Early Design Exploration of Space System Scenarios Using Assume-Guarantee Contracts

Nicolas Rouquette*§, Inigo Incer†§, and Alessandro Pinto*
* Jet Propulsion Laboratory, California Institute of Technology
† California Institute of Technology, Pasadena, California, USA

nfr@jpl.nasa.gov, inigo@caltech.edu, alessandro.pinto@jpl.nasa.gov

***Abstract*—We present a compositional approach to modeling and analyzing space mission operation sequences with steps across multiple viewpoints. We consider different operation tasks such as communication, science observation, trajectory correction, and battery charging; and separate their interactions across discipline viewpoints. In each sequence step, these tasks are modeled as assume-guarantee contracts. They make assumptions on the initial state of a step, and if these assumptions are satisfied, they guarantee desirable properties of the state at the end of the step. These models are then used in Pacti, a tool for reasoning about contracts. We demonstrate a design methodology leveraging Pacti's operations: contract composition for computing the contract for the end-to-end sequence of steps and contract merging for combining them across viewpoints. We also demonstrate applying Pacti's optimization techniques to analyze the figures of merit of admissible sequences satisfying operational requirements for a CubeSat-sized spacecraft performing a small-body asteroid rendez-vous mission. We show that analyzing tens of thousands of combinations of sequences and operational requirements takes just over one minute, confirming the scalability of the approach. The methodology presented in this paper supports the early design phases, including requirement engineering and task modeling.**

***Index Terms*—Space mission, operation scheduling, contracts, assume-guarantee reasoning**

I. Introduction

Space-mission designers partition an overall mission scenario into major operational phases. Each operational phase involves a sequence of segments, each corresponding to multiple tasks of spacecraft subsystems. Within an operational phase, the goal is to fulfill the phase scenario requirements by appropriately selecting subsystem tasks across the scenario's segments.

The designer must negotiate specifications with the subsystem-cognizant engineers responsible for defining subsystem tasks. In the early stages of system design, cognizant engineers guide the mission designer regarding the capabilities and requirements of subsystem tasks. While valuable, this input does not allow the mission designer to efficiently consider how to allocate resources across subsystems involved in a scenario, as this feedback loop requires rapid conversations and negotiations among engineers. A potential solution would be for cognizant engineers to provide the designer with detailed models of their subsystems, which the mission designer could use to simulate various scenarios involving resource allocations to segment subsystems. However, engineers may not have sufficiently detailed simulation models of their subsystems in the early design phases. Moreover, as these simulations typically require significant software setup and run time, they only yield insight into carefully-chosen operating conditions and limit the usefulness of simulation for exploring multiple what-if scenarios quickly.

In the early phases of design, the requirement development process should be supported by an agile and interactive methodology, while decisions should be informed by tools. The elicitation of system and sub-system requirements is a complex and time-consuming activity which is primarily a document-driven process conducted collaboratively by many subject matter experts (SMEs) across different domains. The heterogeneous nature of this expertise makes collaboration and resource negotiation difficult, particularly when complex simulation models are involved. In this phase, one of their goals is to negotiate budgets and resources across sub-systems, which must consider many constraints. Automation in managing these constraints could reduce potential errors that are costly to fix in the later stages of design [1].

Two key aspects of an agile and reliable requirement development process are abstraction and verification speed, which are often related. In the early phases, we seek fast turnaround from a candidate design or scenario to figures of merit, and quick verification that the requirement breakdown satisfies the mission objectives. The input to this agile analysis of trade-offs must take mathematically-meaningful abstract specifications of subsystems rather than their detailed simulation models, which may not yet be available[§]. As subsystems are being developed, specifications can be updated. With proper tool support, the mission designer could get rapid feedback about the impact of changes to subsystem specifications on the overall mission objectives. Finally, the number of SMEs involved in requirement definition spans many domains, demanding methodologies and tools that enable them to focus on a specific aspect or viewpoint of the mission. For example, when analyzing power utilization, the methodology and tools should provide means

§Equal contribution

§As stated in [2]: "information about the capabilities of the subsystems in terms of timing, power consumption, size, weight and other physical aspects transmitted to the system assemblers during design time would go a long way in providing a better opportunity to design space exploration".

to focus only on the relevant power-related concerns, hiding aspects of the design pertaining to other viewpoints, such as navigation or science.

In this paper, we consider exploring the space of combinations of subsystems' tasks for a given scenario using assume-guarantee specifications, or *contracts*, corresponding to abstractions of the subsystem's task behavior. We leverage contract algebra [3], [4] to compose constraints in several ways and demonstrate how it can be used to understand the overall constraints resulting from modeling a scenario sequence as the composition of subsystem tasks. Besides composition, other algebraic operations on contracts provide the designer with powerful analytical means to explore scenario refinements and quantify gaps in scenarios. Furthermore, this algebraic approach extends naturally to analyzing viewpoints as contract refinements, thereby enabling the analysis of multiple viewpoints and the characterization of gaps with respect to overall scenario requirements.

We use the recently-developed tool Pacti [5] to manipulate contracts[§]. Pacti was designed with computational efficiency as a top concern. As a result, a designer can use it interactively to gather quick feedback (less than three seconds in our experiments) about the impact of varying design parameters. Thus, we believe the designer of phase scenarios can benefit significantly by specifying subsystem operational behaviors as task contracts and manipulating them using Pacti.

Our discussion of this contract-based approach revolves around designing and operating a CubeSat-sized spacecraft performing a small-body asteroid rendez-vous mission. We discuss modeling the mission scenario as a sequence of task-specific steps and how to model them as contracts. We use the notion of contract viewpoints to split a complex contract into simple subcontracts focused on a single viewpoint, such as power, science & communication, and navigation. Once we have defined the contracts for each task, we use Pacti to carry out the following tasks: composition for sequencing task-specific steps, merging for fusing such sequences across viewpoints, optimization for computing figures of merit such as bounds on the average battery state of charge across a sequence, and evaluating bounds for state variables at arbitrary steps in the sequence.

Section II provides an overview of contracts and Pacti. Section III presents a case study of early design exploration of a space mission operation involving a small-body asteroid. We model subsystem-specific tasks as contracts and use Pacti to obtain insight into many system-level aspects. We conclude in Section VI. In the remainder of the paper, we will use the terms subsystem and component interchangeably as appropriate for the context. The specifications and analyses described in this paper are available at this repository:

II. Overview of Contracts & Pacti

Pacti [5] helps designers to reason about specifications and to manipulate them. These specifications are given to Pacti as assume-guarantee contracts, which are pairs (A, G) where A is a set of assumptions, and G a set of guarantees. Contracts resemble the form in which requirements are typically written.

For Pacti, a contract has four elements:

- A set of *input variables*.
- A set of *output variables*.
- A set of *assumptions* that are constraints on the input variables.
- A set of *guarantees* that are constraints on both input and output variables.

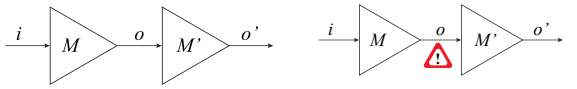
Intuitively, the assumptions define a set of possible environments in which the subsystem can be used. The guarantees define the input-output relation that the subsystem promises to enforce when the assumptions are satisfied. Pacti currently supports constraints expressed as linear inequalities, also called *polyhedral constraints*, but its architecture is extensible to other constraint formalisms.

