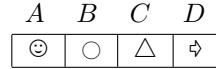


**Part 1 - Artificial Intelligence**  
*(Time to complete the test: 2:30 hours)*

Consider the following scenario. An agent ( $\odot$ ) moves in a map with four cells,  $A$ ,  $B$ ,  $C$ ,  $D$ . Initially, cell  $A$  contains the agent and cells  $B$  and  $C$  contain, respectively, a circle ( $\circlearrowleft$ ) and a triangle ( $\triangle$ ). Cell  $D$  is the exit ( $\diamond$ ). The initial situation is depicted below:



Assume the environment is modelled as follows.

*Non-Fluents:*

- $Item(x)$ , denoting that  $x$  is an item.
- $Right(x_1, x_2)$ , denoting that cell  $x_2$  is next to the right of cell  $x_1$ ;
- $Exit(x)$ , denoting that  $x$  is an exit.

*Fluents:*

- $AgentAt(x)$  denoting that the agent is at cell  $x$ .
- $AgentHas(x)$  denoting that the agent has item  $x$  with itself (the agent can carry many items);
- $ItemAt(x, y)$  denoting that item  $x$  is at cell  $y$ .

*Actions:*

- $move(x)$ , which allows the agent to move to cell  $x$ . The action can be done if cell  $x$  is next to the right of the cell where the agent currently is. The effect is that the agent is in cell  $x$  (and no longer in the cell it moved from) and has with itself all the items present in  $x$  (in addition to those it already had). Further, the collected items are no longer in the cell.
- $drop()$ , which allows the agent to drop in current location all the items it has with itself. The action can always be done. The effect is that all the items the agent had with itself will now be in the current location.
- $exit()$ , which allows the agent to exit the map. The action can be done if the agent is currently in an exit cell and has all the items with itself. The effect is that the agent is no longer in the cell it was before and keeps all items it had with itself.

*Initial situation:*

Cell  $B$  is next to the right of  $A$ ,  $C$  is next to the right of  $B$ , and  $D$  is next to the right of  $C$ . Cell  $D$  is the exit cell. The agent is at cell  $A$ . There are two items: *circle* and *triangle*. Item *circle* is in cell  $B$  and item *triangle* is in cell  $C$ .

**Exercise 1.** First, formalize the above scenario as a Basic Action Theory in Reiter's Situation Calculus. Then, given the sequence of actions  $\varrho = move(B); move(C); move(D)$ , check by regression whether:

1.  $\varrho$  is executable in  $S_0$ ;
2.  $\varrho$  results in a situation where cell  $C$  contains some item;
3.  $\varrho$  results in a situation where the agent has all the items with itself.

**Exercise 2.** Consider the following goal:  $ItemAt(circle, D) \wedge ItemAt(triangle, D)$ . First formalize the above scenario as a PDDL domain file and a PDDL problem file. Then:

1. Draw the corresponding transition system;
2. Solve planning for achieving the above goal by using forward depth-first search (uninformed), reporting the steps of the forward search computation, and return the resulting plan.

### Exercise 3.

- Formalize the following statements in FOL (by introducing suitable predicates, functions and constants):
  - $(\phi_1)$  Every sphere is a solid.
  - $(\phi_2)$  Every cube is a solid.
  - $(\phi_3)$  Every solid is a geometric shape.
  - $(\phi_4)$  Every square is a geometric shape.
- Considering the KB  $\mathcal{K} = \{\phi_1, \dots, \phi_4\}$  defined above, use the tableaux method to check whether it is possible that a square is also a sphere (you must define a suitable FOL formula to express this). If so, show a model  $I$  of  $\mathcal{K}$  where this holds.

### Solution

*Precondition Axioms:*

- $Poss(move(x), s) \equiv \exists y. (AgentAt(y, s) \wedge Right(y, x))$
- $Poss(drop(), s) \equiv true$
- $Poss(exit(), s) \equiv \exists x. (AgentAt(x, s) \wedge Exit(x)) \wedge \forall y. (Item(y) \supset AgentHas(y, s))$

*Successor State Axioms:*

- Start with *Effect Axioms*:

- $a = move(x) \supset (AgentAt(x, do(a, s)) \wedge \forall y. (AgentAt(y, s) \supset \neg AgentAt(y, do(a, s))) \wedge \forall z. (ItemAt(z, x, s) \supset (AgentHas(z, do(a, s)) \wedge \neg ItemAt(z, x, do(a, s)))))$
- $a = drop() \supset ((\forall x. (AgentHas(x, s) \supset \neg agentHas(x, do(a, s)))) \wedge (\forall x \forall y. (AgentHas(x, s) \wedge AgentAt(y, s)) \supset ItemAt(x, y, do(a, s))))$
- $a = exit() \supset \forall x. (AgentAt(x, s) \supset \neg AgentAt(x, do(a, s)))$

