The Barman-HTN Domain for IPC 2020
Max Waters, Lin Padgham, Sebastian Sardina
RMIT University, Melbourne, Australia
max.waters@rmit.edu.au, sebastian.sardina@rmit.edu.au, lin.padgham@rmit.edu.au

Abstract

The Barman-HTN domain is an HTN decomposition of the IPC Barman domain. It extends Barman with a task network that guides cocktail creation while retaining the primitive operators that made the original so challenging for the delete-relaxation heuristic.

The Barman-HTN domain is an HTN extension of the well-known IPC Barman domain (López, Celorrio, and Olaya 2015), in which cocktails must be prepared out of various ingredients. Solving a Barman instance requires careful management of limited resources. For example, shot glasses and shakers must be clean and empty before use, but filling one with an ingredient deletes both of these conditions, meaning that it must be emptied and cleaned before re-use. This makes Barman particularly challenging for planners that use the delete-relaxation heuristic: as action preconditions are frequently deleted and can only be restored by executing further actions, delete-relaxation tends to produce overly optimistic estimates.

This property of the domain operators, combined with recent interest in the delete-relaxation heuristic in HTN planners (Höller, Bercher, and Behnke 2020) and the fact that the cocktail construction task can be naturally decomposed into subtasks, suggest that Barman is a suitable basis for an HTN benchmark domain.

Barman-HTN extends Barman with an HDDL (Höller et al. 2020) task network that guides the pouring and mixing of ingredients and also provides careful resource management. For example, the method MakeCocktail (Figure 1) decomposes the task of mixing and shaking a cocktail. The first subtask, AchieveCleanShaker, ensures that the shaker is clean and empty, and the two instances of AchieveContainsShakerIngredient ensure that it contains the required ingredients. The next two steps, AchieveHandEmpty and AchieveHandOnTable, ensure that one hand is empty and the other is holding the shaker. These subtasks satisfy the preconditions of the final step, the action shake, which results in the shaker containing the cocktail.

Resource management is handled by tasks and methods that bring about a required condition from any given state. For example, the task AchieveHandOnTable(?h, ?c) produces the condition holding(?h, ?c), and is decomposed by two methods. If hand ?h is already holding container ?c, then the empty method AchieveHandOnTableNull is applicable. Otherwise, PickUp (Figure 1) decomposes the task into AchieveHandOnTable(?h) and AchieveOnTable(?c), that satisfy the preconditions of the primitive action grasp(?h, ?c) by ensuring that ?h is empty and ?c can be picked up, respectively. The task network contains other such decompositions for resource management tasks such as cleaning glasses and shakers, and emptying hands.

References

Figure 1: The MakeCocktail and Pickup methods.