinions, findings, conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the Department of the Air Force.

paradigm changes beyond just data formats in order to be effective. We will consider and demonstrate alternate data constructs for space object management in a contested, congested environment where the US Space Force has new and emerging objectives for safety and freedom of action in space.

3 THE HISTORY

The early days of satellite launches were also the early days of computers. While engineers understood enough about orbital mechanics to put a satellite in orbit and predict how that orbit would evolve, the urgency with which these calculations were required increased once satellites were orbiting the earth at speeds >7km/s. This urgency was not only for command and control of one's own spacecraft, but also for monitoring and tracking satellites others launched. This monitoring job sounds less stressing, but it was of great concern given the cold war space rivalry between the USA and the USSR, both of whom wanted to be the first to put national security technology and humans in space and needed to conduct numerous early satellite tests to learn how to do so safely and reliably.

3.1 ORBIT REPRESENTATION AND DATA FORMATS

Tracking a satellite was done by pointing sensors in the expected place where a satellite would rise over the horizon and, if a sensor found it, collecting observation data on the satellite, fitting an orbit to the data using the laws of physics, and sending the resulting orbit to the next sensor site that could point an antenna in the right place at the right time to collect more data on the satellite. This entire data collection, processing, and communication process had to be done before the satellite flew over the next sensor site, which could be in minutes or hours depending on the orbit of the satellite and the respective position of the sites. Each new sensor that collected data would then be used to update the orbit and the cycle of collection would continue.

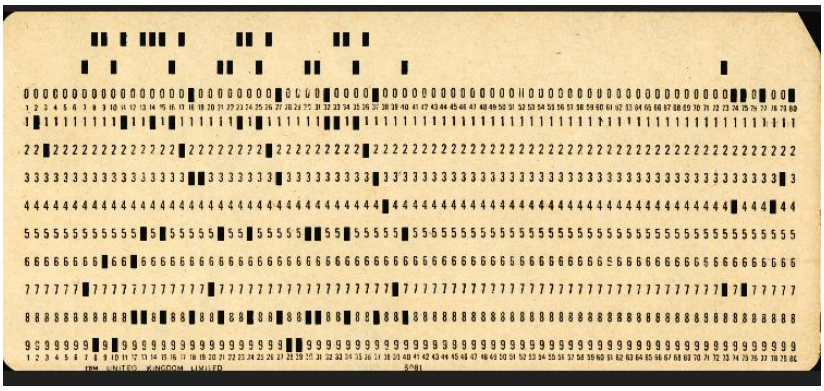

Figure 1: Used punch card, 80 columns wide (© Pete Birkinshaw, license CC BY 2.0)

Satellite tracking began with the launch of Sputnik 1 in 1957, at a time where computers were very large and expensive, and had, by today's standards, very limited processing power and data storage. Magnetic tapes were gradually replacing punch cards as the dominant storage medium, but punch cards would remain a common means of program source code and data entry until the mid-1980s [2]. The Simplified General Perturbations (SGP) family of orbit prediction models, which began development in the 1960s and became operational in the early 1970s, were developed to provide adequate accuracy for the US Air Force Space Surveillance Network within the constraints of the limited computing capacity of the time. The transmission format that was developed for orbit determination software using these models, known as the Two-Line Element Set (TLE) [3], fits into 69 columns of two punch cards. Various orbit determination codes were improved throughout the 1980s and merged by 1990, culminating in the present incarnation of SGP4 as implemented by AFSPC Astrodynamics Standards, the de facto standard propagator of the North American Defense (NORAD) space catalog, with the TLE as the lingua franca for describing satellite orbits. Although advances in computing have since made punch cards obsolete and enabled the development of more accurate and compute-intensive prediction algorithms, SGP4 and TLEs remain common currency as the basis of the space catalog to the present day [4].

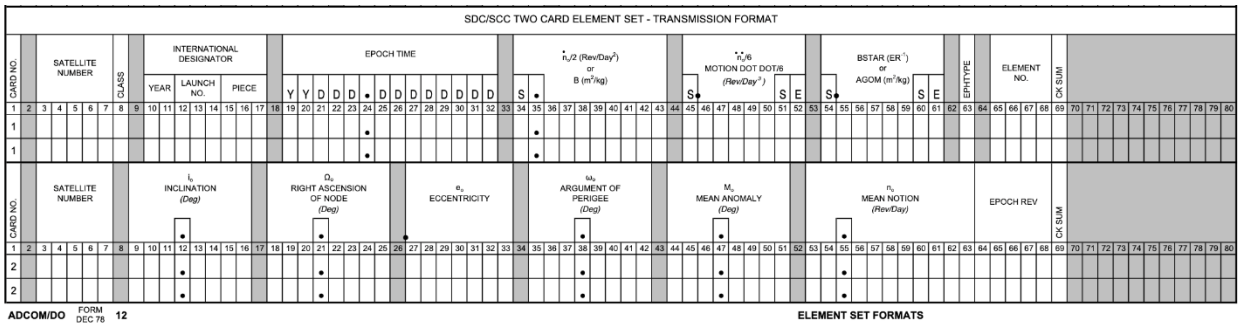

Figure 2: Two-line element set (TLE) column layout, 1978

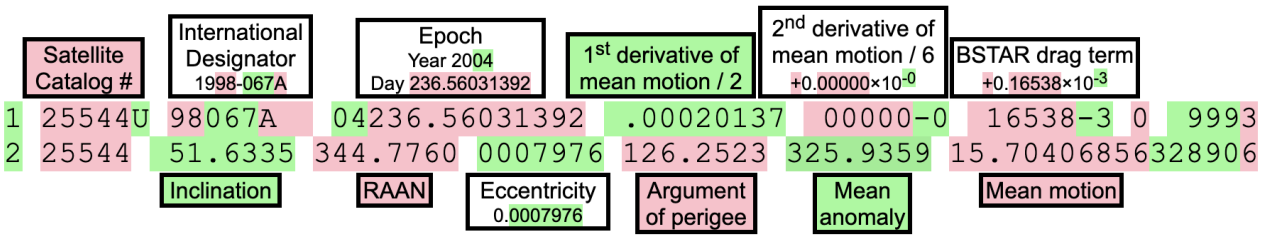

Figure 3: Two-Line Element Set (TLE) for the International Space Station on 09 Dec 2004. Source: space-track.org

SGP4 and TLEs have survived to this day because they remain, for their basic purpose, adequate. The accuracy of SGP4 is considered adequate for tracking the majority of satellites, and, especially for low earth orbit, SGP4 typically runs considerably faster than Special Perturbations (SP) propagators based on numerical integration. The example Two-Line Element Set shown in Fig. 3 is legible to those familiar with the format, as the use of mean Keplerian elements provides basic intuition of the orbit. For example, one can immediately see that the example satellite is in low earth orbit based on its mean motion of 15.7 revolutions per day. However, there are a number of limitations: 5-digit catalog numbers were adequate for the number of satellites at the time but the proliferation of satellite mega-constellations and space debris will soon make them untenable; use of 2-digit years was common practice of the time but the Y2K scare in the years leading up to the turn of the millennium has since shown it to be short-sighted. Other limitations apparent to more technical users are the inability to describe parameter estimation error, and the inability to describe parabolic or hyperbolic orbits, or, more fundamentally, any trajectory not properly modelled by SGP4.

In modern times with modern computers, the idea of omitting two digits in a date string to conserve space and computation is absurd, but removal of computing constraints doesn't motivate abandonment of TLEs given the significant inertia of their use in space subsystems. New orbital data standards have been developed to address 21st century challenges of a long future space horizon, launches to other planets, and the huge number of satellites, some of whom operate close or even dock with each other. These have been or are being adopted across modern space surveillance networks. Even so, a widely unrecognized problem is that the input format itself is merely the surface symptom of vestigial data processing premises that are dissonant with current needs in more fundamental ways. Now that we have covered the background and rationale behind the form of the legacy data, the discussion in the rest of this paper will be about the construction of the space catalog itself, regardless of orbit representation format.

