Calvin Chen

Module 7 Self Check

Block Worlds Continuation

10.

|  |  |  |  |
| --- | --- | --- | --- |
| State | Plan | Stack | Actions |
| ontable(A) | Unstack(C, A) | Goal([on(A, C), on(C, B) | Single unsatisfied goal, so need to find operator. Let us set operator as Stack(A, C). Preconditions holding(A) and clear(C) are pushed to stack. State is unchanged because no operator used. |
| ontable(B) | Stack(C, B) | Goal(on(A, C)) |
| on(C, B) |  |  |
| clear(C) |  |  |
| clear(A) |  |  |
| handempty |  |  |

11.

|  |  |  |  |
| --- | --- | --- | --- |
| State | Plan | Stack | Actions |
| ontable(A) | Unstack(C, A) | Goal([on(A, C), on(C, B) | Goal(Clear(C)) is met, so we can remove it from the State |
| ontable(B) | Stack(C, B) | Op(Stack(A, C)) |
| on(C, B) |  | Goal(Holding(A)) |
| clear(C) |  | Goal(Clear(C)) |
| clear(A) |  |  |
| handempty |  |  |

12

|  |  |  |  |
| --- | --- | --- | --- |
| State | Plan | Stack | Actions |
| ontable(A) | Unstack(C, A) | Goal([on(A, C), on(C, B) | Goal(Holding(A)) is unsatisfied. Let us set operator as Pickup(A). Preconditions are ontable(A), handempty, and Clear(A) |
| ontable(B) | Stack(C, B) | Op(Stack(A, C)) |
| on(C, B) |  | Goal(Holding(A)) |
| clear(A) |  |  |
| handempty |  |  |
|  |  |  |

13, 14, 15

|  |  |  |  |
| --- | --- | --- | --- |
| State | Plan | Stack | Actions |
| ontable(A) | Unstack(C, A) | Goal([on(A, C), on(C, B) | The next three goals, ontable(A), Clear(A), and handempty are all met, so we can pop from stack |
| ontable(B) | Stack(C, B) | Op(Stack(A, C)) |
| on(C, B) |  | Op(Pickup(A)) |
| clear(A) |  | Goal(ontable(A)) |
| handempty |  | Goal(Clear(A)) |
|  |  | Goal(handempty) |

16

|  |  |  |  |
| --- | --- | --- | --- |
| State | Plan | Stack | Actions |
| ontable(B) | Unstack(C, A) | Goal([on(A, C), on(C, B) | All goal states have been met for Op(Pickup(A)), so we can carry it out and apply post-conditions to the state list. Post-condition is holding(A), on(A, C), !ontable(A), !clear(A), and !handempty. Update the stack |
| on(C, B) | Stack(C, B) | Op(Stack(A, C)) |
| ontable(A) |  | Op(Pickup(A)) |
| clear(A) |  |  |
| handempty |  |  |

17

|  |  |  |  |
| --- | --- | --- | --- |
| State | Plan | Stack | Actions |
| ontable(B) | Unstack(C, A) | Goal([on(A, C), on(C, B) | The next operator, Op(Stack(A, C)), has had all of its preconditions fulfilled. We can pop it and update the States with its postconditons. Postconditions are:  on(A, C), clear(A), handempty, !holding(A), !clear(c) |
| on(C, B) | Stack(C, B) | Op(Stack(A, C)) |
| holding(A) | Pickup(A) |  |
| on(A, C) |  |  |

18.

|  |  |  |  |
| --- | --- | --- | --- |
| State | Plan | Stack | Actions |
| ontable(B) | Unstack(C, A) | Goal([on(A, C), on(C, B) | Goal state has been achieved in the State list. |
| on(C, B) | Stack(C, B) |  |
| clear(A) | Pickup(A) |  |
| on(A, C) | Stack(A, C) |  |
| handempty |  |  |

2. Forward Planner

A forward planner uses a DFS-like approach to reach the end or goal state. Successor functions try to unify available actions with the current state to expand the frontier.

Given the State, Goal, and Stack displayed above:

|  |  |  |
| --- | --- | --- |
| State | Plan | Stack |
| ontable(A) | Unstack(C, A) | Goal([on(A, C), on(C, B) |
| ontable(B) | Stack(C, B) | Goal(on(A, C)) |
| on(C, B) |  |  |
| clear(C) |  |  |
| clear(A) |  |  |
| handempty |  |  |

We first try to carry out the action Stack(C, B). However, this is not possible because the preconditions have not been met. Going through the list of actions, we find that Pickup(A) is possible, and append that successor onto the frontier, completing a step.

|  |  |  |
| --- | --- | --- |
| State | Plan | Stack |
| holding(A) | Unstack(C, A) | Goal([on(A, C), on(C, B) |
| ontable(B) | Stack(C, B) |  |
| on(C, B) | Pickup(A) |  |
| clear(C) |  |  |
|  |  |  |
|  |  |  |