AI Exp: 10

IMPLEMENTATION OF BLOCKS WORLD PROBLEM

Submitted By

Name: VINOTH S
Reg No: RA1911030010103
Date: 05-04-2022
GitHub Link: https://github.com/vk1308

AIM:

TO STUDY THE IMPLEMENTATION OF BLOCKS WORLD PROBLEM

CODE (Python)

```
class PREDICATE:
    def __str__(self):
        pass
    def __repr__(self):
        pass
    def __eq__(self, other) :
        pass
    def __hash__(self):
        pass
    def get_action(self, world_state):
        pass


class Operation: def
    ______str__(self):
        pass
    def __repr__(self):
```

```
__pass
_def___eq___(self, other) :
__pass
_def precondition(self):
__pass
_def delete(self):
__pass
_def add(self):
__pass


class ON(PREDICATE):
_def___init___(self, X, Y):
__self.X = X
__self.Y = Y
_def___str___(self):
__return "ON({X},{Y})".format(X=self.X,Y=self.Y)
  def___repr___(self):
__return self._str_() def_eq
  _____(self, other) :
__return self.____dict___== other.____dict___and self.____class___== other.____class___
  def___hash___(self):
___return hash(str(self))
_def get_action(self, world_state):
__return StackOp(self.X,self.Y)
```

```
class ONTABLE(PREDICATE):
    def __init__(self, X):
        self.X = X
    def __str__(self):
        return "ONTABLE({X})".format(X=self.X)
    def __repr__(self):
        return self._str_() def eq
    _____(self, other) :
        return self.____dict___ == other.____dict___and self.____class___ == other.____class___
    def ___hash___(self):
        ___return hash(str(self))
    def get_action(self, world_state):
        return PutdownOp(self.X)


class CLEAR(PREDICATE):
    def __init__(self, X):
        self.X = X
    def __str__(self):
        return "CLEAR({X})".format(X=self.X)
        self.X = X
    def __repr__(self):
        return self._str_() def eq
    _____(self, other) :
```

```
__return self.____dict___== other.___dict___and self.___class___== other.___class___
 def___hash___(self):
__return hash(str(self))
_def get_action(self, world_state):
__for predicate in world_state:
___if isinstance(predicate,ON) and predicate.Y==self.X:
____return UnstackOp(predicate.X, predicate.Y)
__return None


class HOLDING(PREDICATE):
_def___init___(self, X):
__self.X = X
_def___str___(self):
__return "HOLDING({X})".format(X=self.X)
 def___repr___(self):
__return self._str_() def_eq
 ______(self, other) :
__return self.____dict___== other.___dict___and self.___class___== other.___class___
 def___hash___(self):
__return hash(str(self))
_def get_action(self, world_state):
__X = self.X
__if ONTABLE(X) in world_state:
___return PickupOp(X)
```

```
__else:
___for predicate in world_state:
____if isinstance(predicate,ON) and predicate.X==X:
_____return UnstackOp(X,predicate.Y)


class ARMEMPTY(PREDICATE):
_def___init___(self):
__pass
_def___str___(self):
__return "ARMEMPTY"
 def___repr___(self):
__return self._str_() def eq
 _____(self, other) :
__return self.____dict___== other.___dict___and self.___class___== other.___class___
 def___hash___(self):
__return hash(str(self))
_def get_action(self, world_state=[]):
__for predicate in world_state:
___if isinstance(predicate,HOLDING):
____return PutdownOp(predicate.X)
__return None


class StackOp(Operation):def
 _____init___(self, X, Y):
```

```
__self.X = X
__self.Y = Y
_def___str___(self):
__return "STACK({X},{Y})".format(X=self.X,Y=self.Y)
  def___repr___(self):
__return self._str_() def_eq
  ______(self, other) :
__return self.____dict___== other.____dict___and self.____class___== other.____class___
  def precondition(self):
__return [ CLEAR(self.Y) , HOLDING(self.X) ]
  def delete(self):
__return [ CLEAR(self.Y) , HOLDING(self.X) ]
  def add(self):
__return [ ARMEMPTY() , ON(self.X,self.Y) ]


class UnstackOp(Operation):
_def___init___(self, X, Y):
__self.X = X
__self.Y = Y
_def___str___(self):
__return "UNSTACK({X},{Y})".format(X=self.X,Y=self.Y)
  def___repr___(self):
__return self._str_() def_eq
  ______(self, other) :
```

```
__return self.____dict___== other.___dict___and self.___class___== other.___class___
  def precondition(self):
__return [ ARMEMPTY() , ON(self.X,self.Y) , CLEAR(self.X) ]def
  delete(self):
__return [ ARMEMPTY() , ON(self.X,self.Y) ]def
  add(self):
__return [ CLEAR(self.Y) , HOLDING(self.X) ]


class PickupOp(Operation):
_def___init__(self, X):
__self.X = X
_def___str__(self):
__return "PICKUP({X})".format(X=self.X)
  def___repr__(self):
__return self._str_() def_eq
  _____(self, other) :
__return self.____dict___== other.___dict___and self.___class___== other.___class___
  def precondition(self):
__return [ CLEAR(self.X) , ONTABLE(self.X) , ARMEMPTY() ]def
  delete(self):
__return [ ARMEMPTY() , ONTABLE(self.X) ]
  def add(self):
__return [ HOLDING(self.X) ]
```

```
class PutdownOp(Operation):
 def___init___(self, X):
  self.X = X
 def___str___(self):
  return "PUTDOWN({X})".format(X=self.X)def
  ______repr___(self):
  return self._str_() def eq
  ______(self, other) :
  return self.____dict___== other.___dict___and self.___class___== other.___class___
  def precondition(self):
  return [ HOLDING(self.X) ]
  def delete(self):
  return [ HOLDING(self.X) ]
  def add(self):
  return [ ARMEMPTY() , ONTABLE(self.X) ]


def isPredicate(obj):
 predicates = [ON, ONTABLE, CLEAR, HOLDING, ARMEMPTY]
 for predicate in predicates:
  if isinstance(obj,predicate):
   return True
  return False
```

