AI 101
Planks Domain - Report

Hariel Giacomuzzi
School of Informatics of Pontifícia Universidade Católica do Rio Grande do Sul
6681 Av. Ipiranga
Porto Alegre, Rio Grande do Sul

Abstract

In this report We'll quickly explain the planks domain, as well as the concept behind our implementation of the domain description in PDDL.

The planks domain consists of a series of little islands in a archipelago, each island can only retain one agent. In order to a agent reach other islands it must use a plank to create a bridge and then cross to the other island, since the islands are very small the destination island must be empty for the agent to cross. It's not allowed to an agent remain in a plank, otherwise the plank will sink. It's also true that a agent can only handle one plank at time, and he must be able to catch up the plank in order to use it.

As we'll see there's a little variation of the problem domain that's called *Heavy Planks* in this variation of the problem the planks are (as the name suggests) heavier and to handle them it's needed two agents working in parallel to move tha plank around. In this case there must have parallel islands for the agents to walk by since the islands only carry one agent. In the following sections we'll describe how the author implemented this situations in PDDL.

Starting simple

To demonstrate the formalization of the simple domain we'll use the image of the first problem and the structures used.

Figura 1: Representation of first problem

As we can see in the left side of Fig.1 is the initial state of the problem and at the right side the desired state or final state. So first we need a declaration of some aspects of the world, and as the figure shows, we need an island, a plank an actor or agent as predicates. And the actions needed are only the move action.

The predicates

The predicates used here was the *Island* wich may inform to us that the object is or not a island, the *Plank* wich is similar and so says if an object is or not a plank, the *Agent* that tell's us if the object is a agent or not. As you may be wondering this is not the enought for formalizing the envioriment and so we used some "utilities"predicates, are them *at*, *empty* and *adjacent*. This last ones tells us where some object is on relation to other objects, if some object (in this case the island) are empty and if some two objects are adjacent to each other, respectively.

The actions

In this point theo nly action needed is the move action, wich is fomalized as follows.The parameters are the agent, the island, the destination and the plank. The precondicions are that the agent is at the island given by the *at* predicate, the island and the destination are adjacents as informed by *adjacent*, the plank is *at* the island and *at* the destination and the destination is *empty*.

The resolution

Using the planner given by the Professor Dr. Felipe Meneguzzi the plan found was to do one action of *move* the running time was of 220ms and the total memory used was 123MB.

Adding some complexity

Now we have seen the most simple sample let's make things a little more interesant.

Figura 2: Representation of first problem

In this sample we will need some other features in our formalization. In the predicates we add the *carry* and the *canCarry* predicates wich means that the actor is carrying a plank and if the actor is not carrying a plank respectively. The actions now are added by the *pick* action wich receives the the agent the plank and the islandhas as preconditions

the *canCarry* of the agent as well as the agent and the plank have the *at* of the island.The results of this action is that the agent has now the *carry* and the ***not*** *canCarry* and all the islands has the ***not*** *at* plank. The other action needed is the *lay* action, wich has as parameters the agent, plank, island and destination. The precondicions are the agent *carry* the plank, the agent is *at* the island and the destination is *adjacent* of the island. The results of this action is the plank *at* the island and *at* the destination, the agent has now the *canCarry* and the ***not*** *carry*

Results of second problem

The actions of the plan are, *move*, *pick*, *lay*, *move*. The running time was of 275ms and the total amount of memory used was 123MB.

The Heavy Planks Domain

As we told in the begining of this paper we have a little variation of the problem called the *Heavy Planks* this domain has exactly the same rules of the first domain with the exception of the plank need two agents for moving them arround. Basicaly the only change in the model of the problem was in the actions, now we need to assure that two agents has the *canCarry* at the same time and both agents are in a island *adjacent* to the same plank. Once that condicions are OK, we can pick the plank, we also need to assure the move and the lay action to do the same since one agent cannot move without laying the plank and for laying the plank both agents need to do this (we can let one agent lay the plank without "asking"for the other agent but this will not be nice with his friends so in order to every agent be nice and friendly to each other we do not let this happen :)).

Results of the Heavy Planks

Yeah we don't have a image for represnting this :(. The problem consists of four islands organized in pairs with a heavy plank in the middle, one agent in each side of the plank and the objective is to move the plank to the island at the side of each agent. The resulting action of this was, *heavyPick*, *heavyMove* and *heavyLay* in this order. The Execution time was of 854ms and the total mount of memory used was 155.5MB.

Lessons learned

As we can see throu out this quick papper formalizing a envioriment is not that hard, but it need some time and attention to be payed since many times the very "simple"aspects of the envioriment are in genral the most hard to design because as ew humans has a large "database"of known situations we assume that as simple and many times it need a little attention because some situations has some details that makes all the difference in the time of running a planner and ask for a solution. The problem shown here is not so complex and we spent a few hours modeling it. But we also notice that once the base of the problem has been modeled the remaining features or improovements are kind of easy to implement.