The Hierarchical Satellite Domain

Bernd Schattenberg[1]
3B intelligent solutions, Germany, schattenberg@3b-intelligent-solutions.com
[1] The domain was created while still being at the Institute of Artificial Intelligence of Ulm University

Abstract

The *Satellite* domain is one of the classical benchmark domains in the canon of the International Planning Competition. This paper describes our hierarchical take on it.

Introduction

The hierarchical *Satellite* domain is inspired by space applications that are a first step towards the "Ambitious Spacecraft" as described by David Smith at the AIPS 2000 conference (Smith, Frank, and Jónsson 2000). It involves planning a set of stellar observation tasks for multiple autonomous satellites, each equipped with slightly different but possibly overlapping technologies. The equipment consists of observation instruments with different characteristics in terms of data productions, so-called *modes* like thermal images, x-ray, etc., for which corresponding calibration targets are defined. Satellites are motile and can be oriented towards arbitrary stellar target objects by slewing the complete platform between different attitudes/directions. A benchmark problem in this domain is consequently given by an initial state that describes the satellite configurations and stellar phenomena positions, while the goal state specifies of which observation targets an image has to be taken in which mode.

The *Satellite* domain has been introduced as a classical benchmark to the planning community in the 2002 installment of IPC. We have developed a hierarchical version of it in order to analyse planning strategy designs for hybrid planning systems (Schattenberg, Weigl, and Biundo 2005; Schattenberg, Bidot, and Biundo 2007), from which the presented, purely hierarchical version has been derived.

This document focuses on the design decisions that led to the hybrid planning domain model for the formal framework introduced in (Biundo and Schattenberg 2001; Schattenberg 2009), i.e., on adding hierarchical features to a non-hierarchical domain model (cf. (Pragst et al. 2014)).

Types and Relations

The first step in translating the original *Satellite* domain into a hierarchical formalism is to introduce a type hierarchy. While the PDDL encoding already defined the types for satellites, directions, instruments, and (image) modes, we think that this does not capture an essential feature of the problem instances: the defined directions are obviously divided into the actual observation phenomena that are of scientific interest and attitude points that are only used for calibration purposes. This aspect is incorporated in the type hierarchy by providing a general purpose `Direction` as super-type for `Calib_Direction` and `Image_Direction`.

In our formal hybrid-planning framework, we also annotated the direction super type to be *abstract*, i.e. a conceptual type for which no constant declaration is allowed. The rationale for such a language feature is to identify types that are intended to be exclusively used for structuring the application domain concepts, thus supporting modelling tools to validate problem and domain consistency.

When it comes to specifying relation symbols for expressing predicate sentences or facts about the world state, HDDL and PDDL models explicitly provide the relation symbols' signatures in its declaration header. However, they do not explicitly denote whether actions (are allowed to) manipulate the respective attributes, or, in more formal terms, they do not discriminate flexible and rigid relations. Instead, state-invariant features are typically extracted from the domain model during pre-processing.

The original model's documentation plus some common sense suggest to adhere to the following partitioning of relation symbol declarations: Flexible relations are

- $\texttt{pointing}_{\texttt{Satellite,Direction}}$
- $\texttt{power\_avail}_{\texttt{Satellite}}$
- $\texttt{have\_image}_{\texttt{Image\_Direction,Mode}}$
- $\texttt{power\_on}_{\texttt{Instrument}}$
- $\texttt{calibrated}_{\texttt{Instrument}}$

The `pointing` relation is used for expressing that a satellite platform (the first argument in any atom over this relation), and with it all on-board instruments, aim at a given direction (the second argument of such atoms). Slewing the satellite therefore implicitly controls the orientation of the desired instrument as well. `power_avail` and `power_on` reflect that energy is a limited resource on the observation platform and that therefore only one instrument can be served at a time. On-board observation systems typically have to take reference images for calibrating the sensors and if an instrument is ready for taking images, its status changes to `calibrated`. In a state in which the image

of a phenomenon is finally taken a respective atom over have_image is supposed to hold.

The relations representing state-invariant facts are:

- $\texttt{on\_board}_{\texttt{Instrument,Satellite}}$
- $\texttt{supports}_{\texttt{Instrument,Mode}}$
- $\texttt{calibration\_target}_{\texttt{Instrument,Calib\_Direction}}$

This includes the relation for modelling which instruments which satellite carries, which kind of sensor the instrument provides, and what the reference object for calibrating a given instrument is.

