APRES Prototype Mission Planner System Demonstration

John L. Bresina[1], Vijayakumar Baskaran[2], J Benton[3], Hassan Eslami[4], Elif Kurklu[5], David E. Smith[6], Deep Tailor[7], and Ramazan Ushpayev[8]

NASA Ames Research Center[1,3,] KBR Wyle Services, LLC[2,4,5,7,8]
john.l.bresina@nasa.gov[1], vijayakumar.baskaran-1@nasa.gov[2], j.benton@nasa.gov[3], hassan.eslami@nasa.gov[4], elif.kurklu@nasa.gov[5], david.smith@psresearch.xyz[6], ramazan.ushpayev@nasa.gov[7]

Abstract

Activity Planning with Resources for the Exploration of Space (APRES) is a mixed-initiative mission planning system for ground operations. APRES has been designed to support multi-spacecraft missions. The APRES Interface is browser-based and includes a plan editor, a timeline plan display, a temporal constraint editor, display of the state and numeric chronicles, and a violation resolution manager. Automation support is supplied by the APRES Service, which includes components that provide the following capabilities:(1) plan simulation, which determines the state and numeric chronicles (values of the model variables over time) and determines when "processes" are triggered and terminated based on world states in the execution trace, (2) violation detection of constraints and flight rules encoded in the domain model, and of the temporal constraints created by the user, (3) violation resolution suggestions as to how to fix the plan's violations via rescheduling. The user controls when and how to utilize the automation support.

Introduction

The APRES (Activity Planning with Resources for the Exploration of Space) Prototype system is a ground-based, mixed-initiative mission planner, which can be employed for human space missions and robotic missions. The support of multi-spacecraft missions was one of the key design drivers. The resulting features facilitate the creation of multi-spacecraft domain models, enabling an order of magnitude reduction in the model size, likewise for the creation of the UI configuration.

The APRES Prototype design draws primarily from the design of, and operational experience with, the LASS planner (based on SPIFe) deployed on the Lunar Atmosphere Dust Experiment Explorer (LADEE). The primary differences between APRES and LASS are the interface framework, the domain modelling language, and the automated reasoning components. LASS used an Eclipse-based interface, a simpler Activity Dictionary, and a reasoning component based on EUROPA (Frank and Jonsson, 2003).

APRES Architecture Components

The APRES Prototype architecture consists of the following key components (Figure 1).

- **APRES Interface:** browser-based front-end, built on top of the OpenMCT (Mission Control Technologies) ground operations software system (https://nasa.github.io/openmct/).
- **APRES Service:** back-end suite of file management and automated reasoning components
- **ANML Editor:** browser-based smart editor for domain models, specified in the ANML language
- **APRES Data Store:** file-based local storage for all files used in the planning process; accessed by both APRES and the ANML Editor

The APRES Interface includes the following GUI components: Activity Dictionary, Activity Editor, Timeline Viewer, Temporal Constraint Editor, and Tables. The Timeline Viewer includes the UTC Time Zone, Action/Process Timelines, and State and Numeric Chronicles. There are tables for: Initial Assignments, Violations, Temporal Constraints, and Resolutions. From the interface, the user can invoke the Validate Plan operation and the Resolve Violations operation, both of which are performed by the APRES Service.

The APRES Service components are based on the ANML language (Smith, Frank, and Cushing, 2008), which is a highly expressive language for specification of models. ANML enables the creation of more accurate models and more accurate plans. In addition to action definitions, ANML models can include definitions of "processes", which are not under the control of the agent, e.g., the operation of a survival heater that is powered on and off based on the current temperature.

The closest modelling language to ANML is PDDL, which has many variants (for a summary see https://en.wikipedia.org/wiki/Planning_Domain_Definition_Language). ANML has strong notions of action and

state, much like in PDDL; however, ANML uses a variable/value representation and one can specify a richer set of possible conditions and effects than allowed by PDDL. In particular, one can specify conditions at times other than the start and end of an action, and also over arbitrary intervals within the action. Similarly, one can specify effects at times other than the start and end of an action. The same primitives that are used to specify these richer temporal constraints in ANML are also used to specify exogenous conditions, as is done with timed initial literals in PDDL 2.2. In contrast to PPDL+, ANML does not distinguish between events and processes – processes are allowed to include both continuous and discrete effects at arbitrary times and over arbitrary windows.