3.2 THE DEFINITION AND PURPOSE OF THE CATALOG

Consider the original purpose of the catalog described above and shown in Fig. 4 below: to enable any sensor anywhere to know when and where to look for any satellite in space.

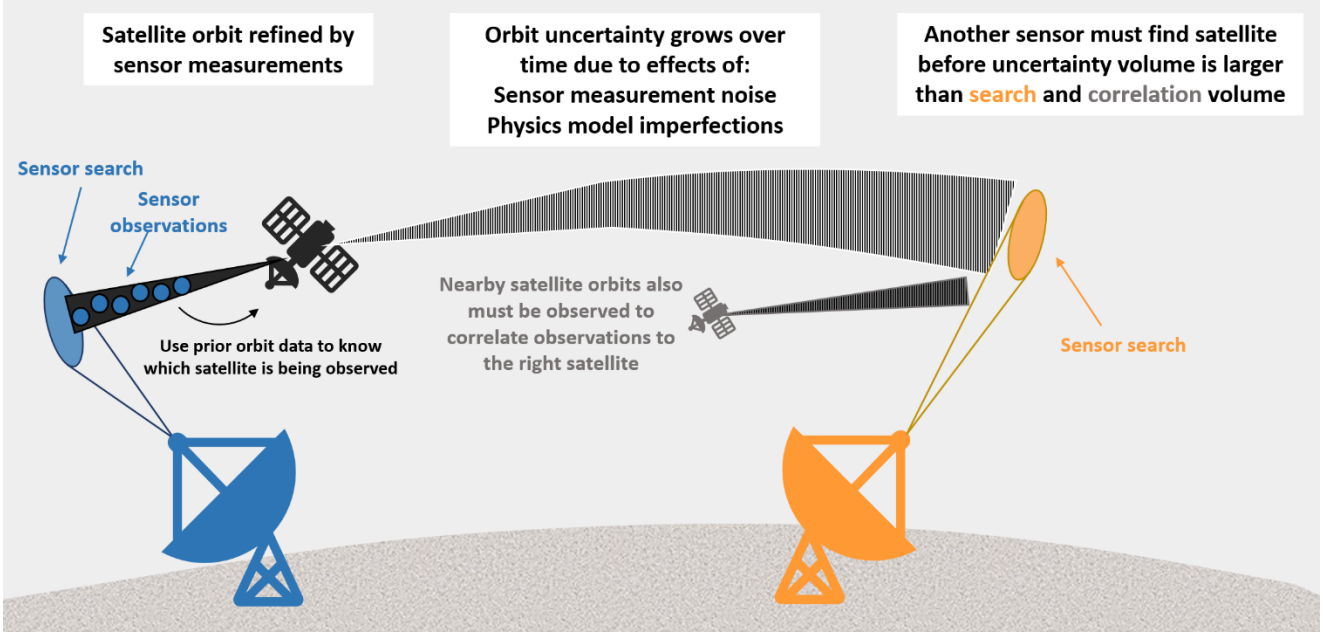

Figure 4: Space Catalog Methodology

The question we propose here is whether a list of latest orbit estimates is the right definition of The Catalog for the more complex tactical awareness and management goals of the space mission in the 21st century. The authors believe it is not. This paper serves to challenge the underlying assumption that a list of latest estimates is appropriate for the mission and to describe our alternate solution and implementation that exhibits significant advantages for today's space awareness and management challenges.

To preface this argument, let us imagine the structure of the current catalog we described above and what updates to a single satellite over time would look like. They would appear as discrete estimations over time, as shown in Fig. 5 where each update for a single satellite is plotted at its epoch time on the x axis. Imagine that the y axis is populated with all the different cataloged satellites: each would have its own flat list of orbital updates over time, dictated by when sensors had observed the objects enough to enable a new parameter fit.

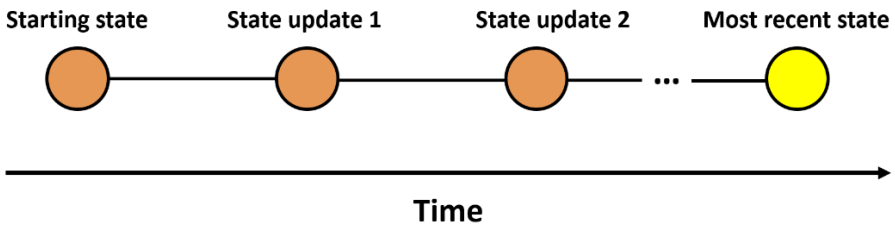

Figure 5: Space catalog construction for an object

This flat list functioned well for early satellites since they did not maneuver on orbit and there were few sensors measuring them. As satellites began to maneuver (and thus changing their orbits), the catalog structure did not change, but anachronisms of this flat list arose.

The first fundamental problem is that, if the satellite maneuvered little enough to still be found by the next sensor, the observations would be taken, used in a differential correction to update the old orbit, and would produce a noisy new orbit that represented neither the pre-maneuver nor post-maneuver orbit well. Over time with more sensor updates, the new orbital parameters could be calculated more correctly, but this would only happen once enough data was collected after the acceleration (maneuver) event. When almost all the observations in the fit span were

after the maneuver, the orbit propagation model, which only includes natural forces, could fit the new orbit well and the resulting catalog update would then be a good description of the new orbit.

This means that, for maneuvering objects, the catalog history would certainly contain orbital updates that were *not* good representations of a satellite's actual behavior. Other reasons orbital updates could be bad include poorly calibrated sensor inputs used in an orbit correction and incorrect assignment of observations on one satellite to a different satellite's orbit. Since the only goal of the catalog was to have the best **latest** updates, errors were considered moot once newer, better estimations were created. Therefore, the solution is to have a team of orbital analysts monitor and correct any catalog problems they find by generating newer, better orbits on problem objects rather than fixing or eliminating orbital updates that are imprecise or incorrect, as older updates essentially get overwritten by newer ones. If orbital estimations are incorrect enough that a satellite cannot be found on a subsequent orbit (or the satellite is new on orbit via a launch), the orbital analyst team can also handle that as part of their job curating the catalog. This is an inadequate solution as the purpose of the catalog changes.

3.3 THE CHANGING PURPOSE OF THE CATALOG

In the early days of satellites, when the USA and the USSR were the only entities capable of mustering the technology to send objects into orbit, spacecraft were mostly experimental. As technology was developed and refined over the years, satellites became more functional, commercial uses became more feasible, and the ability to launch them became more democratized. By the time the USSR collapsed in 1990, commercial uses of satellites had motivated a dramatic and sustained increase in the number of objects launched into orbit [5]. In the present day, both military members and civilians have become dependent on space technology for daily usage from weather forecasts to navigation aids. However, space has become increasingly unsafe: orbits are crowded, rules and treaties are conspicuously absent, and spacefaring nations are competing for use and dominance of space, motivated by military and economic advantage.

In 2020, the US Space Force was created as a distinct service of the Department of the Air Force, which represents a culmination of concerns about present and future space safety given the evolution described above. This signified a clear shift in how the US military viewed its responsibilities in space: it was no longer just to create a catalog and launch satellites for useful purposes, it was to protect and defend the satellites themselves.

This shift in focus makes space more similar to other domains, such as air and maritime, and can be informed by their operations. We understand from these domains that, in order to protect and defend anything, one must detect and anticipate hazards and threats to then assess them, avoid them, or engage them. The definition of *tactical awareness*, whose key tenets are illustrated in Fig. 6, provides a model for the knowledge required to take successful action on threats and hazards. Answering these four questions is essential to determining where, when, how, and what tactics should be used to protect and defend. It is important to note that the questions are inextricably linked: for example, one cannot predict future behavior without a history of past behavior and an idea of what something is.