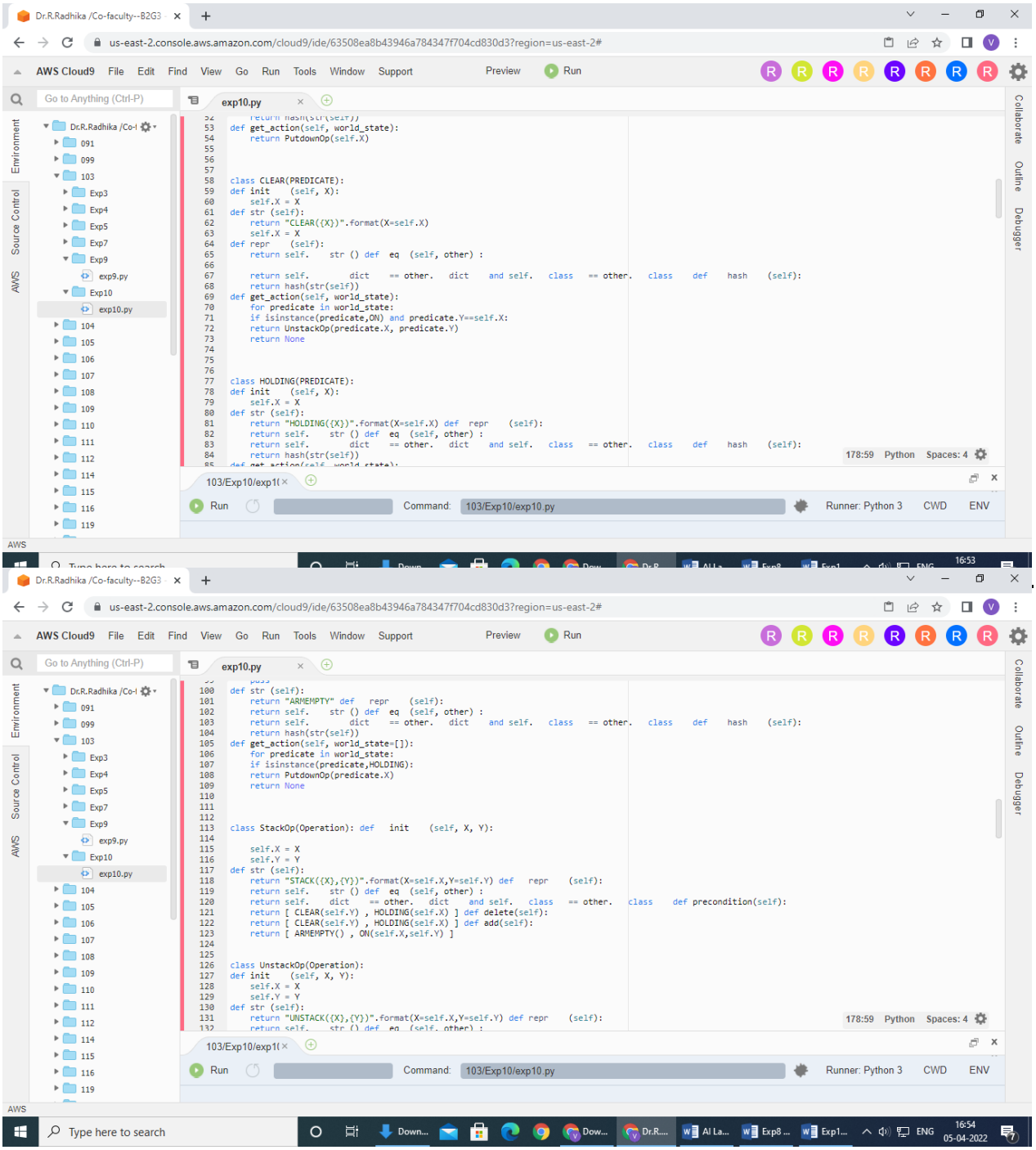
```
def isOperation(obj):
_operations = [StackOp, UnstackOp, PickupOp, PutdownOp]for
  operation in operations:
_
__if isinstance(obj,operation):
___return True
  return False
_


def arm_status(world_state):for
  predicate in world_state:
_
__if isinstance(predicate, HOLDING):
___return predicate
  return ARMEMPTY()
_



class GoalStackPlanner:


_def___init___(self, initial_state, goal_state):
__self.initial_state = initial_state
__self.goal_state = goal_state


_def get_steps(self):
__steps = []
__stack = []
```