We have also developed a web based ANML Editor for creating domain models. The ANML Editor interacts with the ANML Parser to detect errors and interacts with the APRES Data Store for file management.

The APRES Planning Process

The human planner selects which action instances to insert into the plan and schedules them. The automated reasoning components in the APRES Service provide support to the user. Typically, APRES is used to *incrementally* develop a plan; the user alternates between adding actions to extend the plan and invoking the simulator and violation detection process. In order to handle partial plans, the Episodic Plan Simulator (EpSim) is *permissive* and continues the simulation in the face of violations by making enabling assumptions about missing preconditions, conflicting effects, and bounds violations. This gives the user a more complete status of the partial plan and facilitates the violation resolution process.

The simulation determines the *chronicles* for each fluent (variable) in the domain model, specifying the fluent's value over time. Secondly, the simulation also inserts into the plan the process instances that are triggered by simulation episodes. Thirdly, the plan validation detects plan violations based on the domain model and the user-created temporal constraints. The following are the types of violations detected: Unsatisfied Condition, Violated Condition, Inconsistent Effects, Variable Bounds, Temporal Constraint, and Inconsistent Constraints. The first four violation types are detected by EpSim, and the last two types are detected by the Temporal Constraint Checker.

Upon request from the user, suggestions for resolving the existing plan violations are automatically generated, which specify changes to the schedule of actions. The user has control over which if any of these resolution suggestions to perform and which violations to resolve manually via action rescheduling, addition, modification, or deletion.

Any subset of the recommendations can be selected and previewed, showing what the plan would look like and what violations would remain. The user can then select another subset to preview or accept the current resolved plan to replace the original plan or reject all resolution suggestions and manually fix the violations.

Acknowledgments

This work was performed as part of the NASA Autonomous Systems and Operations (ASO) Project, under the Advanced Exploration Systems Program.

References

Bresina, J.L., Activity Planning for a Lunar Orbital Mission. AI magazine, Vol. 28, No. 2, Summer 2007.

Frank, J., and Jonsson, A., Constraint-Based Interval and Attribute Planning, Journal of Constraints Special Issue on Constraints and Planning, 2003.

Smith, D.E., Frank, J., and Cushing, W., The ANML Language, ICAPS Workshop on Knowledge Engineering for Planning and Scheduling (KEPS), 2008.

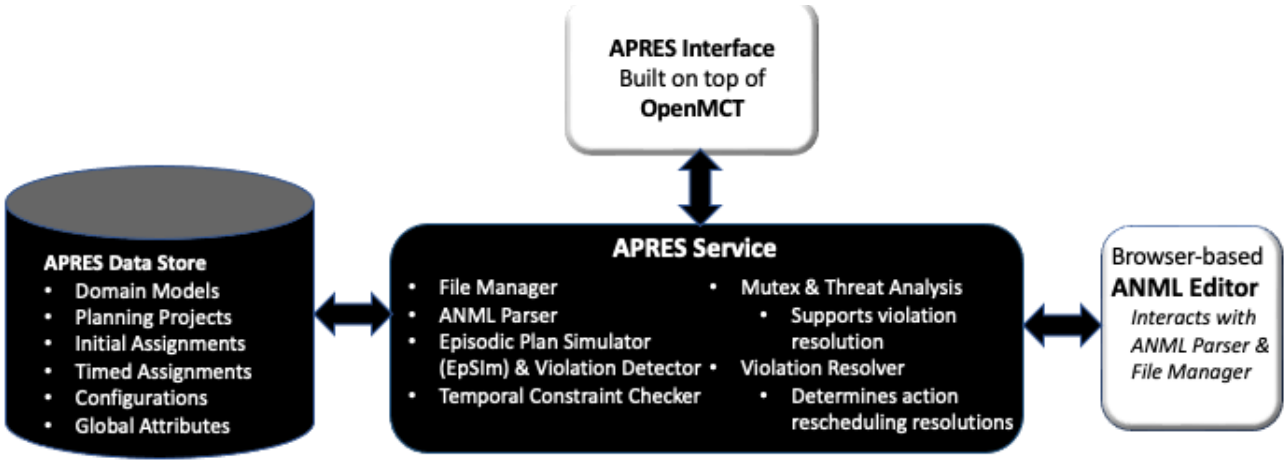

Figure 1: APRES Prototype System Architecture.