Actions, Tasks, and Methods

The action specifications can be directly taken from the original non-hierarchical PDDL description:

- $\texttt{turn\_to}_{\texttt{Satellite,Direction,Direction}}$
- $\texttt{switch\_on}_{\texttt{Instrument,Satellite}}$
- $\texttt{switch\_off}_{\texttt{Instrument,Satellite}}$
- $\texttt{calibrate}_{\texttt{Satellite,Instrument,Calib\_Direction}}$
- $\texttt{take\_image}_{\texttt{Satellite,Image\_Direction,Instrument Mode}}$

The intended meaning of these five operator signatures be self explanatory. The corresponding action definitions basically implement conditional switching operations for the intended state feature, e.g.:

```
(:action switch_on
   :parameters (?so_i - instrument
                ?so_s - satellite)
   :precondition
      (and (on_board ?so_i ?so_s)
           (power_avail ?so_s))
   :effect
      (and (power_on ?so_i)
           (not (calibrated ?so_i))
           (not (power_avail ?so_s))))
```

It is the operator that routes energy to a given instrument on the observation platform, it switches power from available to not available. In this way, no two instruments can be used in parallel on one satellite. The second precondition for the action assures the required instrument to be on board the given satellite, which merely enforces a consistent binding of the two parameters. Please note that although this particular style of modeling introduces a considerable amount of redundancy in the *Satellite* action definitions, any processing that is aware of the underlying state-invariance will reduce the unnecessary branching at this point.

When we analyzed the IPC benchmark problems for this domain, it occurred to us that there obviously exists an *intended procedure* for taking satellite images and that all solutions follow that pattern with minor deviations: Making an observation for a given sensor mode and phenomenon firstly consists of choosing a suitable instrument, which in turn indirectly determines the satellite that performs the observation. In a second step, the instrument has to be routed energy to and properly calibrated. The satellite finally slews in the direction of the target phenomenon and takes the image.

This procedure is plausible enough to be considered not as a specification artefact that has been accidentally introduced by the competition initiators but as an underlying principle in the *Satellite* domain and consequently a clue for a well-reasoned action abstraction. An apparent structure is to build an abstraction for each of the two phases: preparing the instrument and taking the picture becomes an abstract task do_observation with parameters for the desired phenomenon to observe and the mode to support. The preparation phase seems to require an abstraction hierarchy on its own, in order to encapsulate the different ways of getting the sensory system on-line (the instrument is already on and calibrated, some other instrument has to be turned off first in order to raise the energy level properly, etc.). We therefore introduced an abstract action for activating the instrument and for dealing with the calibration. The resulting (complex) tasks are consequently the following three:

- $\texttt{do\_observation}_{\texttt{Image\_Direction,Mode}}$
- $\texttt{activate\_instrument}_{\texttt{Satellite,Instrument}}$
- $\texttt{auto\_calibrate}_{\texttt{Satellite,Instrument}}$

We intended to define the complex task schemata in the fashion of ABSTRIPS operator reductions (Sacerdoti 1974). That means, we do not employ state abstraction axioms (cf. (Biundo and Schattenberg 2001)) but simply generalize the preconditions and effects of the primitive implementations, like in the following example:

```
(:task activate_instrument
   :parameters (?ai_s - satellite
                ?ai_i - instrument)
   :precondition
      (and (on_board ?ai_i ?ai_s))
   :effect (and (power_on ?ai_i)))
```

Given these complex and primitive tasks, the methods of the domain model set up a decomposition hierarchy that implements the different observation procedures as described above. Please note that this decomposition hierarchy does not impose semantic restrictions on the solution space.

The following method method0 implements an observation by sequentially activating the instrument, turning the satellite, and taking the image:

```
(:method method0
 :parameters
  (?d_prev - direction  ?sat - satellite
   ?d_im - image_direction
   ?i - instrument      ?mode - mode)
 :task
  (do_observation ?d_im ?mode)
 :subtasks (and
  (task0 (activate_instrument ?sat ?i))
  (task1 (turn_to ?sat ?d_im ?d_prev))
  (task2 (take_image ?sat ?d_im ?i ?mode)))
 :ordering (and (task0 < task1)
                (task1 < task2))
 :constraints (and
  (sortof ?d_im - image_direction)
  (not (= ?d_im ?d_prev))))
```

The parameters section introduces all variable names used for task node parameters and variable constraints. By binding the same variable consistently to different sub-task expressions, the corresponding task schema parameters are explicitly co-designated. For example, the target image direction for the observation task is the same for the slewing task turn_to and the actual image taking.

The abstract activation task task0 can further be decomposed into two variants, one dealing with another instrument

having to be turned off first, the other for situations in which the satellite has energy already available.