The algebra of contracts has been formalized in several previous works (see, for instance [3], [4], and references therein). The formalization includes the definition of several operators and their properties. These operators can be used to address several tasks relevant to system design, including:

- Building systems out of subsystems. Suppose that we have specified contracts for a set of subsystems. We can define a system as the assembly of such subsystems. The operation of *composition* allows us to compute the contract of such a system from the contracts of the assembled subsystems. In other words, the composition operator provides a mechanism for computing system contracts from subsystem contracts.
- Patching systems. The operation of *quotient* allows us to compute the contract of a subsystem that needs to be composed with an existing subsystem so that the resulting system composition meets a top-level contract. In other words, the quotient finds contracts of missing subsystems from contracts for the system and a partial implementation.
- Validity of decompositions. *Refinement* allows us to tell when a contract is more relaxed, or less demanding than another. When a subsystem satisfies a contract, it is guaranteed to satisfy a more relaxed contract. When a system contract is broken into an assembly of subsystem contracts, refinement allows us to tell whether this decomposition is a valid refinement of the system-level contract.
- Fusing viewpoints. The operation of *merging* allows us to generate a single contract whose assumptions and guarantees require the satisfaction of the assumptions and guarantees of the merged contracts, respectively. In other words, merging fuses multiple contract viewpoints, a common operation in concurrent design.

The following sub-sections provide an overview of how Pacti supports these common tasks.

[§]Pacti is available as an open-source package under a BSD 3-Clause license at https://github.com/pacti-org/pacti

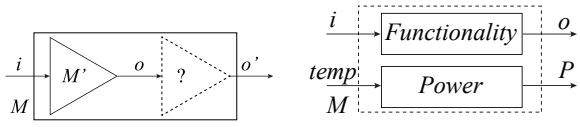



```
c1 =
    PolyhedralContract.from_string(
    InputVars=["i"],
    OutputVars=["o"],
    assumptions=["|i| <= 2"],
    guarantees=["o - i <= 0",
                "i - 2o <= 2"])

c2 =
    PolyhedralContract.from_string(
    InputVars=["o"],
    OutputVars=["o_p"],
    assumptions=["o <= 0.2",
                 "-o <= 1"],
    guarantees=["o_p - o <= 0"])

sys_contract = c1.compose(c2)
print(sys_contract)

# Output:
InVars: [i]
OutVars:[o_p]
A: [
    i <= 0.19999999999999996
    -0.5 i <= 0.0
]
G: [
    -i + o_p <= 0.0
]
```

```
c1_n =
PolyhedralContract.from_string(
    InputVars=["i"],
    OutputVars=["o"],
    assumptions=["|i| <= 2"],
    guarantees=["|o| <= 3"])

c2 =
    PolyhedralContract.from_string(
    InputVars=["o"],
    OutputVars=["o_p"],
    assumptions=["o <= 0.2",
                 "-o <= 1"],
    guarantees=["o_p - o <= 0"])

new_sys_contract = c1_n.compose(c2)

# Output:
IncompatibleArgsError: Could not
eliminate variables ['o'] by
refining the assumptions
[
    o <= 0.19999999999999996
    -o <= 1.0
]
using guarantees
[
    |o| <= 3.0
]
```

(a) (b)

Fig. 1: (a) System composition (b) System diagnostics




```
c_top =
    PolyhedralContract.from_string(
    InputVars=["i"],
    OutputVars=["o_p"],
    assumptions=["|i| <= 1"],
    guarantees=["o_p - 2i = 1"])

c_partial =
    PolyhedralContract.from_string(
    InputVars=["i"],
    OutputVars=["o"],
    assumptions=["|i| <= 2"],
    guarantees=["o - 2i = 0"])

c_missing = c_top.quotient(c_partial)
print(c_missing)

# Output:
InVars: [o]
OutVars:[o_p]
A: [
    |o| <= 2.0
]
G: [
    -o + o_p = 1.0
]
```

```
funct_vp =
    PolyhedralContract.from_string(
    InputVars=["i"],
    OutputVars=["o"],
    assumptions=["|i| <= 2"],
    guarantees=["o - 2i = 1"])

pwr_vp =
    PolyhedralContract.from_string(
    InputVars=["temp"],
    OutputVars=["P"],
    assumptions=["temp <= 90"],
    guarantees=["P <= 2.1"])

sys_cont = funct_vp.merge(pwr_vp)
print(sys_cont)

# Output:
InVars: [i, temp]
OutVars:[o, P]
A: [
  |i| <= 2.0
  temp <= 90.0
]
G: [
  -2.0 i + o = 1.0
  P <= 2.1
]
```

(a) (b)

Fig. 2: (a) Patching systems (b) Fusing viewpoints

A. Computing system specifications

Consider the system shown in Figure 1a. Subsystem M has input i and output o, and M' has input o and output o'. The assumptions and guarantees of M are, respectively, $\{|i| \leq 2\}$ and $\{o \leq i \leq 2o + 2\}$, while the assumptions and guarantees of M' are, respectively, $\{-1 \leq o \leq 0.2\}$ and $\{o' \leq o\}$. The figure shows how we can use Pacti to obtain the contract of the system formed by assembling these two subsystems. Pacti tells us that the system contract has input i, output o', assumptions $\{0 \leq i \leq 0.2\}$, and guarantees $\{o' \leq i\}$. Pacti's answer only involves the top-level input and output variables, having eliminated the intermediate variable, o.

B. System diagnostics

In Figure 1b, we have the same subsystems as those shown in Figure 1a, except that the guarantees of M have been replaced by $\{|o| \leq 3\}$. When we try to form a system using M and M', Pacti tells us that the guarantees of M are insufficient to satisfy the assumptions of M'. Indeed, M' requires its input to be bounded by 0.2, while M guarantees this signal to be bounded by 3. Pacti thus flags a potential flaw in our design.

C. Specifying missing subsystems

Figure 2a shows the situation in which we will implement a system M with input i and output o' having assumptions $\{|i| \leq 1\}$ and guarantees $\{o' = 2i + 1\}$. To implement this system, we will use a subsystem M' with input i, output o, assumptions $\{|i| \leq 2\}$ and guarantees $\{o = 2i\}$. To implement the top-level specification using M', we have to identify the specification of the missing subsystem denoted by a question mark in the figure. Pacti computes this missing-subsystem specification for us, saying that this subsystem will have input o, output o', assumptions $\{|o| \leq 2\}$ and guarantees $\{o' = o + 1\}$.

D. Fusing viewpoints

Contract-based design enables us to organize specifications in categories, or viewpoints. Figure 2b show a subsystem M with two different contracts assigned to it: a functionality contract and a power contract. The operation of merging can generate a single contract that contains both viewpoints of the design. When performing analysis, we use only the subsystem specifications for the task at hand. For example, to carry out power analysis of an entire system, we should be able to use only the power viewpoints of the subsystems that compose it.

III. Small-body asteroid mission

We consider the design and analysis of operational space mission scenarios using Pacti[§]. This requires a paradigm shift from modeling the temporal behavior of subsystems to modeling properties of these behaviors through contracts [2], [3]. We focus on the problem of operating a CubeSat-sized spacecraft performing a small-body asteroid rendezvous mission as described in detail in [6]. For autonomous operation, this would involve an onboard planning and scheduling system based on a generalized timeline representation as described in [7]. Examples of such timeline representation can be found in [8], [9]. Typical models used in planning and scheduling involve computationally efficient linear inequalities of the form $a \cdot t \leq b$ where t is a time, a is a rate constant, and b is a value constant.

[§]This case study is available at https://github.com/pacti-org/cs-space-mission, tag: SMC-IT-2023.

