The Hierarchical Woodworking Domain

Bernd Schattenberg[1,*] and **Pascal Bercher**[2,*]
[1] 3B intelligent solutions, Germany, schattenberg@3b-intelligent-solutions.com
[2] The Australian National University, Australia, pascal.bercher@anu.edu.au
[*] The domain was created while still being at the Institute of Artificial Intelligence of Ulm University, Germany

Abstract

The *Woodworking* 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 *Woodworking* domain models workflows in a workshop setting. Wooden boards are cut into parts of required sizes, which are planed, smoothened, and finally painted in specified colours and qualities. The various spray and varnish paints thereby require different preparation treatments of the respective wooden surface. Combinations of these process steps into proper workflows are provided by the decomposition methods.

The main causal interactions on a task level occur when some of the heavier workshop tools abrade the surface of wooden items, thereby undoing previous treatment steps. Other minor planning-sub-problems emerge when some of the machinery involved only allows for processing one item at a time. In its current version, this merely imposes limitations on possible plan linearisations but may become subject to plan optimization for resource-aware planners.

The *Woodworking* domain has been introduced as a benchmark to the planning community in 2008 for IPC 6. We have developed a hierarchical version of it in order to analyse planning strategy designs for hybrid planning systems using landmarks (Elkawkagy et al. 2012; Bercher, Keen, and Biundo 2014) and this domain model has finally been translated into the current, purely hierarchical version.

This short description focuses on the design decisions that led to the hybrid planning domain model for the formal framework introduced by Biundo and Schattenberg (2001) and Schattenberg (2009), that means, on the specifics of adding hierarchical features to a non-hierarchical domain model (cf. the work by Pragst et al. (2014)).

Mechanics of the Model

The type hierarchy of *Woodworking* establishes three main categories of objects: the wooden targets of creative handicraft (`woodobj`), workshop machines (`machine`), and a general object type. Wooden objects can be either boards or parts with the latter being obtained from the former by cutting them out. The type `object` is the most general type and an entry point for declaring constants representing specific wood materials, colours, and the like.

Regarding the state-variant features, most of the predicates describe the processing states of the processed wooden objects. This includes the following:

- $\texttt{unused}_{\texttt{part}}$
- $\texttt{boardsize}_{\texttt{board,aboardsize}}$
- $\texttt{treatment}_{\texttt{part,treatmentstatus}}$
- $\texttt{surface\_condition}_{\texttt{woodobj,surface}}$
- $\texttt{colour}_{\texttt{part,acolour}}$
- $\texttt{wood}_{\texttt{woodobj,awood}}$
- $\texttt{available}_{\texttt{woodobj}}$

State features built from `unused` and `available` serve as semaphore, respectively book-keeping implementations for the pre-defined pool of part and wooden object constants. E.g., once a wooden object is used in a process step as a `part`, i.e., if a process step objective is assigned the respective constant, that very `part` constant is taken from the pool of available constants by negating its `unused` property. As a side effect of this technique, the `wood` property has to be passed on from the raw board (for which it's technically unchangeable) to its processed part artefact.

While the rest of the above state features is straight forward with more intuitive semantics, the following state-invariant relations require some examination:

- $\texttt{machine\_present}_{\texttt{machine}}$
- $\texttt{has\_colour}_{\texttt{machine,acolour}}$
- $\texttt{goalsize}_{\texttt{part,apartsize}}$
- $\texttt{boardsize\_successor}_{\texttt{aboardsize,aboardsize}}$
- $\texttt{grind\_treatment\_change}_{\texttt{treatmentstatus,treatmentstatus}}$
- $\texttt{is\_smooth}_{\texttt{surface}}$
- $\texttt{contains\_part}_{\texttt{board,part}}$

`Machine_present` denotes the availability of mobile workshop machines and corresponds to availability of wooden objects – of course, machines are not used up in the process. Similarly, `colour` is used to describe painted parts, while `has_colour` represents the colour a respective workshop machine has at its disposal.

Sizes and treatment states are modeled by explicitly stating the available constants in a problem description. On these values, a simple symbolical computation is axiomatised implicitly in the task specifications: Parts, the process target objects, can have a `goalsize` of small, medium or

large. On the other hand, boards, i.e. the process source materials, are described via the state-variant `boardsize`, which is supposed to have at least three discrete values. The board size symbols are arranged according to the facts over the `boardsize_successor` predicate. The rationale is now a very high-level abstraction of a wood-cutting process, in which small sized parts reduce the board size by one `boardsize_successor`, medium sized ones by two, and large sized ones by three.