- Normalize:

- $a = move(x) \supset AgentAt(x, do(a, s))$
- $(\exists x. a = move(x) \wedge AgentAt(y, s)) \supset \neg AgentAt(y, do(a, s))$
- $(\exists x. a = move(x) \wedge ItemAt(z, x, s)) \supset AgentHas(z, do(a, s))$
- $(a = move(x) \wedge ItemAt(z, x, s)) \supset \neg ItemAt(z, x, do(a, s))$
- $(a = drop() \wedge AgentHas(x, s) \wedge AgentAt(y, s)) \supset ItemAt(x, y, do(a, s))$
- $(a = drop() \wedge AgentHas(x, s)) \supset \neg agentHas(x, do(a, s))$
- $(a = exit() \wedge AgentAt(x, s)) \supset \neg AgentAt(x, do(a, s))$

- Obtain Successor-State Axioms (SSAs) (by applying Explanation Closure):

- $AgentAt(x, do(a, s)) \equiv a = move(x) \vee (AgentAt(x, s) \wedge \neg((\exists y. a = move(y) \wedge AgentAt(x, s)) \vee (a = exit() \wedge AgentAt(x, s))))$ ,  
simplified as:  
 $AgentAt(x, do(a, s)) \equiv a = move(x) \vee (AgentAt(x, s) \wedge \neg((\exists y. a = move(y)) \vee a = exit()))$
- $AgentHas(x, do(a, s)) \equiv (\exists y. a = move(y) \wedge ItemAt(x, y, s)) \vee (AgentHas(x, s) \wedge \neg(a = drop() \wedge AgentHas(x, s)))$   
simplified as:  
 $AgentHas(x, do(a, s)) \equiv (\exists y. a = move(y) \wedge ItemAt(x, y, s)) \vee (AgentHas(x, s) \wedge a \neq drop())$
- $ItemAt(x, y, do(a, s)) \equiv (a = drop() \wedge AgentHas(x, s) \wedge AgentAt(y, s)) \vee (ItemAt(x, y, s) \wedge \neg(a = move(y) \wedge ItemAt(x, y, s))),$   
simplified as:  
 $ItemAt(x, y, do(a, s)) \equiv (a = drop() \wedge AgentHas(x, s) \wedge AgentAt(y, s)) \vee (ItemAt(x, y, s) \wedge a \neq move(y))$

*Initial Situation:*

$\mathcal{D}_0$  is the set including the following formulas:

# BASIC ACTION THEORY

## PRECONDITION ACTIONS

$\text{Poss}(\text{move}(x), s) \equiv \exists y. \text{AGENTAT}(y, s) \wedge \text{RIGHT}(y, x)$

$\text{Poss}(\text{DROP}(), s) \equiv \text{TRUE}$

$\text{Poss}(\text{EXIT}(), s) \equiv (\exists x. \text{AGENTAT}(x, s) \wedge \text{EXIT}(x)) \wedge (\forall y. \text{ITEMAT}(y, x) \Rightarrow \text{AGENTHAS}(y, s))$

## SUCCESSOR STATE AXIOMS

### EFFECT AXIOMS

$\alpha = \text{move}(x) \supset \text{AGENTAT}(x, \text{do}(\alpha, s))$   
 $\wedge \forall y. (\text{AGENTAT}(y, s) \wedge x \neq y \supset \neg \text{AGENTAT}(y, \text{do}(\alpha, s)))$   
 $\wedge \forall z. (\text{ITEMAT}(z, x, s) \supset \neg \text{ITEMAT}(z, x, \text{do}(\alpha, s)) \wedge \text{AGENTHAS}(z, \text{do}(\alpha, s)))$

$\alpha = \text{DROP}() \supset (\forall x. (\text{AGENTHAS}(x, s) \supset \neg \text{AGENTHAS}(x, \text{do}(\alpha, s)))) \wedge$   
 $(\forall x, y. (\text{AGENTHAS}(x, s) \wedge \text{AGENTAT}(y, s) \supset \text{ITEMAT}(x, y, \text{do}(\alpha, s))))$