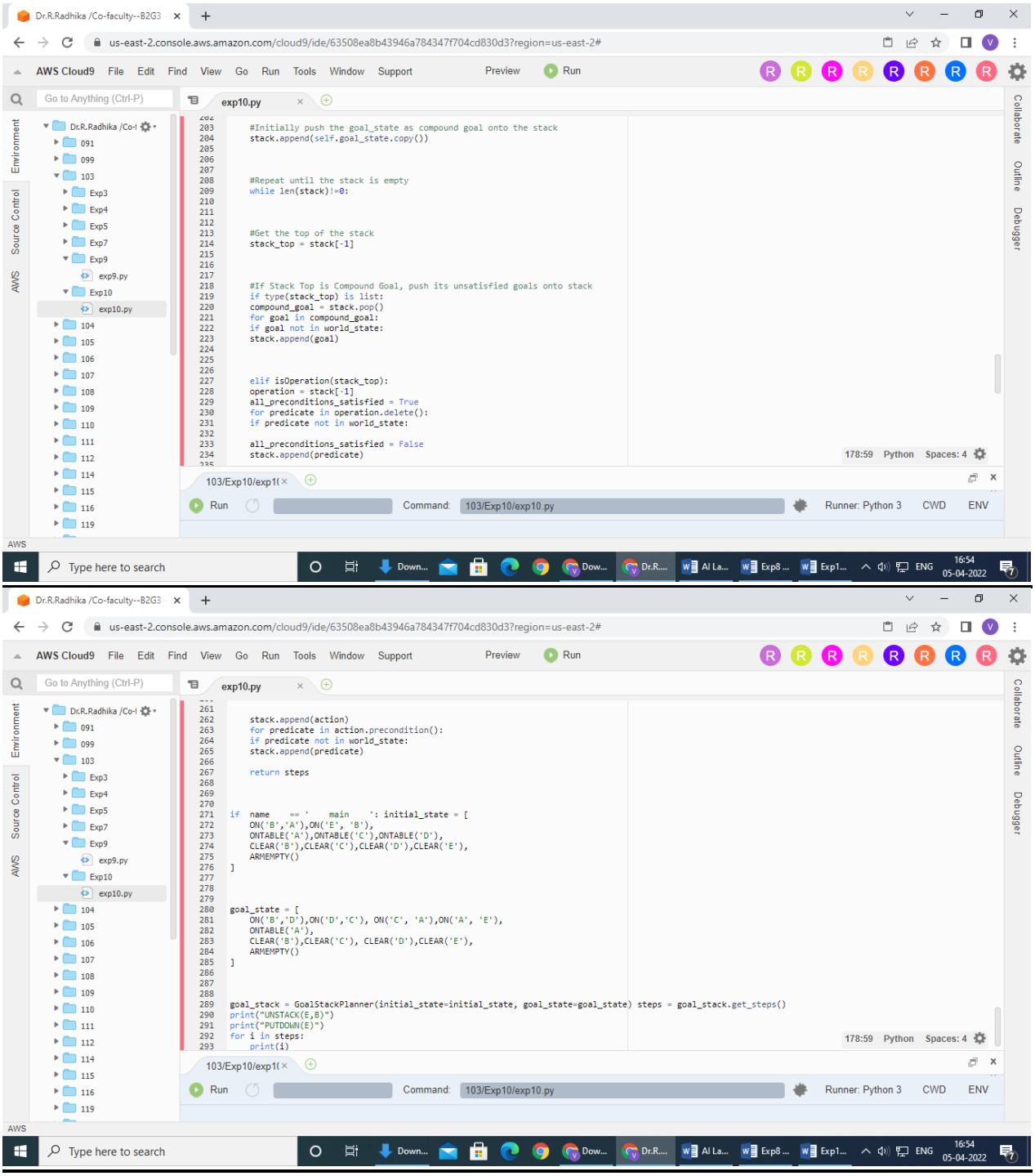
```
__#World State/Knowledge Base
__world_state = self.initial_state.copy()

__#Initially push the goal_state as compound goal onto the stack
__stack.append(self.goal_state.copy())

__#Repeat until the stack is empty
__while len(stack)!=0:

___#Get the top of the stack
___stack_top = stack[-1]

___#If Stack Top is Compound Goal, push its unsatisfied goals onto stack
___if type(stack_top) is list:
____compound_goal = stack.pop()
____for goal in compound_goal:
_____if goal not in world_state:
______stack.append(goal)

___elif isOperation(stack_top):
____operation = stack[-1]
____all_preconditions_satisfied = True
____for predicate in operation.delete():
_____if predicate not in world_state:
```

```
        all_preconditions_satisfied = False
        stack.append(predicate)

    if all_preconditions_satisfied:

        stack.pop()
        steps.append(operation)

        for predicate in operation.delete():
            world_state.remove(predicate)
        for predicate in operation.add():
            world_state.append(predicate)

    elif stack_top in world_state:
        stack.pop()
    else:
        unsatisfied_goal = stack.pop()
        action = unsatisfied_goal.get_action(world_state)

        stack.append(action)
        for predicate in action.precondition():
            if predicate not in world_state:
                stack.append(predicate)
```