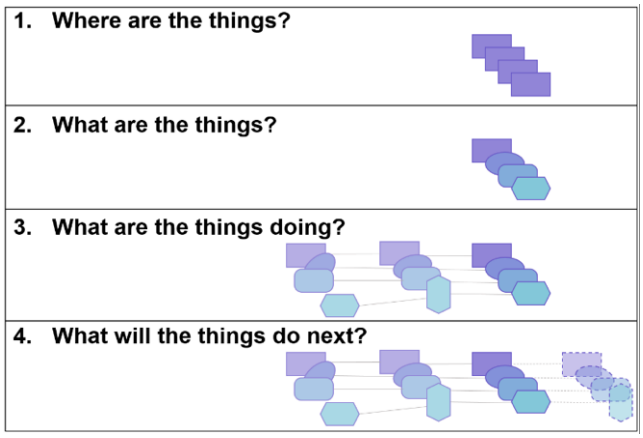

Figure 6: The Tenets of Tactical Awareness

As described in Section 3.2, the legacy space catalog is intended only to address Question 1 (current best estimate of satellite location). Question 2 is addressed only partially, with a list of known objects indexed by NORAD number and COSPAR code, but no capability to handle unknown objects. What about Questions 3 and 4? Understanding observations over time, not just in a single instant, is critical to answering questions about what the object is and what it is doing. Having an idea of what something is doing and can do is critical to making accurate predictions of what it might do next. This concept of understanding over time is where the legacy space catalog concept begins to fall apart.

As discussed previously, the space catalog is a list of latest best guesses; any erroneous or imprecise orbits were ignored, as they were rendered moot by new updates. They are clearly not moot if one wishes to use the catalog as a historic record to understand what a given satellite does over time. In the past, this was not a common or pressing objective, so any deeper questions of this nature could be studied on a case-by-case basis and would require trained analysts to extract a "clean" set of orbital updates to predicate their analysis. However, this would still not be a good representation of the behaviors of a satellite without an additional step: the analyst would need to compute differences between all orbit updates to determine whether the satellite had changed orbit at any time or whether the updates were just a record of measurement imperfections used to refine and confirm a previously computed orbit.

The analyst time to clean up and add on-orbit behavior information to the catalog would be significant, but this was not a concern in the early days because the history of the catalog was not a compelling product. Reporting on satellite behavior was a separate job from cataloging, since it was generally assumed satellites flew according to Kepler and there were not too many exceptions of interest to prohibit manual analysis.

This is no longer true. Not only are there >20,000 satellites in orbit, the challenge is compounded by the modern ubiquity of on-board propulsion. Geosynchronous (GEO) orbits are useful and lucrative, as a satellite in that orbit will see and be seen from a large and consistent portion of the Earth's surface; however, the types of satellites that can reach GEO orbit are energy-intensive, expensive to launch, and are lifetime-limited by the need to maneuver regularly for maintaining "station" over a certain longitude of the earth. Extending the useful life of these satellites was a significant motivator in advancing on-orbit propulsion systems. Today, robust electric propulsion has become commonplace, allowing large sophisticated and small simple satellites alike to include thrusters in their designs. This increases the number of maneuverable payloads even more, which increases the number of bad catalog updates. Even if understanding a satellite's behavior history doesn't at first seem relevant for most benign satellites, today's increased density of space and maneuvers make catalog update correctness relevant for the most basic safety reasons. When fitting maneuvered orbits causes bad catalog states in crowded orbital regimes, this can suggest collisions that aren't real or, even worse, miss ones that are and could be avoided.

4 PROTOTYPING NEW DATA MODELS

We have made the case for three key problems in the current catalog structure.

1. History of satellites is not represented, making it difficult or impossible to answer key tactical awareness questions that come with a military approach to protecting and defending assets in space
2. Solving orbital states while assuming all motion is due to natural forces does not adequately account for the behavior of active payloads, which causes both historical and immediate inaccuracies that impact all safety of spaceflight
3. Not all objects discovered in orbit have a catalog number. Examples include newly launched objects, new debris pieces discovered, sub-satellites ejected or un-docked from larger satellites, satellites that have maneuvered far enough away from their last-seen orbit that they are not immediately identifiable, and orbital maneuvers planned, but not yet instantiated by satellite owner/operators.

We will address solutions to the first two problems next. The proposed solutions will also serve to suggest methods for addressing a variety of uncatalogued space objects.

4.1 PROTOTYPE SOFTWARE SYSTEM

In order to develop the models and abstractions necessary to meet these challenges, the authors have prototyped a satellite analytics software system, shown in Fig. 7. The system is deployed as an operational capability in support of the SDA mission. The remainder of this paper will discuss several components of this system, focusing on the data models and abstractions. An improved **state estimator** is necessary to provide good states to the **state tree** catalog which is the core of the system. Finally, new **user API** constructs are necessary to generalize the access and use of the new data model.

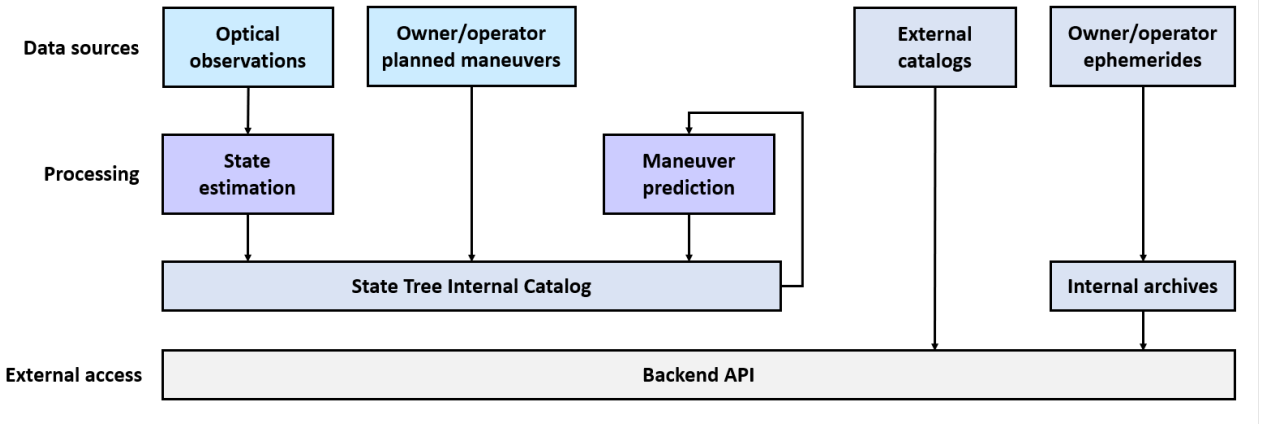

Figure 7: Analytic Software System Diagram

4.2 MANEUVER AWARE STATE ESTIMATION

The first system block to address is the satellite state estimate. General Perturbations (GP) propagators based on the TLE format are clearly not up to the challenge; Special Perturbations (SP) numerical integration approach is required. However, switching propagators alone is not sufficient. For accurate historic analysis, estimated states must specify accurate time bounds, and must account for objects that maneuver.

To correctly estimate the state history of an active satellite, we developed a tracker based on SP that assumes objects will maneuver at any time. It uses each set of incoming observation data to assess whether the measurements suggest that the object is on its current orbit or might have maneuvered, even a small amount. If there is even slight evidence of a maneuver, the tracker will compute an impulsive maneuver hypothesis and start a new hypothetical state. Since there is inherent mathematical ambiguity in the non-deterministic behavior of maneuvers, it is typically not possible to be certain about any particular maneuver hypothesis over a short arc of a new orbit, especially when the orbital change is small and the observations lack range, as is the case for optical sensors which dominate observations of deep-space satellites.

Thus, at any one time, the tracker may have multiple hypotheses, which can include that the object did not maneuver; all of these working hypotheses will be carried forward to be corroborated or debunked by further measurements. As additional data are received, they will either confirm or weaken the generated hypotheses until it is clear that one of them is most likely correct.

The result of this processing is a history of orbital states with interleaved maneuvers, where the states have time bounds precisely at the moment of change. Fig. 8 shows a comparison of the catalog orbit updates with maneuver-aware states, in this case imported from the satellite operator [6]. It is apparent that the maneuver-aware history provides more information about the satellite behavior despite the fact that the catalog has more entries. Furthermore, the adjacent entries of the catalog appear choppy and sometimes have discontinuities that are not representative of reasonable satellite behavior.