$\alpha = \text{EXIT}() \supset \forall x. (\text{AGENTAT}(x, s) \supset \neg \text{AGENTAT}(x, \text{do}(\alpha, s)))$

### NORMALIZE

$\alpha = \text{move}(x) \supset \text{AGENTAT}(x, \text{do}(\alpha, s))$

$(\exists x. \alpha = \text{move}(x) \wedge \text{AGENTAT}(y, s) \wedge x \neq y \supset \neg \text{AGENTAT}(y, \text{do}(\alpha, s)))$

$(\exists x. \alpha = \text{move}(x) \wedge \text{ITEMAT}(z, x, s) \supset \neg \text{ITEMAT}(z, x, \text{do}(\alpha, s)))$

$(\alpha = \text{DROP}() \wedge \text{AGENTHAS}(x, s) \wedge \text{AGENTAT}(y, s) \supset \text{ITEMAT}(x, y, \text{do}(\alpha, s)))$

$(\alpha = \text{DROP}() \wedge \text{AGENTHAS}(x, s) \supset \neg \text{AGENTHAS}(x, \text{do}(\alpha, s)))$

$(\alpha = \text{EXIT}() \wedge \text{AGENTAT}(x, s) \supset \neg \text{AGENTAT}(x, \text{do}(\alpha, s)))$

### EXPLANATION CLOSURE

$\text{AGENTAT}(x, \text{do}(\alpha, s)) \equiv (\alpha = \text{move}(x)) \vee (\text{AGENTAT}(x, s) \wedge$   
 $\neg ((\exists x. \alpha = \text{move}(x) \wedge \text{AGENTAT}(x, s) \wedge x \neq y) \vee$   
 $(\alpha = \text{EXIT}() \wedge \text{AGENTAT}(x, s))))$



$\text{AGENTAT}(x, \text{do}(\alpha, s)) \equiv (\alpha = \text{move}(x)) \vee (\text{AGENTAT}(x, s) \wedge$   
 $\neg ((\exists x. \alpha = \text{move}(x) \wedge x \neq y) \vee (\alpha = \text{EXIT}())))$

$$\text{AGENT HAS}(x, \text{DO}(\alpha, s)) \equiv (\exists y. \alpha = \text{move}(y) \wedge \text{ITEMAT}(x, y, s)) \vee (\text{AGENT HAS}(x, s) \wedge \neg(\alpha = \text{DROP}() \wedge \text{AGENT HAS}(x, s)))$$



$$\text{AGENT HAS}(x, \text{DO}(\alpha, s)) \equiv (\exists y. \alpha = \text{move}(y) \wedge \text{ITEMAT}(x, y, s)) \vee (\text{AGENT HAS}(x, s) \wedge \neg \alpha = \text{DROP}())$$

$$\text{ITEMAT}(x, y, \text{DO}(\alpha, s)) \equiv (\alpha = \text{DROP}() \wedge \text{AGENT HAS}(x, s) \wedge \text{AGENTAT}(y, s)) \vee (\text{ITEMAT}(x, y, s) \wedge \neg(\alpha = \text{move}(y) \wedge \text{ITEMAT}(x, y, s)))$$



$$\text{ITEMAT}(x, y, \text{DO}(\alpha, s)) \equiv (\alpha = \text{DROP}() \wedge \text{AGENT HAS}(x, s) \wedge \text{AGENTAT}(y, s)) \vee (\text{ITEMAT}(x, y, s) \wedge \neg \alpha = \text{move}(y))$$

**Exercise 1.** First, formalize the above scenario as a Basic Action Theory in Reiter's Situation Calculus. Then, given the sequence of actions  $\varrho = \text{move}(B); \text{move}(C); \text{move}(D)$ , check by regression whether:

1.  $\varrho$  is executable in  $S_0$ ;
2.  $\varrho$  results in a situation where cell  $C$  contains some item;
3.  $\varrho$  results in a situation where the agent has all the items with itself.

