CS5446 AI Planning and Decision Making

Semester 1, AY2022-23
Assignment 1

Agrawal, Shubhankar
A0248330L

Sagar, Sanchit
A0232478Y

September 1, 2022

1 Introduction

1.1 Homework Problem 1: The Blocks World Reloaded

a Expressing blocks-world as a planning problem in PDDL

Assumptions

- The table always has clear space, i.e. Clear(Table) is always true.
- The movable predicate defines if an object can be moved, the blocks A and B are always movable hence we always have Movable(A) and Movable(B).

Initial State:

On(B, Table) $\wedge On(A, B) \wedge Block(A) \wedge Block(B) \wedge Clear(Table) \wedge Movable(A) \wedge Movable(B)$

Goal State:

On(A, Table) $\wedge On(B, A)$

Action Schema:

In order to account for making a block $\neg$ Clear when an object is placed on top of it, we create two move actions: Move and MoveToTable. The Move action is to move a block to another block, whereas the MoveToTable action is to move a block to the table. This factors in the assumption that on any block, there is space for only one block.

Action(Move(x, from, to)),
PRECOND: On(x, from) $\wedge\neg\ On(x, to) \wedge Clear(to) \wedge Clear(x) \wedge Movable(x) \wedge Block(to)$
$EFFECT : On(x, to) \wedge \neg\ On(x, from) \wedge Clear(from) \wedge \neg\ Clear(to)$

Action(MoveToTable(x, from)),
PRECOND: On(x, from) $\wedge\neg\ On(x, Table) \wedge Clear(x) \wedge Movable(x)$
$EFFECT : On(x, Table) \wedge \neg\ On(x, from) \wedge Clear(from)$

b Search Tree Forward Search

The figure 1 is attached at the end of the document

c Search Tree Backward Search

The figure 2 is attached at the end of the document

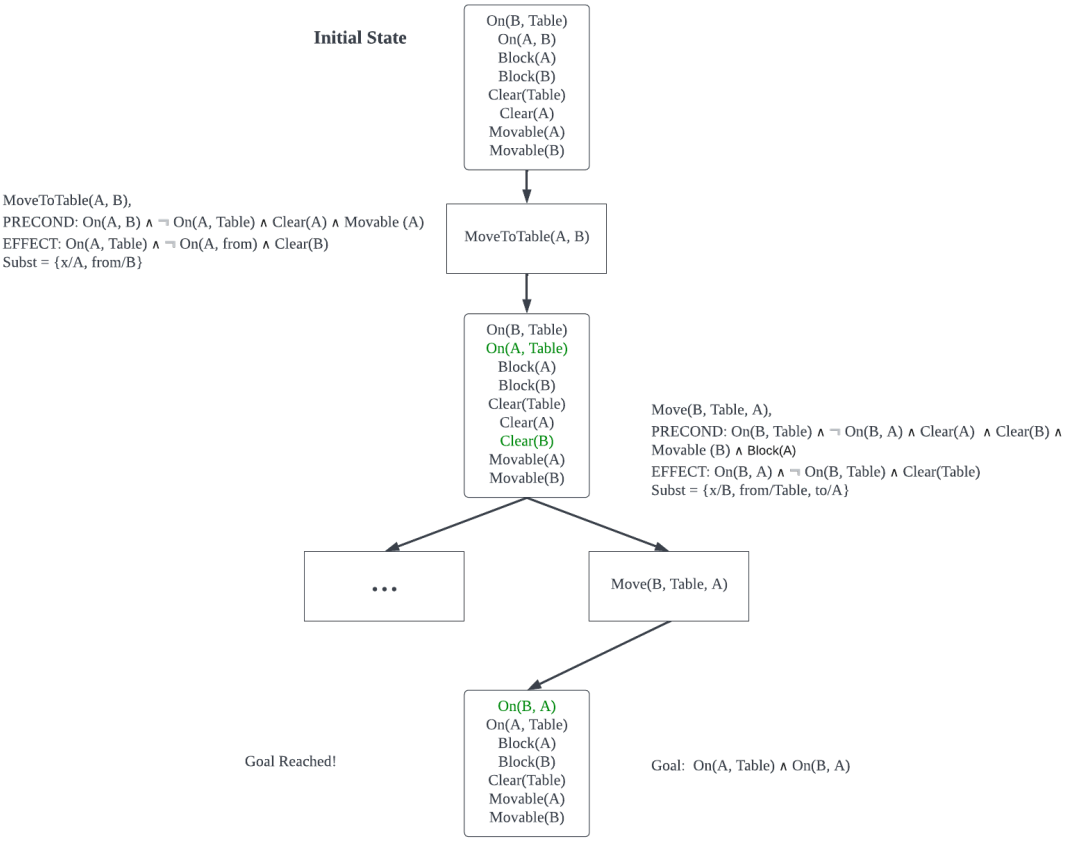

Figure 1: Forward Search Plan

d Valid Plan for Problem

MoveToTable(A, B)
Move(B, Table, A)

1.2 Homework Problem 2: Programming as Planning

Assumptions

- We define 6 predicates, namely At, IsMemory, HasValue,Variable, Value and Null. IsMemory defines the physical space occupied in the memory by a variable, Value defines a value, HasValue defines the value held by a memory location, Variable defines the name/reference to the memory location, Null defines whether a value is null or not and finally At defines that a variable is at a particular memory location.
- The variables in the assignment problem are always declared.
- The variable being assigned to another variable is always initialised i.e. there will not be a situation wherein an uninitialized variable is assigned to another variable. Eg. If a = b, then we assume that b always has an initial value i.e. it is initialized.
- When a new variable is created it is always assigned the null value. In the case of the assignment a = b, the variable being assigned to (i.e. a), the value at a's memory can be either null or defined.