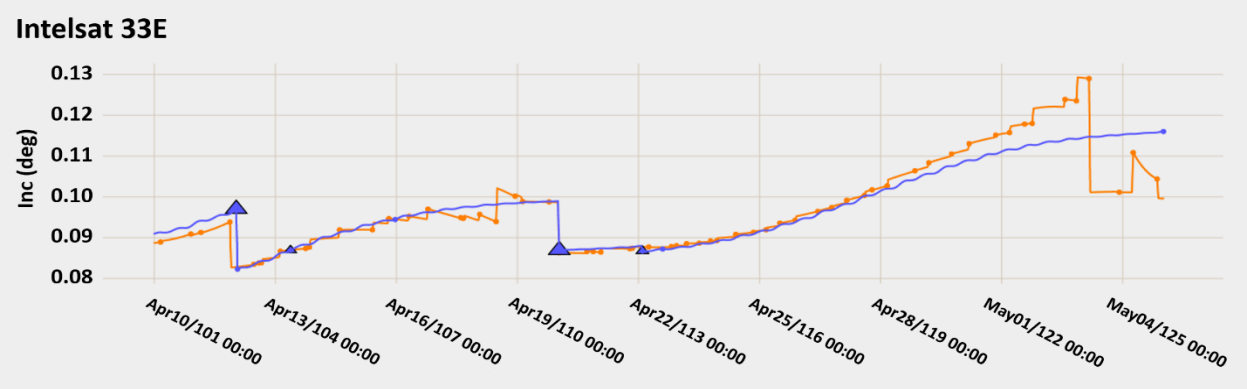

Figure 8: Intelsat 33E orbital states over 4 weeks. Standard TLE-based updates (orange) and maneuver-aware states (blue)

The choppiness of the catalog orbits is possibly due to its use of the Simplified General Perturbations (SGP4) analytical modeling method, which is relatively inaccurate for deep space objects, but it is evident from analyzing many orbits that updates surrounding maneuvers show more discontinuities; this is most likely due to the fact that only one latest state is allowed and it is often possible to slop through solving a single orbit solution using data both before and after a maneuver. The presence of discontinuities in both maneuver and non-maneuver cases debunks the working assumption the epoch-based catalog is meant for: that the newest catalog update is always the best. This is important when we return to the notion of tactical awareness and the consideration that we must have the best understanding of present behavior to assess and predict future actions or engagements.

4.3 Alternate Catalog Structure: The State Tree

As shown in the prior section in Fig. 5, the traditional catalog is a flat list of orbital updates that is driven by sensor activity, not satellite activity. As discussed, this was appropriate when the primary use of the catalog was to document and drive sensor collections on a list of known satellites. When satellites do not have static orbits and the objectives include understanding the history of their behaviors, a different structure is required. This new structure must focus on accurate object history, and must support multiple hypothesis representation of object state in order to support the maneuver aware states described in the previous section.

We propose a *state tree*, illustrated in Fig. 9 with a simplified logical representation of the state of one object. Each unique orbital state is a node in the tree, and the edges represent changes between states, usually impulsive maneuvers. The structure is inspired by the object model of the Git distributed version control system.

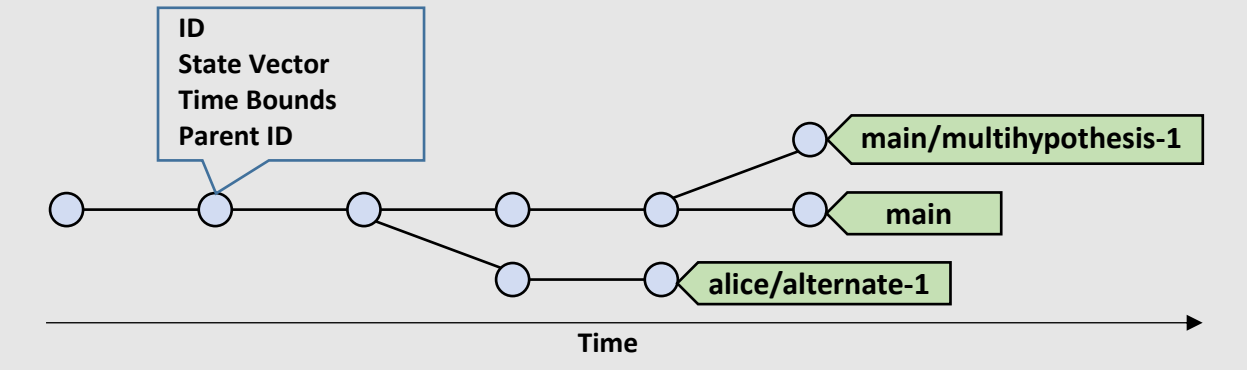

Figure 9: Illustration of State Tree data model for storing a history of orbital states of one object. The labels on the leaves indicate multiple hypotheses for the current state of the object. "main" represents the primary catalog state, while the other labels represent both human- and machine- generated alternate states.

In Fig. 10, we show the realization of the state tree as produced by tracking Intelsat 33E over 4 weeks, and compare it to the space catalog updates. Each node on the tree represents a new state, created after a maneuver happens, and contains the time bounds for its validity. Truncated branches are also shown – these states are tracker hypotheses that ended up not being supported by new data. In addition to the state tree construct's advantage of prioritizing the satellite's history of behaviors, it also provides a level of time clarity that the list of orbits by epoch does not. Specifically, we now know the start and end time of an orbit's validity, whereas the epoch-list-based catalog leaves the question of whether to propagate any particular orbit forward or backward from its epoch as an ambiguity the user must interpret.

Notice the maneuvers displayed as triangles on the smooth parameter history plot. Also notice how many fewer state updates there are in the state tree version: a historical state entry should not be updated if the propagation model accuracy is still high, and if the object has not maneuvered.