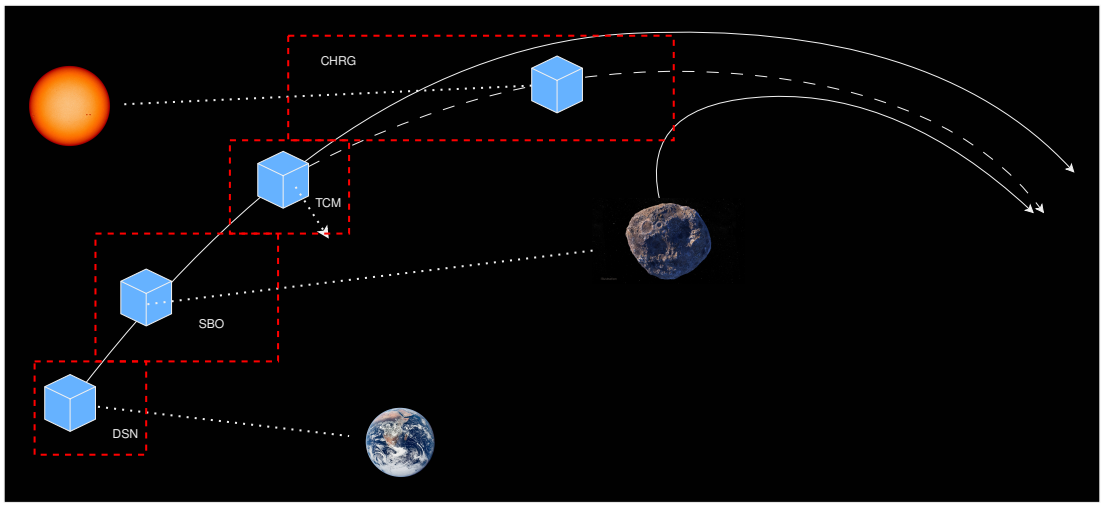


Fig. 3: Segmented space mission scenario for asteroid approach

This class of constraint formulas fits within the expressiveness of Pacti's polyhedral constraints. Thus, we explore modeling tasks as assume-guarantee contracts using Pacti. Figure 3 illustrates a simplification of the small-body asteroid approach scenario described in more detail in [6][§]. To communicate with Earth, the spacecraft (blue cube) must orient its fixed antennas towards Earth (DSN). Depending on the trajectory, this orientation may be sub-optimal for the spacecraft panels to produce maximum electrical power (CHRG). Optical science measurements require orienting the spacecraft's camera towards the asteroid for observation (SBO). Autonomous navigation requires a different orientation to yield the desired velocity change when performing a Trajectory Correction Maneuver, TCM. Reaching the asteroid will require a scenario involving multiple occurrences of these tasks DSN, SBO, TCM, and CHRG while ensuring the spacecraft remains powered, and the ebb and flow of science data measurements and downlink keep the onboard storage from overflowing.

A. Scenario modeling methodology

Figure 4 illustrates the application of the composition-based strategy of II-A for modeling scenario steps as contracts and sequences of such steps as the composition of contracts. Each scenario step, e.g., A,B, involves a Pacti contract for a corresponding task instance specifying the system behavior during that step in terms of entry and exit conditions on state variables as assume and guarantee constraints, respectively. For example, task A specifies assumptions in terms of its input variables $\{V^{\mathtt{A}}_{\mathtt{entry}}, \Delta T_{\mathtt{A}}\}$ only, and guarantees in terms of both input and output variables $\{V^{\mathtt{A}}_{\mathtt{entry}}, V^{\mathtt{A}}_{\mathtt{exit}}, \Delta T_{\mathtt{A}}\}$. The left-to-right direction within each task from entry-to-exit and between tasks from a previous task's output, e.g. $V^{\mathtt{A}}_{\mathtt{exit}}$, to the next task's input, e.g., $V^{\mathtt{B}}_{\mathtt{entry}}$, helps convey the overall causality of modeling an ordered temporal sequence of steps using Pacti's contracts. Schedulability analysis, by which we mean the joint exploration of schedule steps, resources, and operational requirements, involves refining the scenario sequence contract, e.g., task B starting after the completion of A, with operational requirements. Such requirements specify constraints on the initial conditions, $V_{\mathtt{initial}}$, final conditions, $V_{\mathtt{final}}$, task durations, $\Delta T_{\mathtt{A}}, \Delta T_{\mathtt{B}}$, and performance, $V^{\mathtt{A}}_{\mathtt{exit}}, V^{\mathtt{B}}_{\mathtt{exit}}$. Finally, extending Pacti contracts with constant hyperparameters, such as minimum and maximum power generation $gen^{\mathtt{A}}_{\mathtt{min,max}}$ for A, and minimum and maximum power consumption $cons^{\mathtt{B}}_{\mathtt{min,max}}$ for B, enables applying hyperparameter sampling techniques to generate multiple Pacti scenario contracts and to perform schedulability analysis for each scenario. Our experience so far is that the computational complexity of this approach remains within practical considerations given that analysis in Pacti requires solving a number of linear programming problems proportional to the number of constraints involved. This means that constructing the contract for a scenario involving a finite number of steps to be composed and a finite number of operational requirements to be refined will result in a fixed upper bound on the number of linear programming problems to be solved. Since computing schedulability analyses over hyperparameter samples is easily parallelizable, this exploratory methodology enables a rapid turnaround between scenario contract modeling and analysis results.

B. Pacti contracts for scenario tasks

The following summarizes the different task types shown in Fig. 3 in terms of the different spacecraft attitude pointing requirements:

- DSN Orient the spacecraft's antenna towards Earth to downlink science data.
- SBO Orient the spacecraft's camera towards the asteroid for science and navigation observations.
- TCM Orient the spacecraft's chemical thrusters in a direction to perform a Trajectory Correction Maneuver

[§]The Sun, Earth, spacecraft, and small-body asteroid are shown at different scales for illustration purposes.

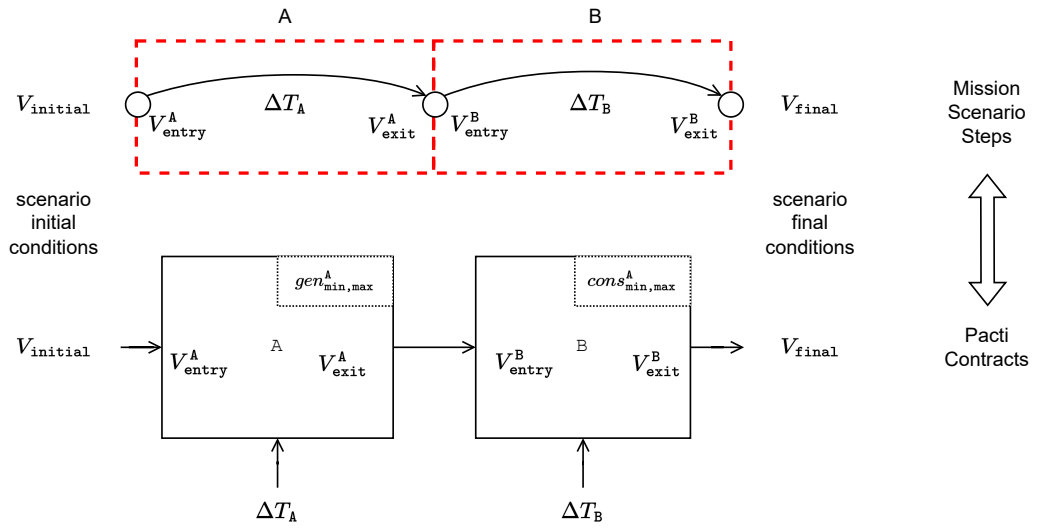


Fig. 4: Contract modeling of scenario steps and sequences

Viewpoint	State	DSN	SBO	TCM	CHRG
Power	soc	$-$	$-$	$-$	$+$
Science & Communication	d	$-$	$+$	0	0
	c	0	$+$	0	0
Navigation	u	$+$	$-$	$+$	$+$
	r	0	0	$+$	0

TABLE I: Qualitative Task Impacts

Variable	Values	Description
soc	$[0, 100\%]$	Battery state of charge
d	$[0, 100\%]$	Onboard science data storage
c	$[0, \infty]$	Cumulative science data acquired
u	$[0, 100\%]$	Relative trajectory estimation uncertainty
r	$[0, 100\%]$	Relative trajectory progress

TABLE II: State Variables

computed onboard to bring the spacecraft's trajectory closer to that of the asteroid.