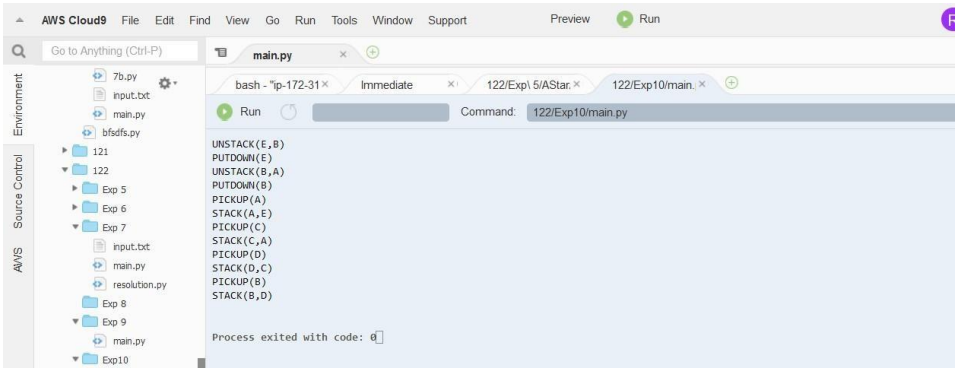
```
__return steps


if___name____== '___main___':
  initial_state = [
__ON('B','A'),ON('E', 'B'),
__ONTABLE('A'),ONTABLE('C'),ONTABLE('D'),
__CLEAR('B'),CLEAR('C'),CLEAR('D'),CLEAR('E'),
__ARMEMPTY()
_]


_goal_state = [
__ON('B','D'),ON('D','C'), ON('C', 'A'),ON('A', 'E'),
__ONTABLE('A'),
__CLEAR('B'),CLEAR('C'), CLEAR('D'),CLEAR('E'),
__ARMEMPTY()
_]


_goal_stack = GoalStackPlanner(initial_state=initial_state, goal_state=goal_state)steps =
  goal_stack.get_steps()
_print("UNSTACK(E,B)")
_print("PUTDOWN(E)")
_for i in steps:
__print(i)
```