## INITIAL SITUATIONS

$$\text{RIGHT}(x, y) \equiv (x = A \wedge y = B) \vee (x = B \wedge y = C) \vee (x = C \wedge y = D)$$

$$\text{EXIT}(x) \equiv x = D$$

$$\text{AGENTAT}(x, S_0) \equiv x = A$$

$$\text{ITEM}(x) \equiv (x = \text{CIRCLE}) \vee (x = \text{TRIANGLE})$$

$$\text{ITEMAT}(x, y, S_0) \equiv (x = \text{CIRCLE} \wedge y = B) \vee (x = \text{TRIANGLE} \wedge y = C)$$

## REGRESSION

$$S_1 = \text{DO}(\text{move}(B), S_0)$$

$$\varrho = \text{move}(B); \text{move}(C); \text{move}(D)$$

$$S_2 = \text{DO}(\text{move}(C), S_1)$$

$$S_3 = \text{DO}(\text{move}(D), S_2)$$

I)  $\varrho$  IS EXECUTABLE IN  $S_0$  IF:

$\text{move}(B)$  IS EXECUTABLE IN  $S_0$

$\text{move}(C)$  IS EXECUTABLE IN  $S_1$

$\text{move}(D)$  IS EXECUTABLE IN  $S_2$



a.  $D_0 \cup \text{Dunia} \models R[\text{Poss}(\text{move}(B), S_0)]$

b.  $D_0 \cup \text{Dunia} \models R[\text{Poss}(\text{move}(C), S_1)]$

c.  $D_0 \cup \text{Dunia} \models R[\text{Poss}(\text{move}(D), S_2)]$

LET'S REGRESS EACH FORMULA:

$$\text{a. } R[\text{Poss}(\text{move}(B), S_0)] = \exists y. R[\text{AGENTAT}(y, S_0) \wedge \text{RIGHT}(y, B)] = \\ = \exists y. R[\text{AGENTAT}(y, S_0)] \wedge \text{RIGHT}(y, B) = \exists y. \text{AGENTAT}(y, S_0) \wedge \text{RIGHT}(y, B)$$

TRUE FOR  $y=A \rightarrow \text{move}(B)$  IS EXECUTABLE IN  $S_0$  ✓

$$\text{b. } R[\text{Poss}(\text{move}(C), S_1)] = \exists y. R[\text{AGENTAT}(y, S_1) \wedge \text{RIGHT}(y, C)] = \\ = \exists y. R[\text{AGENTAT}(y, S_1)] \wedge \text{RIGHT}(y, C) = \\ = \exists y. R[\text{AGENTAT}(y, \text{DO}(\text{move}(B), S_0))] \wedge \text{RIGHT}(y, C) = \\ = \exists y. R[(\text{move}(B) = \text{move}(y)) \vee (\text{AGENTAT}(y, S_0) \wedge \\ \neg((\exists x. \text{move}(B) = \text{move}(x)) \vee (\text{move}(B) = \text{EXIT}())))] \wedge \text{RIGHT}(y, C) = \\ = \exists y. R[(\text{move}(B) = \text{move}(y)) \vee (\text{AGENTAT}(y, S_0) \wedge \neg(\text{TRUE} \vee \text{FALSE}))] \\ \wedge \text{RIGHT}(y, C) = \\ = \exists y. R[(\text{move}(B) = \text{move}(y))] \wedge \text{RIGHT}(y, C) = \\ = \exists y. \text{move}(B) = \text{move}(y) \wedge \text{RIGHT}(y, C)$$

TRUE FOR  $y=B \rightarrow \text{move}(C)$  IS EXECUTABLE IN  $S_1$  ✓

$$\text{c. } R[\text{Poss}(\text{move}(D), S_2)] = \exists y. R[\text{AGENTAT}(y, S_2) \wedge \text{RIGHT}(y, D)] = \\ = \exists y. R[\text{AGENTAT}(y, S_2)] \wedge \text{RIGHT}(y, D) = \\ = \exists y. R[\text{AGENTAT}(y, \text{DO}(\text{move}(C), S_1))] \wedge \text{RIGHT}(y, D) = \\ = \exists y. R[(\text{move}(C) = \text{move}(y)) \vee (\text{AGENTAT}(y, S_1) \wedge \\ \neg((\exists x. \text{move}(C) = \text{move}(x)) \vee (\text{move}(C) = \text{EXIT}())))] \wedge \text{RIGHT}(y, D) = \\ = \exists y. R[(\text{move}(C) = \text{move}(y)) \vee (\text{AGENTAT}(y, S_1) \wedge \text{FALSE})] \wedge \text{RIGHT}(y, D) = \\ = \exists y. R[\text{move}(C) = \text{move}(y)] \wedge \text{RIGHT}(y, D) = \\ = \exists y. \text{move}(C) = \text{move}(y) \wedge \text{RIGHT}(y, D)$$