`CHRG` Orient the spacecraft's solar panels towards the Sun to charge the battery.

Table I summarizes the qualitative impacts of each type of task on key mission parameters, where $+$, 0, and $-$ denote, respectively, positive, independent, and negative correlation of state variables described in Table II with respect to task duration.

To describe the Pacti contracts we authored in the case study, we name constants to convey their nature and adopt the following concise notation:

- $x \in [\gamma_{\texttt{min}\cdots\texttt{max}}]\Delta T = \{x \mid \gamma_{\texttt{min}}\Delta T \leq x \leq \gamma_{\texttt{max}}\Delta T\}$
- $[\gamma_{\texttt{min}\cdots\texttt{max}}] = [\gamma_{\texttt{min}}, \gamma_{\texttt{max}}]$
- $v_{\texttt{exit}-\texttt{entry}} = v_{\texttt{exit}} - v_{\texttt{entry}}$

where x is an expression, γ_{min} and γ_{max} are constants, and v_{entry} and v_{exist} are variables. All tasks assume valid ranges of input variables: task duration must be positive, if applicable, and other inputs must be within valid ranges. The salient differences arise in the guarantees modeling the task impacts on the state variables as described in the following viewpoint subsections.

Before we can analyze the schedulability of a mission operation scenario against operational requirements in Section III-E, we need to construct the scenario contract. Figure 5 shows a representative operation scenario involving a sequence of the following tasks: `DSN`, `CHRG`, `SBO`, and `TCM`, which is decomposed into two subtasks, heating, `TCM_h`, and a delta-v maneuver, `TCM_dv`. We leverage Pacti's support for fusing viewpoints and break down the specification of each task across the three viewpoints described above: power, science & communication, and navigation.

C. Pacti contracts in the power viewpoint

Qualitatively, `DSN`, `SBO`, and `TCM` have similar power-consuming contracts, whereas `CHRG` has a power-generation contract. The guarantees assert that the change in state of charge will be proportional to a generation or consumption rate applied for the task's duration. `TCM` involves two different power-consuming behaviors, thruster heating and delta-V, modeled as two subcontracts: `TCM_h` and `TCM_dv`. Thus, the 4-step scenario becomes the composition of 5 steps.

The task template notation below uses the following abbreviations: `CST` for constant hyperparameters, `In,Out` for input and output variables, respectively, and `A,G` for contract assumptions and guarantees, respectively.

Task template for $\mathcal{T}$=`CHRG`

`CST`	$\text{pgen}^{\mathcal{T}}_{\texttt{min,max}}$
`In`	$\text{soc}_{\texttt{entry}}, \Delta T^{\mathcal{T}}$
`Out`	$\text{soc}_{\texttt{exit}}$
`A`	$\Delta T^{\mathcal{T}} \geq 0 \;\;\&\;\; \text{soc}_{\texttt{entry}} \geq 0$
`G`	$\text{soc}_{\texttt{exit}-\texttt{entry}} \in [\text{pgen}_{\texttt{min}\cdots\texttt{max}}]\Delta T^{\mathcal{T}}$ $\text{soc}_{\texttt{exit}} \in [0, 100]$

The constant hyperparameter, $\text{pgen}^{\mathcal{T}}_{\texttt{min,max}}$, defines the range of power generation charging the battery during an

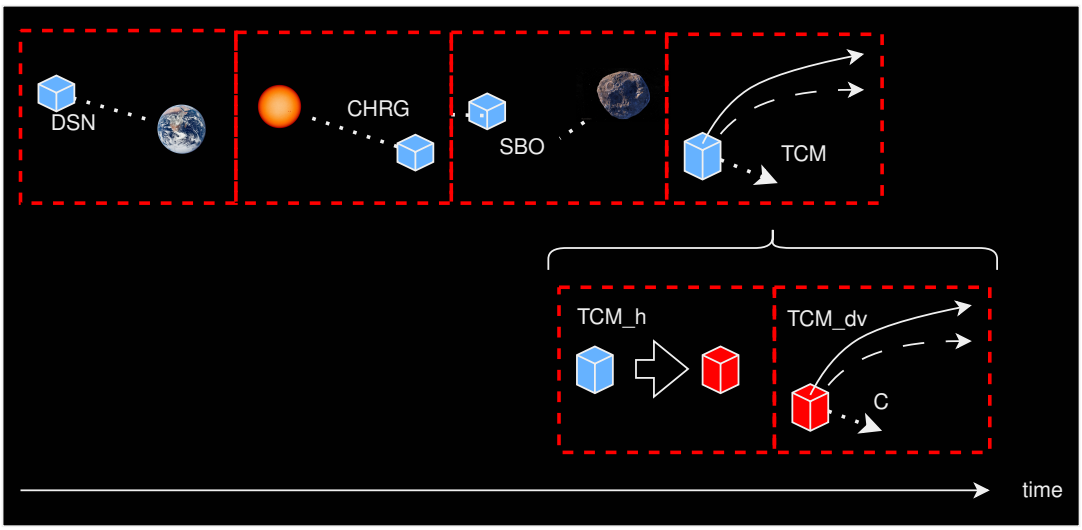


Fig. 5: Composite contract modeling a 5-step scenario sequence

instance of this task, which guarantees that the difference between the exit and entry state of charge will be proportional to the power generation rate interval constant, $[\text{pgen}^{\mathcal{T}}_{\texttt{min}\cdots\texttt{max}}]$, times the task duration input variable, $\Delta T^{\mathcal{T}}$.

Task templates for $\mathcal{T} \in \{\texttt{DSN}, \texttt{SBO}, \texttt{TCM\_h}, \texttt{TCM\_dv}\}$

CST	$\text{cons}^{\mathcal{T}}_{\texttt{min},\texttt{max}}$
In	$\text{soc}_{\texttt{entry}}, \Delta T^{\mathcal{T}}$
Out	$\text{soc}_{\texttt{exit}}$
A	$\Delta T^{\mathcal{T}} \geq 0 \;\;\&\;\; \text{soc}_{\texttt{entry}} \geq 0$
G	$\text{soc}_{\texttt{entry}-\texttt{exit}} \in [\text{cons}_{\texttt{min}\cdots\texttt{max}}]\Delta T^{\mathcal{T}}$ $\text{soc}_{\texttt{exit}} \in [0, 100]$

The constant hyperparameter, $\text{cons}^{\mathcal{T}}_{\texttt{min},\texttt{max}}$, defines the range of power consumption depleting the battery during an instance of this task, which guarantees that the difference between the entry and exit state of charge will be proportional to the power consumption rate interval constant, $[\text{cons}^{\mathcal{T}}_{\texttt{min}\cdots\texttt{max}}]$, times the task duration input variable, $\Delta T^{\mathcal{T}}$.