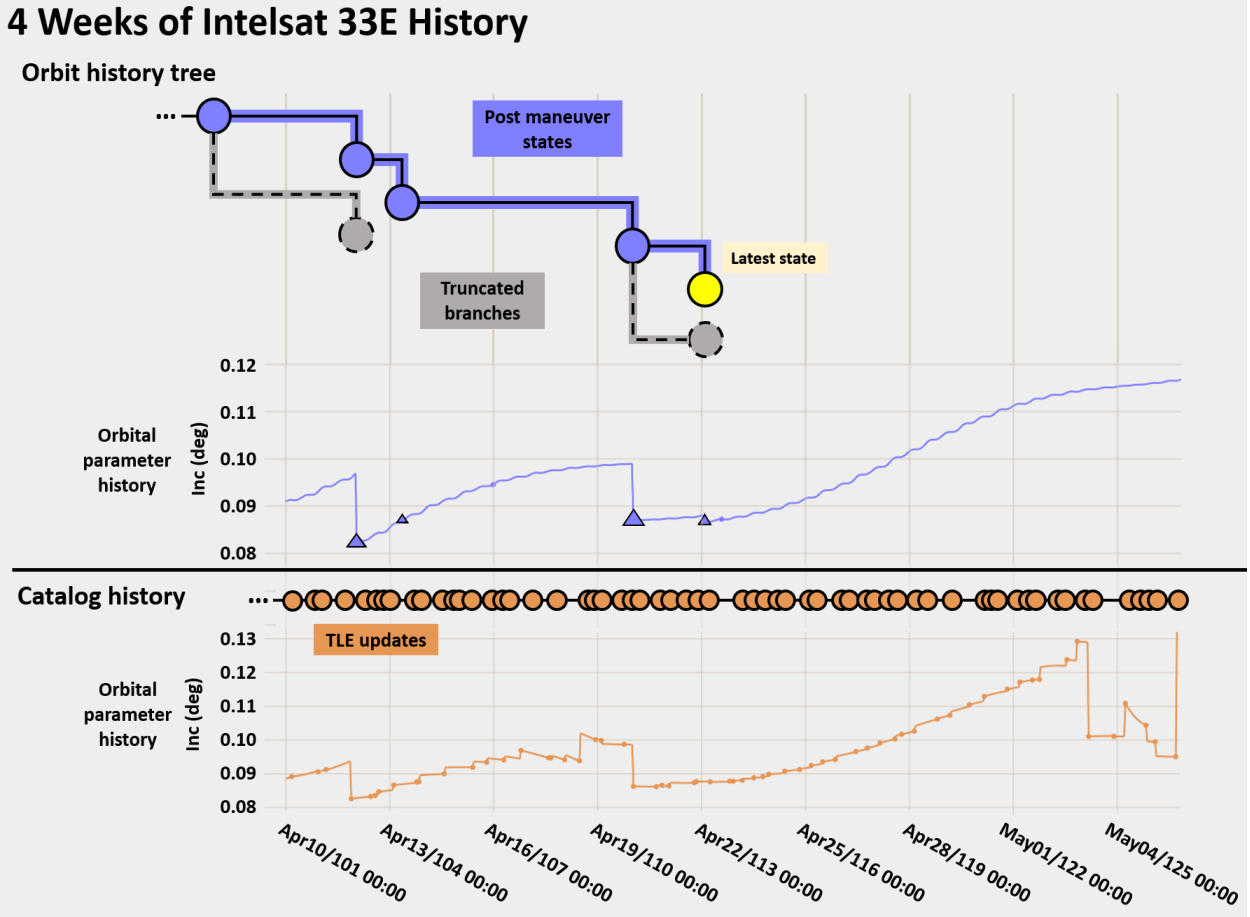

Figure 10: State tree vs flat catalog state list results on a geosynchronous, station-keeping satellite. Purple lines in the state tree represent current "main" leaf, while gray lines show past, pruned maneuver estimation hypotheses.

4.4 FURTHER ADVANTAGES OF STATE TREES

With the state tree model in place, we can step back and consider transformations on the tree structure, and the mapping into common analyst and operator use cases.

4.4.1 Revising history

Fig. 11 shows the use of the tree to revise a bad historical state. First a replacement state is created and pushed onto the tree. Then, the existing correct states that follow are rebased onto the replacement state. This revision process

does not perform any delete or update operations on the tree nodes, but instead pushes new information. This an important feature for distributed operations.

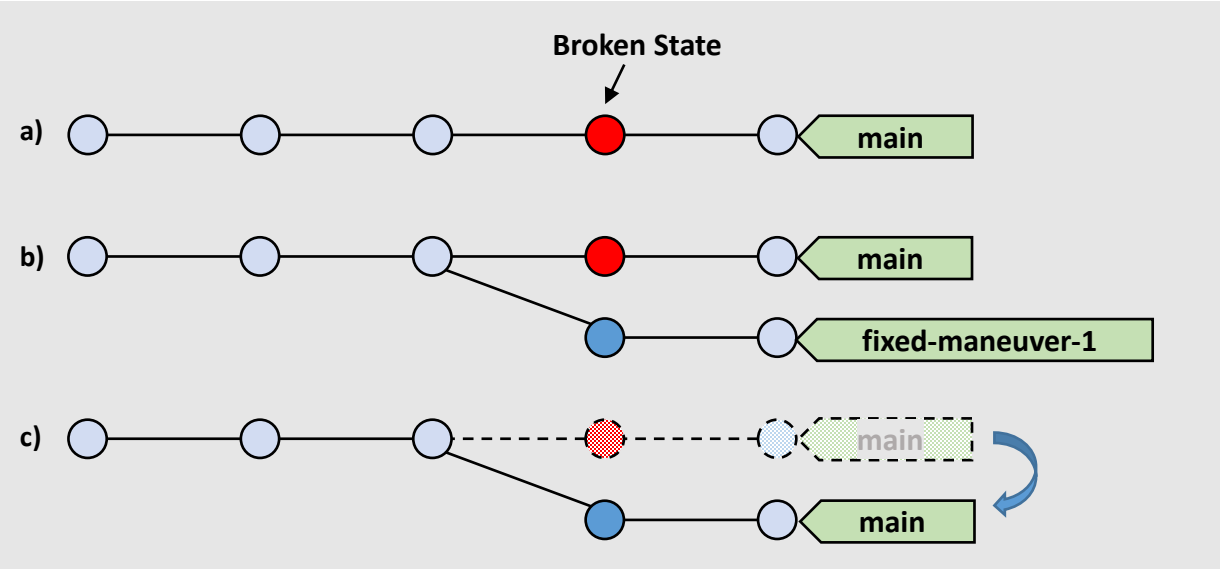

Figure 11: Editing out a bad state: a) State history with a bad entry b) Use working branch to create revised state, and rebase existing correct states on top of new state c) Working branch promoted to "main"

4.4.2 Distributed operations

Fig. 12 shows multiple (perhaps decentralized) users working on finding a lost object, and independently pushing new provisional states. The users might have access to different internal data sources or intelligence, or might come to different conclusions from the same data. These provisional states can be immediately used by sensors, communicated, and compared. Eventually the catalog authority can promote the best of the tree segments to the primary.

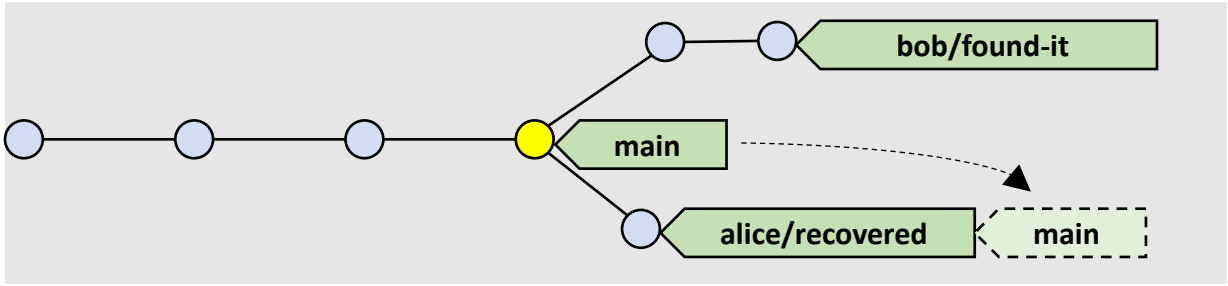

Figure 12: Provisional states. Two users are both working on recovering the same object. Both of these states are discoverable, useful, and easily communicated. The catalog authority can promote to main by relabeling

4.4.3 Future prediction

The state tree does more than represent and describe the history of a satellite: it provides a data model that allows one to predict and store future behaviors. The epoch-based linear catalog cannot handle the concept of future states that may or may not happen, but there are ready sources of information like this. For example, many satellite owner-operators publish their intended maneuvers ahead of time.

The state tree provides an obvious extension from creating multiple current (tracker) hypotheses to creating future (operator invented or planned) hypotheses. Since the state tree must already reconcile multiple working states, it has no conflict ingesting distinctly derived past, present, or future orbital states for sources such as these planned

maneuvers. These operator plan states are labeled and coexist with current tracker hypotheses as a superset set of working states on the satellite. This means one can store as many additional hypotheses or plans as desired while maintaining measurement-based states for comparison. Future states can include multiple maneuvers, for example:

- a series of intended orbit transfers in a deep-space launch sequence
- the station keeping maneuvers to be executed over a day in the life of a geosynchronous, electric-powered satellite.

Additionally, future states derived from advanced predictive algorithms can be stored and used here. Examples might include predicted station keeping from pattern-of-life models. Each set of planned orbits can simply be entered into a satellite's state tree as a hypothesis branch and any satellite can have multiple possible branches, waiting to be confirmed or denied by measured data used by the tracker. Consider that these distinct branches could also be used for upstream functions like observation correlation; Now correlation of sensor measurements could be done for each clear and specific orbit interval defined in the branch. This means that, not only could satellites be correlated and confirmed present, but expected behaviors could be automatically detected and confirmed as well.

Fig. 13 illustrates the processing of multiple potential future states with the state tree.