IMPLEMENTATION:


Dr.R.Radhika /Co-faculty--B2G3
us-east-2.console.aws.amazon.com/cloud9/ide/63508ea8b43946a784347f704cd830d3?region=us-east-2#
AWS Cloud9 File Edit Find View Go Run Tools Window Support Preview Run
Go to Anything (Ctrl-P)
exp10.py
Environment
Source Control
AWS
Collaborate
Outline
Debugger
Dr.R.Radhika /Co-I
091
099
103
Exp3
Exp4
Exp5
Exp7
Exp9
exp9.py
Exp10
exp10.py
104
105
106
107
108
109
110
111
112
114
115
116
119
def get_action(self, world_state):
return PutdownOp(self.X)
class CLEAR(PREDICATE):
def init (self, X):
self.X = X
def str (self):
return "CLEAR({X})".format(X=self.X)
self.X = X
def repr (self):
return self. str () def eq (self, other) :
return self. dict == other. dict and self. class == other. class def hash (self):
return hash(str(self))
def get_action(self, world_state):
for predicate in world_state:
if isinstance(predicate,ON) and predicate.Y==self.X:
return UnstackOp(predicate.X, predicate.Y)
return None
class HOLDING(PREDICATE):
def init (self, X):
self.X = X
def str (self):
return "HOLDING({X})".format(X=self.X) def repr (self):
return self. str () def eq (self, other) :
return self. dict == other. dict and self. class == other. class def hash (self):
return hash(str(self))
178:59 Python Spaces: 4
103/Exp10/exp1(
Run
Command: 103/Exp10/exp10.py
Runner: Python 3 CWD ENV
AWS
Type here to search
16:53
Dr.R.Radhika /Co-faculty--B2G3
us-east-2.console.aws.amazon.com/cloud9/ide/63508ea8b43946a784347f704cd830d3?region=us-east-2#
AWS Cloud9 File Edit Find View Go Run Tools Window Support Preview Run
Go to Anything (Ctrl-P)
exp10.py
def str (self):
return "ARMEMPTY" def repr (self):
return self. str () def eq (self, other) :
return self. dict == other. dict and self. class == other. class def hash (self):
return hash(str(self))
def get_action(self, world_state=[]):
for predicate in world_state:
if isinstance(predicate,HOLDING):
return PutdownOp(predicate.X)
return None
class StackOp(Operation): def init (self, X, Y):
self.X = X
self.Y = Y
def str (self):
return "STACK({X},{Y})".format(X=self.X,Y=self.Y) def repr (self):
return self. str () def eq (self, other) :
return self. dict == other. dict and self. class == other. class def precondition(self):
return [CLEAR(self.Y) , HOLDING(self.X)] def delete(self):
return [CLEAR(self.Y) , HOLDING(self.X)] def add(self):
return [ARMEMPTY() , ON(self.X,self.Y)]
class UnstackOp(Operation):
def init (self, X, Y):
self.X = X
self.Y = Y
def str (self):
return "UNSTACK({X},{Y})".format(X=self.X,Y=self.Y) def repr (self):
178:59 Python Spaces: 4
103/Exp10/exp1(
Run
Command: 103/Exp10/exp10.py
Runner: Python 3 CWD ENV
AWS
Type here to search
16:54
05-04-2022