CST	$\text{cons}^{\texttt{DSN},\texttt{SBO},\texttt{TCM\_h},\texttt{TCM\_dv}}_{\texttt{min}\cdots\texttt{max}}, \text{pgen}^{\texttt{CHRG}}_{\texttt{min}\cdots\texttt{max}}$
In	$\text{soc}_{\texttt{entry}}, \Delta T_{\texttt{DSN},\texttt{SBO},\texttt{TCM\_h},\texttt{TCM\_dv}}$
Out	$\text{soc}^{\texttt{DSN},\texttt{SBO},\texttt{TCM\_h},\texttt{TCM\_dv}}_{\texttt{exit}}$
A	$\Delta T_{\texttt{DSN},\texttt{SBO},\texttt{TCM\_h},\texttt{TCM\_dv}} \geq 0$ $\text{soc}_{\texttt{entry}} - \text{cons}^{\texttt{DSN}}_{\texttt{max}} \Delta T_{\texttt{DSN}} \geq 0$
G	$\Delta\text{soc}^{\texttt{DSN}} \in [\text{cons}^{\texttt{DSN}}_{\texttt{min}\cdots\texttt{max}}]\Delta T_{\texttt{DSN}}$ $\text{soc}^{\texttt{CHRG}}_{\texttt{exit}} - \text{soc}^{\texttt{DSN}}_{\texttt{exit}} \in [\text{pgen}^{\texttt{CHRG}}_{\texttt{min}\cdots\texttt{max}}]\Delta T_{\texttt{CHRG}}$ $\text{soc}^{\texttt{CHRG}}_{\texttt{exit}} \leq 100$ $\text{soc}^{\texttt{CHRG}}_{\texttt{exit}} - \text{soc}^{\texttt{SBO}}_{\texttt{exit}} \in [\text{cons}^{\texttt{SBO}}_{\texttt{min}\cdots\texttt{max}}]\Delta T_{\texttt{SBO}}$ $\text{soc}^{\texttt{SBO}}_{\texttt{exit}} - \text{soc}^{\texttt{TCM\_h}}_{\texttt{exit}} \in [\text{cons}^{\texttt{TCM\_h}}_{\texttt{min}\cdots\texttt{max}}]\Delta T_{\texttt{TCM\_h}}$ $\text{soc}^{\texttt{TCM\_h}}_{\texttt{exit}} - \text{soc}^{\texttt{TCM\_dv}}_{\texttt{exit}} \in [\text{cons}^{\texttt{TCM\_dv}}_{\texttt{min}\cdots\texttt{max}}]\Delta T_{\texttt{TCM\_dv}}$

TABLE III: 5-step sequence power viewpoint composition

Composing the above for the 5-step scenario yields the contract in Table III. Notice the counter-intuitive assumption about the 1st step, DSN, requiring the scenario's initial state of charge to be greater than the worst-case consumption during the DSN step; Pacti derived this assumption from the 1st step's contract guarantees. Furthermore, although each step contract guarantees an upper bound on the state of charge, Pacti's algebraic operations effectively captured the fact that this upper bound constraint is necessary for the CHRG task and is otherwise implied for the subsequent power-consuming steps due to the chaining of the state of charge effects.

D. *Pacti contracts in the science & communication viewpoint*

Qualitatively, this viewpoint is unaffected by CHRG and TCM tasks: their contracts reduce to a no-change guarantee that the science state variables on exit are equal to those on entry. For DSN, the range of downlink rate during the task instance depletes the onboard science data storage but leaves the cumulative science data acquired unaffected. For SBO, the science data generation rate range during the task instance increases both onboard science data storage and cumulative science data acquired.

Task template for $\mathcal{T}$=DSN

CST	$[\text{rate}_{\texttt{min}\cdots\texttt{max}}]$
In	$d_{\texttt{entry}}, c_{\texttt{entry}}, \Delta T^{\mathcal{T}}$
Out	$d_{\texttt{exit}}, c_{\texttt{exit}}$
A	$\Delta T^{\mathcal{T}} \geq 0 \;\;\&\;\; d_{\texttt{entry}} \in [0, 100]$
G	$d_{\texttt{exit}-\texttt{entry}} \in [\text{rate}_{\texttt{min}\cdots\texttt{max}}]\Delta T^{\mathcal{T}}$ $c_{\texttt{exit}-\texttt{entry}} = 0$

The constant hyperparameter, $\text{rate}^{\mathcal{T}}_{\texttt{min},\texttt{max}}$, defines the range of downlink rate draining the onboard science data storage during an instance of this task, which guarantees that the difference between the exit and entry data storage will be proportional to the downlink rate interval constant, $[\text{rate}^{\mathcal{T}}_{\texttt{min}\cdots\texttt{max}}]$, times the task duration input variable, $\Delta T^{\mathcal{T}}$.

Task template for $\mathcal{T}$=SBO

CST	$\text{sgen}_{\text{min}\cdots\text{max}}$
In	$d_{\text{entry}}, c_{\text{entry}}, \Delta T^{\mathcal{T}}$
Out	$d_{\text{exit}}, c_{\text{exit}}$
A	$\Delta T^{\mathcal{T}} \geq 0 \ \ \& \ \ c_{\text{entry}} \geq 0$ $d_{entry} \in [0, 100 - \text{sgen}_{max} \Delta T_{SBO}]$
G	$d_{\text{exit}} \leq 100$ $d_{\text{exit}-\text{entry}} \in [\text{sgen}_{\text{min}\cdots\text{max}}] \Delta T^{\mathcal{T}}$ $c_{\text{exit}-\text{entry}} \in [\text{sgen}_{\text{min}\cdots\text{max}}] \Delta T^{\mathcal{T}}$

The constant hyperparameter, $\text{sgen}^{\mathcal{T}}_{\text{min,max}}$, defines the range of science data generation rate accumulating in the onboard science data storage during an instance of this task, which guarantees that the difference between the exit and entry data storage (onboard and cumulative) will be proportional to the generation rate interval constant, $[\text{sgen}^{\mathcal{T}}_{\text{min}\cdots\text{max}}]$, times the task duration input variable, $\Delta T^{\mathcal{T}}$.

Note that the TCM_h, TCM_dv tasks have trivial no-change contracts with the exit variables, u, r, equal to the corresponding entry variables. The overall science and communication viewpoint contract for the 5-step scenario yields the science and communication contract in Table IV.

CST	$\text{rate}^{\text{DSN}}_{\text{min,max}}, \text{sgen}^{\text{SBO}}_{\text{min,max}}$
In	$d_{\text{entry}}, c_{\text{entry}}, \Delta T_{\text{DSN, SBO}}$
Out	$d^{\text{DSN, CHRG, SBO, TCM\_h, TCM\_dv}}_{\text{exit}}$, $c^{\text{DSN, CHRG, SBO, TCM\_h, TCM\_dv}}_{\text{exit}}$
A	$\Delta T_{\text{DSN, SBO}} \geq 0$ $d_{\text{entry}} \in [0, 100]$
G	$d_{\text{exit}} \leq 100$ $d^{\text{DSN}}_{\text{entry}-\text{exit}} \in [\text{rate}_{\text{min}\cdots\text{max}}] \Delta T_{\text{DSN}}$ $d^{\text{SBO}}_{\text{exit}-\text{entry}} \in [\text{sgen}_{\text{min}\cdots\text{max}}] \Delta T_{\text{SBO}}$ $c^{\text{SBO}}_{\text{exit}-\text{entry}} \in [\text{sgen}_{\text{min}\cdots\text{max}}] \Delta T_{\text{SBO}}$ $d^{\text{TCM\_h, TCM\_dv}}_{\text{exit}} = d^{\text{SBO}}_{\text{exit}}$ $c^{\text{TCM\_h, TCM\_dv}}_{\text{exit}} = c^{\text{SBO}}_{\text{exit}}$

TABLE IV: 5-step sequence science & communication viewpoint composition

E. Pacti contracts in the navigation viewpoint

Qualitatively, DSN and CHRG have similar impacts where the trajectory estimation uncertainty increases according to a noise range due to a change of spacecraft orientation performed during an instance of such tasks. Thanks to optimal measurements of the asteroid performed during this task, the onboard auto-navigation software can reduce the trajectory estimation uncertainty within some range of improvement. TCM_h has no impact on uncertainty. All three tasks leave the relative trajectory distance unchanged. Due to performing a long-duration change of velocity, the TCM_dv task injects additional trajectory estimation uncertainty proportional to a noise range; however, it reduces the relative trajectory distance proportional to an improvement range.