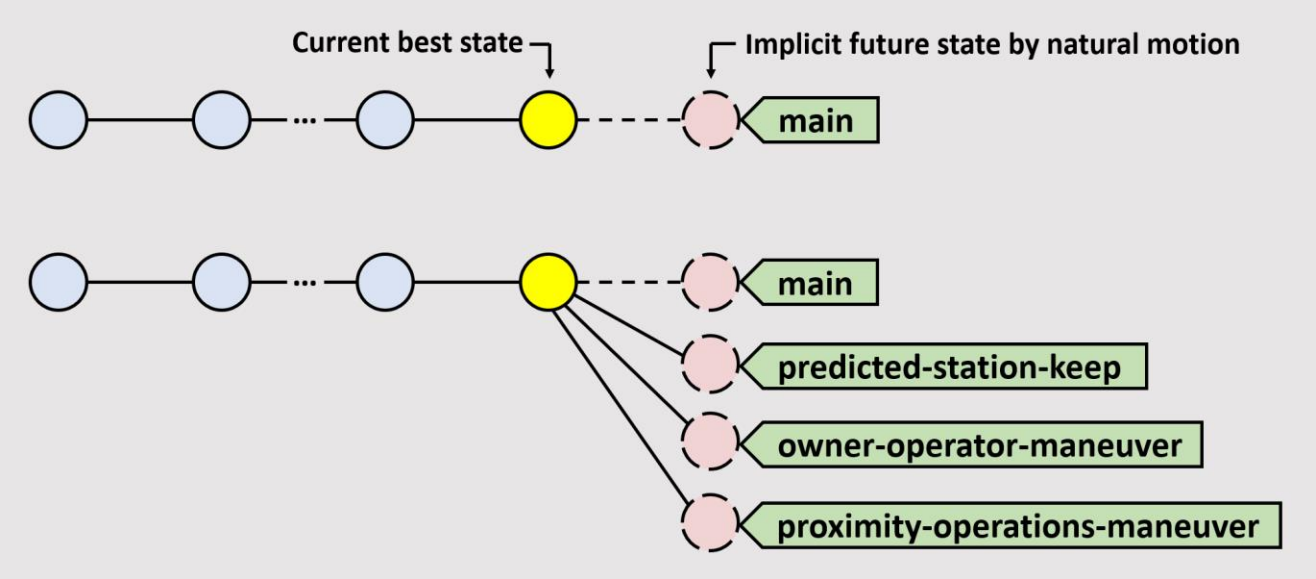

Figure 13: Hypothetical future states: Every state implicitly predicts a future state by natural motion propagation. However, alternate possibilities can be stored, communicated, used for sensor cueing, and checked for conjunctions

4.5 BEYOND THE MONOLITH CATALOG

Space operators regularly make use of data from multiple sources. For example, a large owner/operator such as Intelsat has an entire catalog of its own spacecraft that contains high-accuracy ephemerides, reflecting not only the past but also future planned orbits and maneuvers. Other sources include commercial and foreign SDA catalogs. The regular use of external sources is not readily possible in the epoch-based traditional catalog, which is considered a monolith of authoritative orbital data. To both challenge and solve this problem, we have implemented two related constructs: *object namespaces* and the *Universal State Locator* (USL*)*, which together provide the means for catalogs derived by different means to co-exist.

4.5.1 Object Namespaces

An object namespace is a prefix on the identifier of an object that sets the interpretation of that identifier to a (potentially external) source of data. Examples include NORAD number, owner-operator serial numbers, and international designators. The incorporation of this namespace into user-facing interfaces and software APIs allows

not only external data references, but also techniques like non-destructive object renaming. Table 1 shows a set of namespaces with an example identifier.

Table 1 Example Object Namespaces

Namespace Description	Object Identifier	Example Identifier	Fully-qualified Object
NORAD SCC Number	5-digit number	28358	norad.28358
Owner-operator ID	Internal serial number	IS-1002	intelsat.is-1002
Foreign space catalog	Satellite number	3531604	rohan.3531604
Discovered but unknown objects	Discovery id	365-01-001W	disc.365-01-001W
Objects to be launched	Provisional object names	17-OBJECTC	launch.17-objectc
User-defined objects	Temporary human-readable names	alice.recovered-object	u.alice.recovered-object

4.5.2 Universal State Locator

A Universal State Locator is a string descriptor that identifies a particular orbital state from an internal or external catalog on any object. Inspired by and named after the familiar URL, the USL provides a simple and human-readable interface that can function across systems, without imposing opinions on the format of the catalog itself. The form of the USL is **namespace.id:catalog/reference|hypotheses**. Fig. 14 shows the grammar for the USL. As an example, for a state from 30 January 2020 on Intelsat 1002, the USL would look like this: norad.28358:spadoc/20032.34959491.

The first element is a fully qualified object identifier, including the **namespace** as defined above.

The second element is the **catalog**, the particular database or ephemeris authority that stores the orbital state of an object. Examples of catalogs include spacetrack.org, owner-operator ephemerides such as Intelsat, commercial SDA providers, sensor-specific local catalogs, and R&D systems.

The **reference** is the third element, referring to a specific instance or variation of the object's state. This can be used to reference a particular orbit in a catalog at a specific time, or a particular branch of a state tree (see Section 4.3).

Finally, the USL supports the chaining of dynamic orbital **hypotheses** onto any referenced orbital state. Examples of hypothesis models that have been prototyped include "circularize" an orbit or "plane match" an orbit to another satellite's orbit at specified times (user chosen) or orbit events (e.g. apogee or nodal crossing). See Section 4.5.3 for more details on the hypothesis engine.

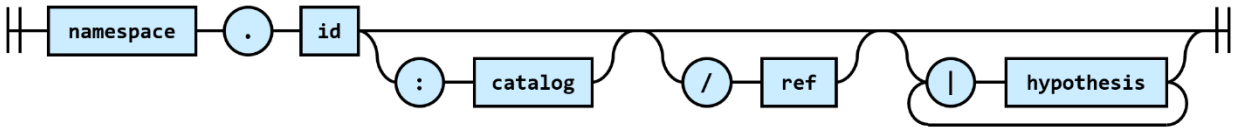

Figure 14: Universal State Locator (USL) grammar

The USL construct as prototyped has enabled and generalized many orbital analysis functions. A large variety of operator and analyst use-cases can be supported on top of a relatively small number of primitives that operate on USLs. Examples include:

- Analyze proximity operations by comparing two objects from the same catalog
- Evaluate a novel tracking algorithm by comparing the same object from two catalogs
- Plot a natural motion trajectory vs. a predicted maneuver trajectory
- Compare station keeping prediction algorithms vs owner-operator reported station keeping

4.5.3 USL Hypothesis Engine

A typical approach to producing hypotheses for maneuvering satellites is to use sophisticated analysis software such as AGI's STK to construct and visualize various scenarios and then export them as a set of TLEs for use in operations. While the ability to construct hypotheses is limited only by the sophistication of the analyst and analysis software, a number of operational difficulties arise:

- The hypothesis TLEs cannot simply be inserted into the catalog under the satellite's catalog number, as that would create confusion as to which states are based on observation and which are based on conjecture.
- Instead, the established practice is to use an "analyst number" in a reserved range of catalog numbers. However, there is no established way to express the relationship between analyst numbers and the catalog number of the object they are based on. Operators must coordinate use of analyst numbers.
- If observation substantially invalidates the prior state upon which a hypothesis was constructed, the entire sequence of TLEs must be reconstructed.
- While an analyst may be able to glean information from changes in orbital elements, a sequence of TLEs simply does not provide a clear description of the sequence of events.

While the state tree catalog inherently supports the concept of hypotheses, the process of revising hypotheses in the presence of new information remains burdensome as long as they are described in terms of static data. To address this, the USL allows description of hypotheses to be dynamically applied to any state history. Formally,

- A *state history* is a function of time that evaluates to position, velocity, and force model parameters (a state tree node). This may be realized by the linear history from a branch of a state tree, in which case the state history also describes the maneuvers between states; for the purposes of providing a base upon which hypotheses are applied a sequence of TLEs or any other form of ephemeris may also be used.
- A *hypothesis node* consists of a state history and time of validity.
- A *hypothesis* is a function of a hypothesis node and any number of arguments needed to describe the hypothetical action, evaluating to another hypothesis node.