The implementation of the calibration process `auto_calibrate` is either atomic in the context of other observation tasks or it has to perform a preparatory slew into the calibration direction first.

The main combinatorial problem in this domain boils down to the question of how to establish the `pointing` state features. Along the decomposition hierarchy, any observation is self-contained such that the turning operation after a calibration step is properly instantiated and eventually will be fully causally supported from within the surrounding network actions. If a plan, however, contains multiple observation operations that are only developed to an intermediate level, there is typically some confusion about causal support with respect to the orientation of the satellite. Let us therefore briefly investigate the issues of implementing an observation in the presence of other observations.

A problem in the *Satellite* domain is typically given by a number of abstract observation tasks. The domain model offers four implementation variants, the applicability of which depends on the observation contexts:

1. First, the instrument is activated, then the satellite turns to the direction of scientific interest and finally takes the image. This is the base case for isolated (sub-) problems as explained above.
2. The instrument might be properly calibrated from previous observations. In this case, it suffices to slew the satellite and take the image.
3. Problems with many jobs may take advantage of decompositions that provide an activation-imaging skeleton for which the slews can be filled in later by task insertion modifications. This method therefore provides the causal information that connects activation and usage of the sensor. Please note that this requires planners that perform task insertion as well.[1]
4. If the configuration supports task insertion (like in the previous case) or if we have to deal with exceptional situations in which more than one image is required of a phenomenon, the fourth variant solely consists of a direct translation into taking the image.

Concluding Remarks

Satellite induces refinement spaces that contain many isomorphic plans, constructed around exactly one complex task: observation. The different methods do thereby not provide *alternative* ways of performing that task but rather define the configuration or situational environment of the observation process: with or without calibration, with or without preparation slews, and the like.

But the *Satellite* domain has several nice properties due to which it qualifies as an interesting demonstration and benchmark domain for hierarchical planning systems.

The main advantage is the intuitive simplicity of the application domain and the underlying principles. This also holds for future model extensions like incorporating temporal information, addressing energy consumption, etc. Any modification can easily be explained and its effects on the solution generation process investigated. It is also a relatively simple task to algorithmically generate problems, to validate solutions, and to judge problem complexity as well as solution quality. In contrast to other simple benchmark scenarios, planning for satellite observations exhibits satisfying variety and extension options.

Another observation on *Satellite* benchmark problems is that this domain allows to control problem complexity, for example in order to determine the scaling behaviour of a strategy, in several dimensions.

For example, the mildly sophisticated scaling by cloning observation jobs and scientific equipment, leads to an increasing number of self-similar sub-problems. Although this may be the intended scientific focus, it has to be taken into account that this kind of complexity may not favour a generally well performing search strategy. It makes solving ten times more observations "more difficult" in a similar way than stacking ten times more blocks did for non-hierarchical planning at the time.

In contrast, we can define benchmark problems with an increasing number of observation jobs that require an increasing number of modes and instruments on a constant number of satellites. This induces an increasing number of interacting sub-goals and causal interferences. As a consequence, solution density in the search space will decline.

A last aspect is the amount of overlapping target requirements, respectively instrument capabilities: if a set of observations can be performed by single platforms sequentially as well as by multiple platforms in parallel, optimality of the solutions becomes more and more an issue.

[1] Like PDDL, HDDL requires types to be disjunct and therefore no calibration target can, at the same time, be an object of scientific interest, which in turn allows no solutions for this method.

References

Biundo, S., and Schattenberg, B. 2001. From abstract crisis to concrete relief – a preliminary report on combining state abstraction and HTN planning. In *ECP 2001*, 157–168. AAAI Press.

Pragst, L.; Richter, F.; Bercher, P.; Schattenberg, B.; and Biundo, S. 2014. Introducing hierarchy to non-hierarchical planning models - a case study for behavioral adversary models. In *PuK 2014*.

Sacerdoti, E. D. 1974. Planning in a hierarchy of abstraction spaces. *Artificial Intelligence* 5(2):115–135.

Schattenberg, B.; Bidot, J.; and Biundo, S. 2007. On the construction and evaluation of flexible plan-refinement strategies. In *KI 2007*, 367–381. Springer.

Schattenberg, B.; Weigl, A.; and Biundo, S. 2005. Hybrid planning using flexible strategies. In *KI 2005*, 249–263. Springer.

Schattenberg, B. 2009. *Hybrid Planning & Scheduling*. Ph.D. Dissertation, Ulm University, Germany.

Smith, D. E.; Frank, J.; and Jónsson, A. K. 2000. Bridging the gap between planning and scheduling. *The Knowledge Engineering Review* 15(1):47–83.