Task template for $\mathcal{T} \in \{$DSN, CHRG$\}$

CST	$\text{noise}_{\text{min}\cdots\text{max}}$
In	$u_{\text{entry}}, r_{\text{entry}}$
Out	$u_{\text{exit}}, r_{\text{exit}}$
A	$u_{\text{entry}} \in [0, 100] \ \ \& \ \ r_{\text{entry}} \in [0, 100]$
G	$r_{\text{exit}} = r_{\text{entry}} \ \ \& \ \ u_{exit} \leq 100$ $u_{\text{exit}-\text{entry}} \in [\text{noise}_{\text{min}\cdots\text{max}}]$

The constant hyperparameter, $\text{noise}_{\text{min}\cdots\text{max}}$, defines the range of trajectory estimation uncertainty noise injected in an instance of this task due to a single change of spacecraft orientation.

Task template for $\mathcal{T}$=SBO

CST	$\text{imp}_{\text{min}\cdots\text{max}}$
In	$u_{\text{entry}}, r_{\text{entry}}, \Delta T^{\mathcal{T}}$
Out	$u_{\text{exit}}, r_{\text{exit}}$
A	$\Delta T^{\mathcal{T}} \geq 0 \ \ \& \ \ u_{\text{entry}} \leq 100$
G	$r_{\text{exit}} = r_{\text{entry}} \ \ \& \ \ u_{exit} \in [0, 100]$ $u_{\text{exit}-\text{entry}} \in [\text{imp}_{\text{min}\cdots\text{max}}] \Delta T^{\mathcal{T}}$

The constant hyperparameter, $\text{imp}_{\text{min}\cdots\text{max}}$, defines the range of trajectory estimation uncertainty improvement during an instance of this task due to onboard autonomous navigation calculations, which guarantees that the difference between the exit and entry uncertainty will be proportional to the improvement rate interval constant[§], $[\text{imp}_{\text{min}\cdots\text{max}}]$, times the task duration input variable, $\Delta T^{\mathcal{T}}$.

Note that the TCM_h task has a trivial no-change contract with the exit variables, u, r, equal to the corresponding entry variables.

Task template for $\mathcal{T}$=TCM_dv

CST	$\text{imp}_{\text{min}\cdots\text{max}}, noise_{\text{min}\cdots\text{max}}$
In	$u_{\text{entry}}, r_{\text{entry}}, \Delta T^{\mathcal{T}}$
Out	$u_{\text{exit}}, r_{\text{exit}}$
A	$\Delta T^{\mathcal{T}} \geq 0 \ \ \& \ \ u_{\text{entry}} \in [0, 100] \ \ \& \ \ r_{entry} \leq 100$
G	$r_{\text{exit}} \geq 0 \ \ \& \ \ u_{\text{exit}} \in [0, 100]$ $r_{\text{exit}-\text{entry}} \in [\text{imp}_{\text{min}\cdots\text{max}}] \Delta T^{\mathcal{T}}$ $u_{\text{exit}-\text{entry}} \in [\text{noise}_{\text{min}\cdots\text{max}}] \Delta T^{\mathcal{T}}$

The constant hyperparameter, $\text{imp}_{\text{min}\cdots\text{max}}$, defines the range of relative trajectory progress improvement during an instance of this task due to onboard autonomous navigation calculations, which guarantees that the difference between the exit and entry progress will be proportional to the improvement rate interval constant, $[\text{imp}_{\text{min}\cdots\text{max}}]$, times the task duration input variable, $\Delta T^{\mathcal{T}}$. The constant hyperparameter, $\text{noise}_{\text{min}\cdots\text{max}}$, defines the range of trajectory estimation uncertainty degradation during an instance of this task due to velocity change being performed, which guarantees that the difference between the exit and entry uncertainty will be proportional to the noise interval constant, $[\text{noise}_{\text{min}\cdots\text{max}}]$, times the task duration input variable, $\Delta T^{\mathcal{T}}$.

Composing the above for the 5-step scenario yields the navigation contract in Table V.

[§]An improvement interval with a negative lower bound corresponds to the possibility of a navigation trajectory deterioration.

CST	$\text{noise}^{\texttt{DSN,CHRG,TCM\_dv}}_{\text{min,max}}, \text{imp}^{\texttt{SBO,TCM\_dv}}_{\text{min,max}}$
In	$u_{\texttt{entry}}, r_{\texttt{entry}}, \Delta T_{\texttt{SBO,TCM\_dv}}$
Out	$u^{\texttt{DSN,CHRG,SBO,TCM\_h,TCM\_dv}}_{\texttt{exit}}$, $r^{\texttt{DSN,CHRG,SBO,TCM\_h,TCM\_dv}}_{\texttt{exit}}$
A	$\Delta T_{\texttt{SBO,TCM\_dv}} \geq 0$ $u_{\texttt{entry}} \in [0,100]$ & $r_{\texttt{entry}} \in [0,100]$
G	$u^{\texttt{DSN}}_{\texttt{exit}-\texttt{entry}} \in [\text{noise}^{\texttt{DSN}}_{\text{min}\cdots\text{max}}]$ $r^{\texttt{DSN}}_{\texttt{exit}} = r_{\texttt{entry}}$ $u^{\texttt{CHRG}}_{\texttt{exit}} - u^{\texttt{DSN}}_{\texttt{exit}} \in [\text{noise}^{\texttt{CHRG}}_{\text{min}\cdots\text{max}}]$ $u^{\texttt{CHRG}}_{\texttt{exit}} \leq 100$ $u^{\texttt{CHRG}}_{\texttt{exit}} - u^{\texttt{SBO}}_{\texttt{exit}} \in [\text{imp}^{\texttt{SBO}}_{\text{min}\cdots\text{max}}]\Delta T_{\texttt{SBO}}$ $u^{\texttt{SBO}}_{\texttt{exit}} \geq 0$ & $r^{\texttt{SBO}}_{\texttt{exit}} = r^{\texttt{CHRG}}_{\texttt{exit}}$ $u^{\texttt{TCM\_h}}_{\texttt{exit}} = r^{\texttt{SBO}}_{\texttt{exit}}$ & $r^{\texttt{TCM\_h}}_{\texttt{exit}} = r^{\texttt{SBO}}_{\texttt{exit}}$ $u^{\texttt{TCM\_dv}}_{\texttt{exit}} - u^{\texttt{TCM\_h}}_{\texttt{exit}} \in [\text{noise}^{\texttt{TCM\_dv}}_{\text{min}\cdots\text{max}}]\Delta T_{\texttt{TCM\_dv}}$ $r^{\texttt{TCM\_dv}}_{\texttt{exit}} - r^{\texttt{TCM\_h}}_{\texttt{exit}} \in [\text{imp}^{\texttt{TCM\_dv}}_{\text{min}\cdots\text{max}}]\Delta T_{\texttt{TCM\_dv}}$

TABLE V: 5-step sequence navigation viewpoint composition

IV. Schedulability Analysis

Our schedulability analysis methodology reflects separating design from operation concerns. Design concerns correspond to the capability characteristics of each task:

- range of power consumption for each of `DSN`, `SBO`, `TCM_h`, `TCM_dv` tasks.
- range of power generation for the `DSN` task.
- min,max range of downlink speed for the `DSN` task.
- range of science data acquisition rate for the `SBO` task.
- range of trajectory estimation uncertainty noise injection for each of `DSN`, `CHRG`, `TCM_dv` tasks.
- range of trajectory estimation uncertainty improvement due to optimal small body measurements for the `SBO` task.
- range of relative trajectory progress for the `TCM_dv` task.