Please note that the current version of the hierarchical `Woodworking` domain, as does its non-hierarchical origin, does not yet support a re-use of the remaining boards after a part has been cut from them. Furthermore, the intention behind `contains_part` was unfortunately not documented, as it had not been used in any domain element.

The remaining physics of surface treatment by grinding (levels of removing varnish from wooden surfaces) is represented by the `grind_treatment_change` in a similar way like what we have shown for board sizes.

Regarding action specifications, the following operation for varnishing a part by means of immersion can serve as a common example for the domain:

```
(:action do_immersion_varnish
  :parameters (?p - part
               ?m - immersion_varnisher
               ?c - acolour
               ?s - surface)
  :precondition
   (and (available ?p)
        (has_colour ?m ?c)
        (surface_condition ?p ?s)
        (is_smooth ?s)
        (treatment ?p untreated))
  :effect
   (and (not (treatment ?p untreated))
        (treatment ?p varnished)
        (colour ?p ?c)))
        (not (colour ?p natural))
```

Similar actions specifications are used for other means of applying colour to the wood object, basically depending on surface condition and treatment status. Grinding and planing are implemented analogously, emphasising the change of surface condition, colour stripping, and the like.

When analysing the original IPC benchmark problems for this domain, it becomes apparent that the *intended procedure* for *processing* wooden parts follow a general pattern: cutting and sawing a board in order to obtain a suitably sized part, grinding or planing that part to achieve the desired surface condition, and finally applying a specific paint to realize a specific colour and treatment. The hierarchical *Woodworking* domain captures this pattern of sub-processes by defining corresponding abstract tasks like `cut_and_saw`, `grindNplane` and the like. The resulting decomposition hierarchy is relatively flat with one major intermediate level of abstract tasks that allow for alternative decompositions into the process options as described above.

We defined the complex task schemata in the fashion of ABSTRIPS operator reductions (Sacerdoti 1974). That means, we do not employ state abstraction axioms as described by Biundo and Schattenberg (2001) but simply generalise the preconditions and effects of the primitive implementations. The most abstract task, the processing objective, is thus defined as follows:

```
(:task process
  :parameters (?p - part ?c - acolour
               ?oldS - surface
               ?newS - surface)
  :effect (and (colour ?p ?c)))
```

On this level of abstraction, processing consists of colouring a wooden part with all causal interactions delegated to the expansion methods. The methods themselves implement the different process variants by combining related tasks into modular subroutines. Please note that the decomposition hierarchy does not impose semantic restrictions on the solution space.

Properties of the Model

The domain is partially ordered and acyclic. It contains six abstract tasks, 13 primitive tasks, and 14 methods, where each task has between two and four methods. The IPC set contains 30 problem instances of various degrees of hardness. The first eleven instances were modeled by hand by the authors and are relatively easy with maximal shortest solution lengths of 15 steps. The remaining problem instances were created by a random generator, written by Gregor Behnke (based on an existing one for the original domain). The hardest instance has a shortest solution with 178 steps. Using the grounder by Behnke et al. (2020), we can report that the number of ground primitive and abstract tasks as well as decomposition methods ranges from a few dozen for the lower first (smaller) problem instances until 87.680 primitive tasks, 120.819 abstract tasks, and 592.235 decomposition methods for the largest problem instance.

References

Behnke, G.; Höller, D.; Schmid, A.; Bercher, P.; and Biundo, S. 2020. On succinct groundings of HTN planning problems. In *Proceedings of AAAI 2020*, 9775–9784. AAAI.

Bercher, P.; Keen, S.; and Biundo, S. 2014. Hybrid planning heuristics based on task decomposition graphs. In *Proceedings of SoCS 2014*, 35–43. AAAI.

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

Elkawkagy, M.; Bercher, P.; Schattenberg, B.; and Biundo, S. 2012. Improving hierarchical planning performance by the use of landmarks. In *Proceedings of AAAI 2012*, 1763–1769. 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 *Proceedings of PuK 2014*.

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

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