TRUE FOR  $y=C \rightarrow \text{move}(D)$  IS EXECUTABLE IN  $S_2$  ✓

P IS EXECUTABLE IN  $S_0$  ✓

2) WE NEED TO CHECK IF:  $D_0 \cup D_{\text{unav}} \models R[\exists x. \text{ITEMAT}(x, c, S_3)]$

$$R[\exists x. \text{ITEMAT}(x, c, S_3)] = \exists x. R[\text{ITEMAT}(x, c, \text{DO}(\text{move}(D), S_2))] = \\ = \exists x. R[(\text{move}(D) = \text{DROP}() \wedge \text{AGENTHAS}(x, S_2) \wedge \text{AGENTAT}(c, S_2)) \vee \\ (\text{ITEMAT}(x, c, S_2) \wedge \text{move}(D) \neq \text{move}(c))] = \\ = \exists x. R[\text{ITEMAT}(x, c, S_2)] = \\ = \exists x. R[\text{ITEMAT}(x, c, \text{DO}(\text{move}(C), S_1))] = \\ = \exists x. R[(\text{move}(C) = \text{DROP}() \wedge \text{AGENTHAS}(x, S_1) \wedge \text{AGENTAT}(c, S_1)) \vee \\ (\text{ITEMAT}(x, c, S_1) \wedge \text{move}(C) \neq \text{move}(c))] = \\ = \exists x. R[\text{FALSE}]$$

FALSE → P DOESN'T RESULTS IN A SITUATION WHERE C CONTAINS SOME ITEM X

3) WE NEED TO CHECK IF:  $D_0 \cup D_{\text{unav}} \models R[\forall x. \text{ITEM}(x) \supset \text{AGENTHAS}(x, S_3)]$

$$R[\forall x. \text{ITEM}(x) \supset \text{AGENTHAS}(x, S_3)] = \forall x. R[\text{ITEM}(x)] \supset R[\text{AGENTHAS}(x, S_3)] =$$

$$= \forall x. \text{ITEM}(x) \supset R[\text{AGENTHAS}(x, S_3)] =$$

$$R[\text{AGENTHAS}(x, S_3)] = R[\text{AGENTHAS}(x, \text{DO}(\text{move}(D), S_1))] =$$

$$= R[(\exists y. \text{move}(D) = \text{move}(y) \wedge \text{ITEMAT}(x, y, S_2)) \vee (\text{AGENTHAS}(x, S_2) \wedge \text{move}(D) \neq \text{DROP}())] =$$

$$= \exists y. R[(\text{move}(D) = \text{move}(y) \wedge \text{ITEMAT}(x, y, S_2)) \vee R[\text{AGENTHAS}(x, S_2)]] =$$

$$= (\exists y. \text{move}(D) = \text{move}(y) \wedge R[\text{ITEMAT}(x, y, S_2)]) \vee R[\text{AGENTHAS}(x, S_2)] =$$

$$R[\text{ITEMAT}(x, y, S_2)] = R[\text{ITEMAT}(x, y, \text{DO}(\text{move}(c), S_1))] =$$

$$= R[(\text{move}(c) = \text{DROP}() \wedge \text{AGENTHAS}(x, S_1) \wedge \text{AGENTAT}(y, S_1)) \vee (\text{ITEMAT}(x, y, S_1) \wedge \text{move}(c) \neq \text{move}(y))] =$$

$$= R[\text{ITEMAT}(x, y, S_1)] \wedge \text{move}(c) \neq \text{move}(y) =$$

$$R[\text{ITEMAT}(x, y, S_1)] = R[\text{ITEMAT}(x, y, \text{DO}(\text{move}(B), S_0))] =$$

$$= R[(\text{move}(B) = \text{DROP}() \wedge \text{AGENTHAS}(x, S_0) \wedge \text{AGENTAT}(y, S_0)) \vee (\text{ITEMAT}(x, y, S_0) \wedge \text{move}(B) \neq \text{move}(y))] =$$

$$= R[\text{ITEMAT}(x, y, S_0) \wedge \text{move}(B) \neq \text{move}(y)] =$$

$$= \text{ITEMAT}(x, y, S_0) \wedge \text{move}(B) \neq \text{move}(y)$$

$$R[\text{ITEMAT}(x, y, S_0)] =$$

$$= \text{ITEMAT}(x, y, S_0) \wedge \text{move}(B) \neq \text{move}(y) \wedge \text{move}(c) \neq \text{move}(y)$$

$$R[\text{AGENTHAS}(x, S_2)] = R[\text{AGENTHAS}(x, \text{DO}(\text{move}(c), S_1))] =$$