We defined these capability characteristics as contract hyperparameters as discussed in Section III-A. On the other hand, we defined operational requirements as constraints on entry/exit variables:

- Minimum battery state of charge; range: 60-90
- Minimum task duration for each step; range: 10-50 seconds
- Initial science data volume; range: 60-100
- Initial trajectory estimation uncertainty: 40-90

Methodologically, we defined schedulability as the compatibility between a schedule (based on a given choice of capability hyperparameters) and a set of operational requirements (based on a given choice of values for entry/exit variables). We applied the Latin hypercube statistical sampler to generate multiple combinations of scenarios and operational requirements as summarized in Table VI using a Windows 10 workstation powered by an AMD Threadripper Pro 3955WX processor with 16 cores and 128GB RAM running Ubuntu 20.04 under Windows 10's WSL2[§]. For scenario generation, we sampled 200 distributions to generate mean and deviation for specifying the range of each of the 12 capability hyperparameters. The second column shows the statistics for producing a short 5-step scenario and a long 20-step scenario given such a sample. For varying operational requirements, we generated 100 random values within predefined ranges of requirement constraints. We computed schedulability using Pacti's merge operation for all combinations of 200 scenarios and 100 operational requirements. The third column shows the statistics for this schedulability analysis. The scarcity of admissible solutions (i.e. less than 1%) and the efficiency of schedulability analysis[§] demonstrates the usefuless of Pacti for rapid exploration of design and operational constraints.

Pacti's API provides additional capabilities to get useful insights into admissible schedules. For example, Figures 6a and 6b show two results of the visualizing the bounds of battery state-of-charge at the entry and exit of each step in the schedule using Pacti's `get_variable_bounds()` API. Note that this range visualization is qualitatively different from a simulation timeline: a single value over time. These figures also illustrate a subtle aspect of Pacti's polyhedral contract algebra where the effect of composing the sequence step contracts results in relaxing the guarantees [5, Sec. 4], broadening the possible exit variable ranges since the final variables are unconstrained. Conversely, the middle section of the scenario shows greater precision in the bound calculations since the subsequent contracts force constraints on the exit variables, thereby preventing their relaxation. Aside from the subtleties of contract relaxation, these two figures show a stark contrast between different scenario characteristics and operational requirements combinations. Such differences would compellingly motivate cross-validating the Pacti contract approximations with appropriate simulation models to get additional insights into these differences.

With Pacti's `optimize()` API, we computed the minimum and maximum values of a linear optimization metric, the average of all states of charges at the end of each step, and plotted these results in Figure 7 by scoring each admissible schedule w.r.t the scenario and operational requirement. For scenario scoring, we took the average of adding/subtracting all viewpoint-specific positive/negative capabilities: generation vs. consumption for the power viewpoint, downlink speed vs. observation rate for the science viewpoint, and noise vs. improvement for the navigation viewpoint. For operational requirement scoring, we averaged all constraints since the difficulty of achieving them increases with their magnitude. The clustering of admissible schedules suggests that designers could get more insight by performing this scoring on specific viewpoints instead of combining them as was done here.

V. Related Work

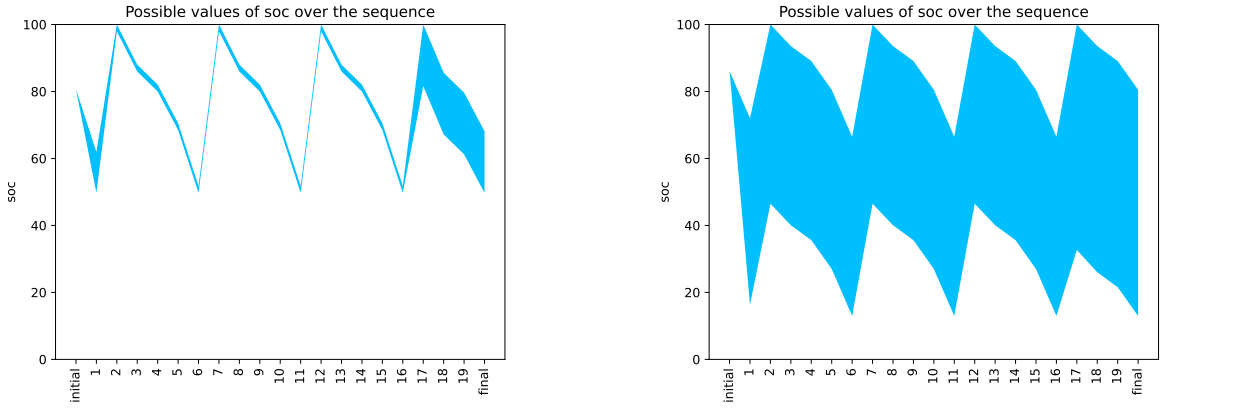
We review previous work in several areas related to the modeling and analysis approach presented here. From the

[§]For details about performance measurement and API statistics, see https://github.com/pacti-org/pacti-instrumentation.

[§]Over 100 combinations per second (5-step scenario) and over 25 combinations per second (20-step scenario) using up to 32 concurrent jobs.

Scenario	Pacti operations for scenario generation	Pacti operations for schedulability analysis
5-step	compose: 2016, up to 22 constraints, 12 variables merge: 1680, up to 44 constraints, 23 variables total time: 16.3 seconds	merge: 181,988, up to 81 constraints, 35 variables admissible solutions: 401 out of 20,000 combinations total time: 3 minutes, 9.6 seconds
20-step	compose: 10,200, up to 187 constraints, 95 variables merge: 8,000, up to 44 constraints, 23 variables total time: 97.8 seconds	merge: 781,331, up to 275 constraints, 125 variables admissible solutions: 244 out of 20,000 combinations total time: 4 minutes, 11 seconds

TABLE VI: Scenario generation (100 hyperparameter samples) & schedulability analysis (100 operational requirement samples)



(a) Low-variability schedule with high min. soc levels

(b) High-variability schedule with potentially low soc levels

Fig. 6: Examples of schedulability analysis results

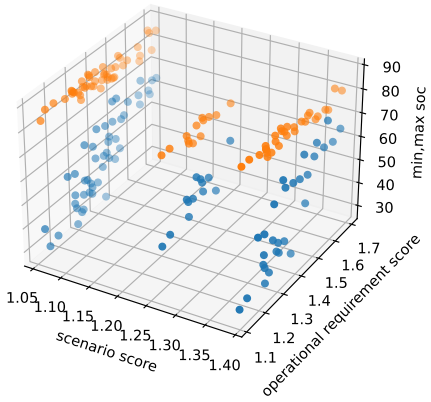


Fig. 7: Score-based visualization of the min (blue) and max (orange) battery state of charge over all admissible schedules

modeling standpoint, several languages have been defined in the past to specifically deal with sequences of operations. Most notably, the AI planning community uses the Planning Domain Definition Language (PDDL) [10], including extensions for hierarchical planning [11]. These languages allow users to model state transitions as a result of actions and focus on succinctness and efficiency for the purpose of planning. On the other hand, the contract framework in Pacti focuses on system and sub-system modeling from different viewpoints, and it is expressive enough to encode state transitions. The contract framework focuses on concurrent, component-based engineering of complex systems. Another important distinction is in the difference between assumptions in contracts and preconditions in PDDL. Actions are only applicable if their preconditions are satisfied, while the assumptions of a component can be violated (albeit at the price of the component not satisfying its guarantees).