Therefore, a sequence of hypotheses might be described in mathematical notation:

$$\boldsymbol{H_3(H_2\big(H_1(base, X_1)\big), X_3, Y_3\big)}$$

but recognizing that this notation is rather cumbersome for chains of hypotheses, the USL instead uses a "pipeline" notation:

$$\boldsymbol{base|H_1(X_1)|H_2|H_3(X_3, Y_3)}$$

In order to realize the benefits of state tree representation, the state history produced by a hypothesis must be realized as a state history pair, consisting of:

- a base state history, which can be any kind state history (e.g. state tree, TLE sequence, ephemeris)
- a hypothesis state history, which must be a state tree history

The state history pair evaluates as the base state history before the lower bound of the hypothesis state history, and as the hypothesis state history thereafter. The lower bound of the hypothesis state history is the validity time of the parent hypothesis node. Let us consider some concrete examples of hypotheses. The simplest hypothesis is *at(time),* which simply evaluates the current node's state history at a given time. This can be used to express natural motion propagation to a specific time beyond the span of observations supporting the base history, or to construct hypotheses based on a past time, to be compared to the observed state history. In addition to absolute (*at(time)*) and relative (*after(duration)*) times, one may also wish to describe orbital events; for example, an inclination lowering maneuver might be expected to occur at a nodal crossing, in which case one might specify *at_DN* or *at_AN* for descending node or ascending node, respectively, or *at_node* for the first nodal crossing, descending or ascending.

In addition to evaluating state history at given times and events, one must also be able to specify maneuvers. This can be done explicitly as in *vhr(DVv,DVh,DVr)*, specifying a 3D impulse in VHR (along track, cross track, radial) coordinates, or declaratively, as in *set_inclination(i)* which would produce a minimum delta-V maneuver resulting in a given inclination, or as close as possible at the current validity time. Note the lack of a time argument; it is desirable, when possible, to decouple the impulse with the time so that time and maneuver constructs can be composed freely as in:

at_DN|set_inclination(10)|set_meanmotion(2.27)

At this point one may be tempted to decompose all complex maneuvers into primitive components, e.g. expressing a Hohmann maneuver from GEO to a 1°/day eastward drift as:

set_perigee(35708346)|at_perigee|circularize

but not only is this excessively verbose for a common maneuver -- it also requires a different definition for altitude raising and lowering maneuvers. Therefore, we allow hypotheses to define arbitrarily complex maneuvers, e.g.

hohmann(1,by=drift_rate)

where the *hohmann* hypothesis is responsible for generating both maneuvers and the by keyword argument specifies the interpretation of the value of the first argument, so that one need not define a different hypothesis for every possible desired input form or resort to external calculations to convert the input.

Finally, one may wish to consider interactions with other satellites. Suppose we wish to hypothesize an approach to perform rendezvous proximity operations (RPO) with Eutelsat 10A. One might describe this hypothesis:

change_plane(raan=20.3,inclination=0.06,frame=TEME)|hohmann(1,by=drift_rate)
|geo_insert(10)

To hypothesize an approach to a different target in this manner would require looking up and inputting different orbital elements. By allowing the use of USLs as arguments to hypotheses, we can instead describe these maneuvers more straightforwardly:

plane_match(norad.34710:opeds)|hohmann(1,by=drift_rate)
|geo_insert_target(norad.34710:opeds)

The USL argument (norad.34710:opeds) results in a query for the target state history with the same time bounds as the base USL.

In order to evaluate hypothesis outcomes, we need to store their associated hypothesis state history. Fig. 15 illustrates how two hypothesis strings, starting from the same base state history, would be saved into a state tree.

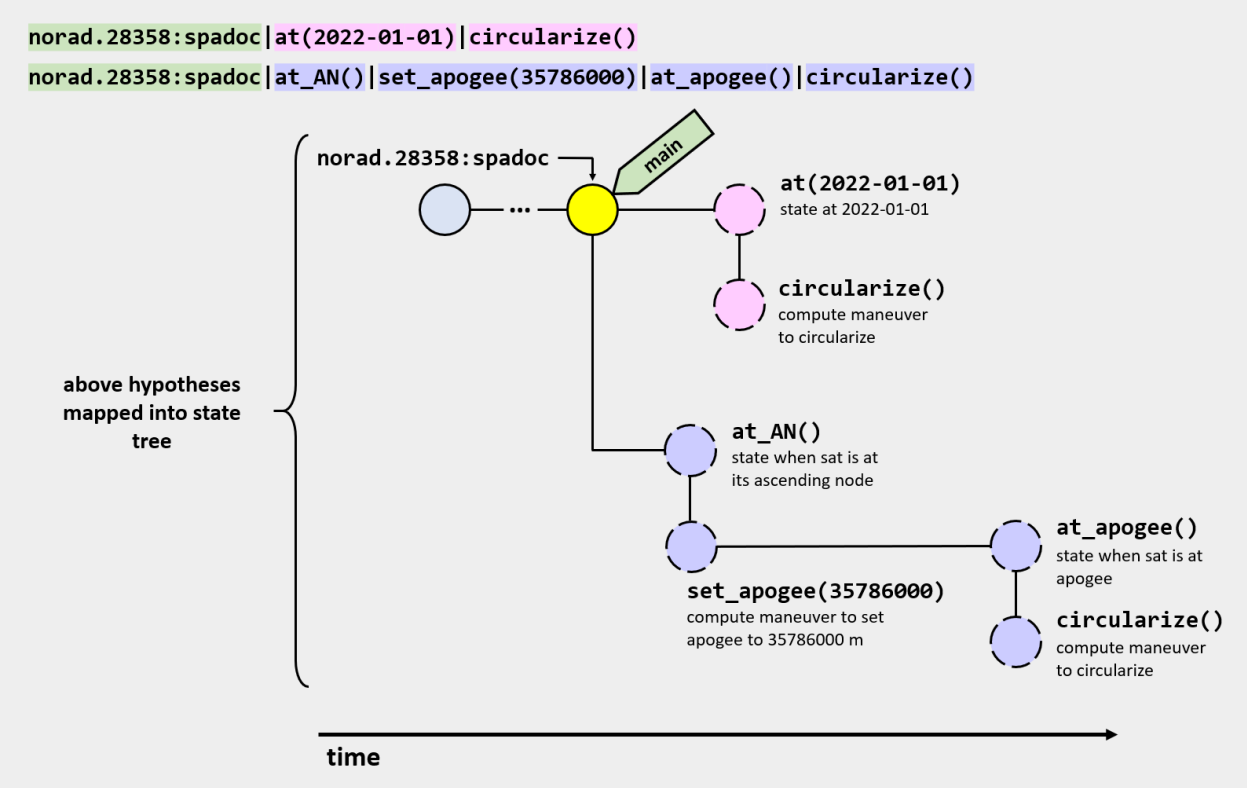

Figure 15: The creation of dynamic hypothesized states in the state tree, expressed using chained hypothesis transformations in the USL

5 DISTRIBUTED STATE TREE IMPLEMENTATION

The prototype version of the state tree written by the authors is implemented on top of a standard relational database. Each node is a row, storing both the orbital state and the row id of the parent node. A separate table lists the *refs* or *branch heads* for each object. With the proper indices, the trees for single objects can be efficiently retrieved, either in full or time-bounded. For efficiency, the states themselves should have time extents as long as possible, as opposed to storing every noise-control update as a new node.

A better implementation might be based on a true graph database, or a key-value store that implements graph operations. Regardless of the data store, a mature and operational state tree implementation has these features:

1. UUID-based identifiers for state and maneuver data.
2. Hash-based identifiers for tree nodes: Tree operations can be performed on decentralized systems and produce the same global state. As in Git, the hash of each node incorporates the node data and the parent identifiers.
3. Immutable data: New nodes and states can be added, but old ones should not be updated or removed, as they may be referenced by a different branch, or on a different distributed system.
4. A mechanism (e.g. a Merkle tree) to ensure synchronized global catalog state.