Dr.R.Radhika /Co-faculty--B2G3
us-east-2.console.aws.amazon.com/cloud9/ide/63508ea8b43946a784347f704cd830d3?region=us-east-2#
AWS Cloud9 File Edit Find View Go Run Tools Window Support Preview Run
Go to Anything (Ctrl-P)
exp10.py
Environment
Source Control
AWS
Collaborate
Outline
Debugger
Dr.R.Radhika /Co-l
091
099
103
Exp3
Exp4
Exp5
Exp7
Exp9
exp9.py
Exp10
exp10.py
104
105
106
107
108
109
110
111
112
114
115
116
119
#Initially push the goal_state as compound goal onto the stack
stack.append(self.goal_state.copy())
#Repeat until the stack is empty
while len(stack)!=0:
#Get the top of the stack
stack_top = stack[-1]
#If Stack Top is Compound Goal, push its unsatisfied goals onto stack
if type(stack_top) is list:
compound_goal = stack.pop()
for goal in compound_goal:
if goal not in world_state:
stack.append(goal)
elif isOperation(stack_top):
operation = stack[-1]
all_preconditions_satisfied = True
for predicate in operation.delete():
if predicate not in world_state:
all_preconditions_satisfied = False
stack.append(predicate)
178:59 Python Spaces: 4
103/Exp10/exp1(
Run
Command: 103/Exp10/exp10.py
Runner: Python 3 CWD ENV
AWS
Type here to search
ENG 16:54 05-04-2022
Dr.R.Radhika /Co-faculty--B2G3
us-east-2.console.aws.amazon.com/cloud9/ide/63508ea8b43946a784347f704cd830d3?region=us-east-2#
AWS Cloud9 File Edit Find View Go Run Tools Window Support Preview Run
Go to Anything (Ctrl-P)
exp10.py
stack.append(action)
for predicate in action.precondition():
if predicate not in world_state:
stack.append(predicate)
return steps
if __name__ == '__main__': initial_state = [
ON('B','A'),ON('E', 'B'),
ONTABLE('A'),ONTABLE('C'),ONTABLE('D'),
CLEAR('B'),CLEAR('C'),CLEAR('D'),CLEAR('E'),
ARMEMPTY()
]
goal_state = [
ON('B','D'),ON('D','C'), ON('C', 'A'),ON('A', 'E'),
ONTABLE('A'),
CLEAR('B'),CLEAR('C'), CLEAR('D'),CLEAR('E'),
ARMEMPTY()
]
goal_stack = GoalStackPlanner(initial_state=initial_state, goal_state=goal_state) steps = goal_stack.get_steps()
print("UNSTACK(E,B)")
print("PUTDOWN(E)")
for i in steps:
print(i)
178:59 Python Spaces: 4
103/Exp10/exp1(
Run
Command: 103/Exp10/exp10.py
Runner: Python 3 CWD ENV
AWS
Type here to search
ENG 16:54 05-04-2022

OUTPUT:



RESULT:

Therefore, the implementation of block world problem has been completedsuccessfully