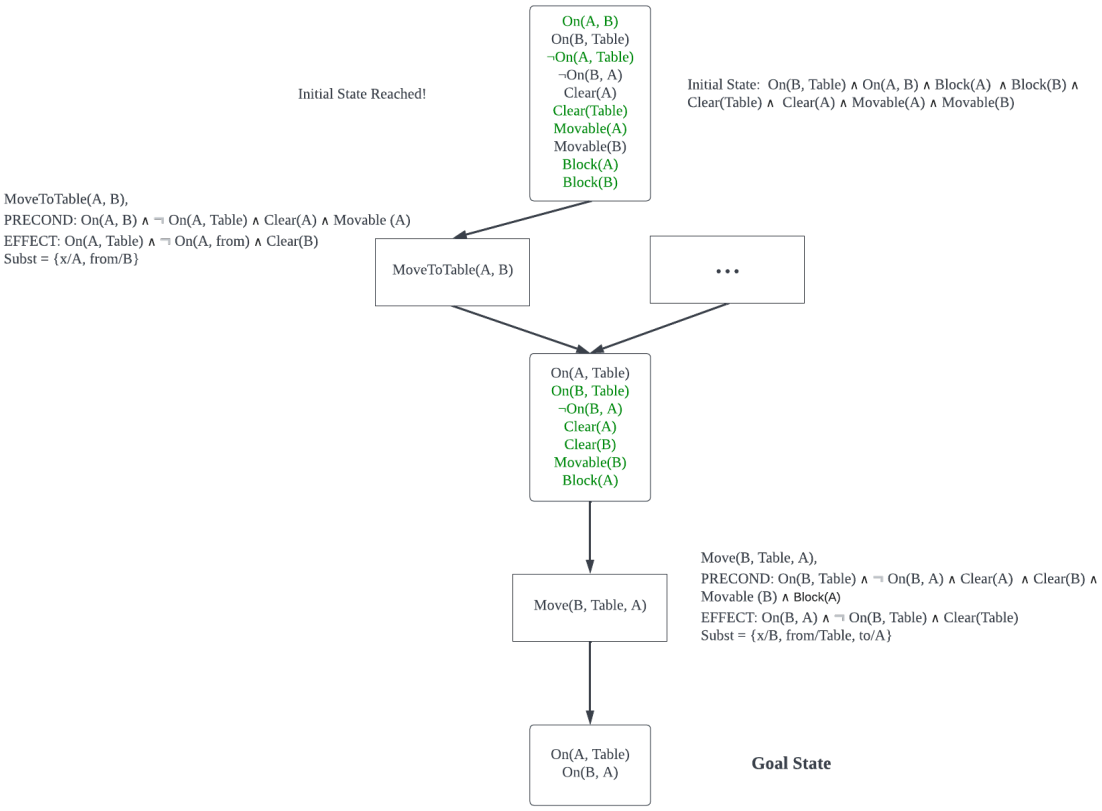

Figure 2: Backward Search Plan

a Action Schema for Assignment Operator

Action(Assign(a,b,m1,m2,v1,v2)),
PRECOND: Variable(a) $\wedge Variable(b) \wedge At(a, m1) \wedge At(b, m2) \wedge IsMemory(m1) \wedge IsMemory(m2) \wedge HasValue(m1, v1) \wedge HasValue(m2, v2) \wedge (Value(v1) \vee Null(v1)) \wedge Value(v2)$
$EFFECT : HasValue(m1, v2) \wedge \neg\ HasValue(m1, v1)$

b Plan using object creation for exchanging values

We add define another action schema for object creation following which we show how it can be used along with the assignment action to exchange the values of two variables.

Action(CreateObject(a, m1, v1)),
PRECOND: $\neg\ Variable(a) \wedge Memory(m1) \wedge (Value(v1) \vee Null(v1))$
$EFFECT : Variable(a) \wedge At(a, m1) \wedge HasValue(m1, v1)$

Using these two actions of Assign and CreateObject, we create a plan to exchange the value of two variables, to perform the following operation using a temporary variable.

```
temp = a
a = b
b = temp
```

We assume a and b have been declared and defined with a value. Variable a is at location m1 with value v1

and variable b is at location m2 with value v2

Action Plan

CreateObject(temp, m3, null)
Assign(temp, a, m3, m1, null, v1)
Assign(a, b, m1, m2, v1, v2)
Assign(b, temp, m2, m3, v2, v1)