Planning/scheduling systems designed for embedded systems impose significant restrictions on the expressiveness of the planning language to ensure responsiveness in limited computing environments. In [7], [8] the authors describe a generalized timeline representation restricting planning constraints to a single variable linear constraint. In contrast, Pacti supports linear constraints with multiple variables as shown in the case study above.

The work presented in this paper also differs from classical planning and scheduling. In planning, we are given a set of task models, an initial state, and a goal state; the problem is to find a sequence of task instances that bring a system from the initial state to the goal state [12]. Given the complexity of the general problem, planning algorithms are typically specialized in solving problems in restricted domains that allow the development of efficient heuristics. However, developing the task models (also referred to as domain authoring) is a major contributor to efficiency, and verification and validation of such models are hard and time-consuming activities [13]. In our work, we address the joint exploration of system requirements allocation and domain authoring which then leads to high-quality planning domains.

Finally, several frameworks for contract-based modeling and verification such as AGREE [14] and OCRA [15] are available. Assumptions and guarantees are specified in a synchronous language for the former, and Linear Temporal Logic for the latter. These systems allow users to instantiate and connect components and to check whether such composition refines a higher-level contract. However, unique to Pacti is the explicit support for multiple viewpoints and the ability to compute the result of composition in the form of a new contract at the interface of the system. This capability provides the actual expression of the contract implemented by the composition of components. Moreover, Pacti implements algorithms that can compute the quotient of a specification with respect to a partial system to obtain the specification of a missing component.

VI. Concluding remarks

The ability to analyze complex space science missions early on in the design cycle is essential to mission success. In the early stages of design, the development of requirements and their allocation to sub-systems involves a collaborative and iterative effort. Tool support would be beneficial to evaluate trade-offs among multiple viewpoints with respect to the system design and its planned operation.

Considering this objective, we presented an agile methodology for designing and analyzing such systems across multiple viewpoints based on a compositional, contract-based modeling paradigm using Pacti [5]. Pacti supports the theory of polyhedral constraints for specifying assume-guarantee contracts. Since this formalism involves linear constraints for specifying assumptions and guarantee constraints, the modeling paradigm should be accessible to most stakeholders and appropriate for communicating across stakeholders with diverse expertise and viewpoints the intricacies of space mission design, requirements, and operations formulated in this manner.

We demonstrated the scalability of several of Pacti's API operations for composing and merging contracts and for computing bounds for variables and linear optimization criteria. The performance benchmark results we presented confirm not only the scalability and efficiency of Pacti's novel algorithms [5] but also provide a compelling case for applying this methodology to complex space mission design and operation problems. We collected a list of lessons learned from this experiment. Mainly, Pacti is very good at combining contracts, whether these are for system or component specification purposes or for operational requirement purposes. Despite Polyhedra algebra being mathematically simple, complex Polyhedra contracts can be difficult to understand. In these cases, we found that computing minimum and maximum bounds for contract variables yields valuable information for elaborating operational requirements. In engineering, behavior is typically thought in terms of simulating the state variables as a function of time. Pacti requires systems engineers to rethink about behavior as bounds on steps; on the other hand, Pacti's contract algebra provides powerful tools to help understand bounded behavior. For example, developing a time-based simulation model involves a risk that one could get lost in the details and lose track of the overall modeling objective. In contrast, Pacti's polyhedral contract algebra coerces the system engineer to think about which aspects of behavior are important to characterize with bounds.

Acknowledgments

JPL/Caltech Copyright: © 2023 California Institute of Technology. Government sponsorship acknowledged. The research was carried out at the Jet Propulsion Laboratory, California Institute of Technology, under a contract with the National Aeronautics and Space Administration (80NM0018D0004). This work was partially supported by the JPL Researcher On Campus (JROC) program for which we gratefully acknowledge Prof. Richard M. Murray's support. This work was also partially supported by NSF and ASEE through an eFellows postdoctoral fellowship.

References

[1] S. Planning, "The economic impacts of inadequate infrastructure for software testing," *National Institute of Standards and Technology*, vol. 1, 2002.

[2] A. Sangiovanni-Vincentelli, W. Damm, and R. Passerone, "Taming Dr. Frankenstein: Contract-Based Design for Cyber-Physical Systems," *European Journal of Control*, vol. 18, no. 3, pp. 217 – 238, 2012.

[3] A. Benveniste, B. Caillaud, D. Nickovic, R. Passerone, J.-B. Raclet, P. Reinkemeier, A. Sangiovanni-Vincentelli, W. Damm, T. A. Henzinger, and K. G. Larsen, "Contracts for system design," *Foundations and Trends® in Electronic Design Automation*, vol. 12, no. 2-3, pp. 124–400, 2018.

[4] I. Incer, *The Algebra of Contracts*. PhD thesis, EECS Department, University of California, Berkeley, May 2022.

[5] I. Incer, A. Badithela, J. Graebener, P. Mallozzi, A. Pandey, S.-J. Yu, A. Benveniste, B. Caillaud, R. M. Murray, A. Sangiovanni-Vincentelli, *et al.*, "Pacti: Scaling assume-guarantee reasoning for system analysis and design," *arXiv preprint arXiv:2303.17751*, 2023.

[6] I. A. Nesnas, B. J. Hockman, S. Bandopadhyay, B. J. Morrell, D. P. Lubey, J. Villa, D. S. Bayard, A. Osmundson, B. Jarvis, M. Bersani, *et al.*, "Autonomous exploration of small bodies toward greater autonomy for deep space missions," *Frontiers in Robotics and AI*, p. 270, 2021.

[7] S. Chien, "A generalized timeline representation, services, and interface for automating space mission operations," in *SpaceOps 2012*, p. 1275459, 2012.

[8] G. Rabideau, S. Chien, M. Galer, F. Nespoli, and M. Costa, "Managing spacecraft memory buffers with concurrent data collection and downlink," *Journal of Aerospace Information Systems*, vol. 14, no. 12, pp. 637–651, 2017.

[9] Y. Chen and B. W. Wah, "Automated planning and scheduling using calculus of variations in discrete space.," in *ICAPS*, pp. 2–11, 2003.

[10] M. Fox and D. Long, "PDDL2.1: An extension to PDDL for expressing temporal planning domains," *Journal of artificial intelligence research*, vol. 20, pp. 61–124, 2003.

[11] I. Georgievski and M. Aiello, "HTN planning: Overview, comparison, and beyond," *Artificial Intelligence*, vol. 222, pp. 124–156, 2015.

[12] M. Ghallab, D. Nau, and P. Traverso, *Automated Planning: theory and practice*. Elsevier, 2004.

[13] T. L. McCluskey, T. S. Vaquero, and M. Vallati, "Engineering knowledge for automated planning: Towards a notion of quality," in *Proceedings of the Knowledge Capture Conference*, pp. 1–8, 2017.

[14] D. Cofer, A. Gacek, S. Miller, M. W. Whalen, B. LaValley, and L. Sha, "Compositional verification of architectural models," in *NASA Formal Methods* (A. E. Goodloe and S. Person, eds.), (Berlin, Heidelberg), pp. 126–140, Springer Berlin Heidelberg, 2012.

[15] A. Cimatti, M. Dorigatti, and S. Tonetta, "OCRA: A tool for checking the refinement of temporal contracts," in *2013 28th IEEE/ACM International Conference on Automated Software Engineering (ASE)*, pp. 702–705, 2013.