An illustration of such a tree implementation is shown in Fig. 16. In this implementation, maneuver and state data are each stored in nodes, rather than storing the maneuvers edges. This keeps the node objects themselves as lightweight as possible, so that operations like rebasing can use the same state data.

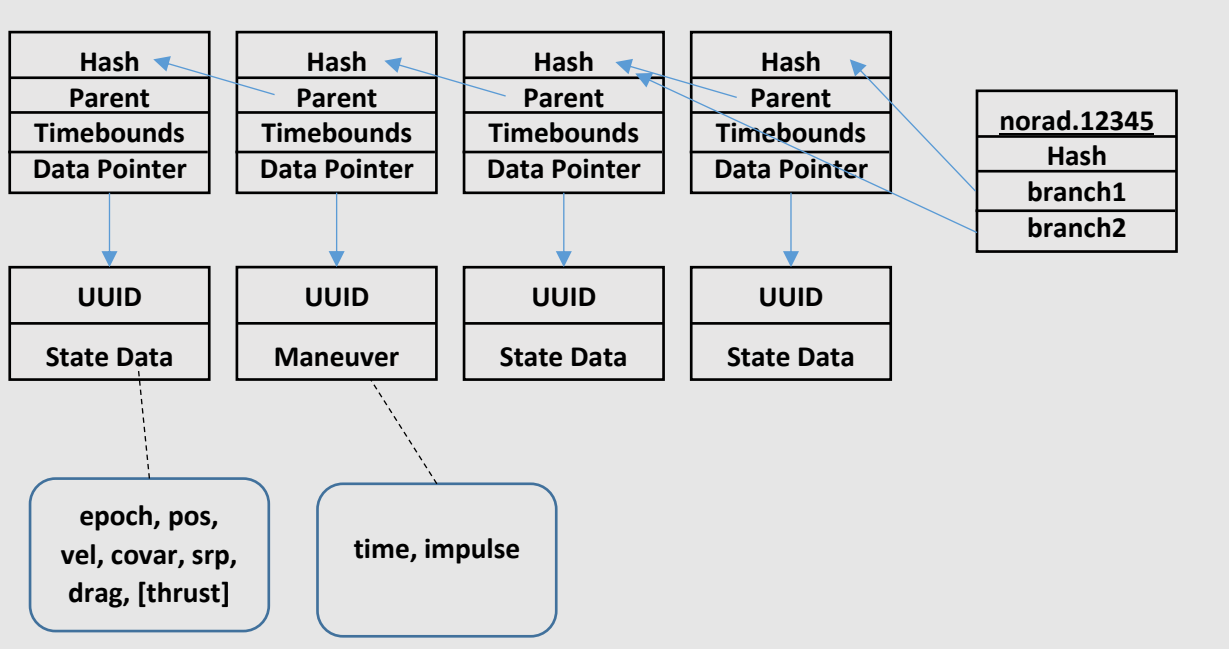

Figure 16: State tree for a single object based on UUID and hash node identifiers for distributed catalog maintenance

The management of an entire catalog of such trees is shown in Fig. 17. A Merkle tree is shown as the mechanism to synchronize the integrity of the tree. When the tree is transmitted to another system, any out-of-sync subtrees show up immediately as failed hash checks, allowing rapid identification of the subtree that requires replacement. As in the object state tree itself, all operations are "insert-only", so that the single top node of the Merkle tree captures the entire state of the catalog at any point in time. If the object tree heads were sorted from least- to most- active, a global catalog snapshot becomes an extremely lightweight operation: just store the top node of the Merkle tree. This allows full retrieval of the entire state of the catalog at any point in time.

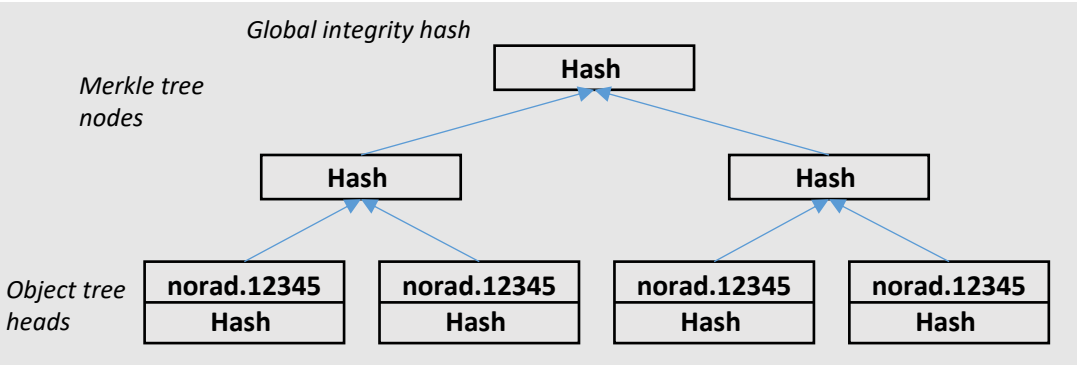

Figure 17: A Merkle tree is a potential technique to manage the synchronization and integrity of a large catalog of object trees.

6 CONCLUSION

Accomplishing today's space mission, which focuses on tactical awareness for hazard mitigation, requires that we modernize the data models we use to describe objects in space. While we propose complicating the simple list-of-latest-states catalog that was formerly good enough to constitute Space Situational Awareness, we argue that upgrading to a state tree gives us capabilities that are essential in a congested, contested, and distributed space

operations environment. As we have described in this paper, it is possible and useful to construct our alternative state tree model using real sensor data on deep space objects, particularly those in geosynchronous orbits that maneuver frequently. A top-level summary of the capabilities gained from the multi-hypothesis state tree we propose is as follows:

1. A satellite-focused state tree offers physics-realistic models of satellite behavior (orbit state and maneuver) history over time
2. A state tree offers the means to describe multiple simultaneous possible states, whether that is required due to measurement ambiguities or functionally useful for alternate future behavior hypotheses based on automated pattern of life analytics or an analyst's contextual cues about a satellite's intent
3. Ability to describe multiple simultaneous possible states permits description and storage of planned or hypothesized future satellite behaviors as well as links to metadata about a satellite's associated behaviors or interactions
4. Extension of tree model to a forest of multiple possible input sources on a satellite's location and behavior permits concurrent use of measured and planned satellite actions and/or cataloged and uncatalogued orbital states by a set of distributed users

We have described the importance of the capabilities above in terms of answering basic tactical awareness questions required to understand behaviors of satellites over time to enable pro-active operations in space. We have also described how the model maps easily to distributed ledger accounting, which allows improved data participation between military, civilian, and international users with different information and data sources about satellites. Our current work, which builds on this data construct, involves the use of Artificial Intelligence and Machine Learning techniques to compliment measured data to add both automated and user-initiated contextual information to the data model. We have found that the state tree is essential to capturing sound inputs to classification and prediction algorithms we develop and its flexible distributed nature important to linking non-measured contextual information to orbital data. We invite other members of the community to experiment with adopting the state trees we describe for their own applications.

REFERENCES

[1] D. S. F. Portree and J. P. J. Loftus, *Orbital Debris: A Chronology*, NASA Johnson Space Center Technical Publication, 1999.

[2] "Punched Card," [Online]. Available: https://en.wikipedia.org/wiki/Punched_card.

[3] F. R. Hoots and R. L. Roehrich, "Spacetrack Report No. 3: Models for propagation of NORAD element sets," 1980.

[4] D. Vallado, P. Crawford, R. Hujsak and T. Kelso, "Revisiting Spacetrack Report #3," *AIAA/AAS Astrodynamics Specialist Conference and Exhibit*, 2006.

[5] A. Murtaza, S. J. H. Pirzada, T. Xu and L. Jianwei, "Orbital Debris Threat for Space Sustainability and Way Forward (Review Article)," *IEEE Access*, pp. 61000-61019, 2020.

[6] "Intelsat Ephemeris Data," [Online]. Available: https://my.intelsat.com/ephemeris/public.