$$= R[(\exists y. \text{move}(c) = \text{move}(y) \wedge \text{ITEMAT}(x, y, S_1)) \vee (\text{AGENTHAS}(x, S_1) \wedge \text{move}(c) \neq \text{DROP}())] =$$

$$= (\exists y. \text{move}(c) = \text{move}(y) \wedge R[\text{ITEMAT}(x, y, S_1)]) \vee R[\text{AGENTHAS}(x, S_1)] =$$

$$R[\text{AGENTHAS}(x, S_1)] = R[\text{AGENTHAS}(x, \text{DO}(\text{move}(B), S_0))] =$$

$$= R[(\exists y. \text{move}(B) = \text{move}(y) \wedge \text{ITEMAT}(x, y, S_0)) \vee (\text{AGENTHAS}(x, S_0) \wedge \text{move}(B) \neq \text{DROP}())] =$$

$$= (\exists y. \text{move}(B) = \text{move}(y) \wedge \text{ITEMAT}(x, y, S_0)) \vee \text{AGENTHAS}(x, S_0)$$

$$R[\text{AGENTHAS}(x, S_0)] = (\exists y. \text{move}(c) = \text{move}(y) \wedge \text{ITEMAT}(x, y, S_0) \wedge \text{move}(B) \neq \text{move}(y)) \vee (\exists y. \text{move}(B) = \text{move}(y) \wedge \text{ITEMAT}(x, y, S_0)) \vee \text{AGENTHAS}(x, S_0)$$

$$R[\text{AGENTHAS}(x, S_3)] = (\exists y. \text{move}(D) = \text{move}(y) \wedge \text{ITEMAT}(x, y, S_0) \wedge \text{move}(B) \neq \text{move}(y) \wedge \text{move}(c) \neq \text{move}(y)) \vee (\exists y. \text{move}(c) = \text{move}(y) \wedge \text{ITEMAT}(x, y, S_0) \wedge \text{move}(B) \neq \text{move}(y)) \vee (\exists y. \text{move}(B) = \text{move}(y) \wedge \text{ITEMAT}(x, y, S_0)) \vee \text{AGENTHAS}(x, S_0)$$

$$R[\forall x. \text{ITEM}(x) \supset \text{AGENTHAS}(x, S_3)] =$$

$$= \forall x. \text{ITEM}(x) \supset (\exists y. \text{move}(D) = \text{move}(y) \wedge \text{ITEMAT}(x, y, S_0) \wedge \text{move}(B) \neq \text{move}(y) \wedge \text{move}(c) \neq \text{move}(y)) \vee (\exists y. \text{move}(c) = \text{move}(y) \wedge \text{ITEMAT}(x, y, S_0) \wedge \text{move}(B) \neq \text{move}(y)) \vee (\exists y. \text{move}(B) = \text{move}(y) \wedge \text{ITEMAT}(x, y, S_0)) \vee \text{AGENTHAS}(x, S_0)$$

TRUE → P RESULTS IN A SITUATION WHERE THE  
AGENT HAS ALL ITEMS WITH ITSELF ✓

**Exercise 2.** Consider the following goal:  $\text{ItemAt}(\text{circle}, D) \wedge \text{ItemAt}(\text{triangle}, D)$ . First formalize the above scenario as a PDDL domain file and a PDDL problem file. Then:

1. Draw the corresponding transition system;
2. Solve planning for achieving the above goal by using forward depth-first search (uninformed), reporting the steps of the forward search computation, and return the resulting plan.

## PDDL

```

(: DEFINE ( DOMAIN  x - DOMAIN )
  (: REQUIREMENTS ·ADL)
  (: TYPES LOCATION ITEM)
  (: PREDICATES
    ( ITEM ?x - ITEM)
    (RIGHT ?x ?y - LOCATION)
    (EXIT ?x - LOCATION)
    (AGENTAT ?x - LOCATION)
    (AGENTHAS ?x - ITEM)
    (ITEMAT ?x - ITEM ?y - LOCATION)
  )
  (: ACTION MOVE
    : PARAMETERS ( ?x - LOCATION)
    : PRECONDITION (EXISTS ( ?y - LOCATION))(AND(AGENTAT ?y)(RIGHT ?y ?x)))
    : EFFECT ( AND
      (AGENTAT ?x)
      (FORALL (?y - LOCATION)
        (WHEN
          (AGENTAT ?y)
          (NOT(AGENTAT ?y))
        )
      )
      (FORALL (?z - ITEM)
        (WHEN
          (ITEMAT ?z ?x)
          (AND(NOT(ITEMAT ?z ?x))(AGENTHAS ?z))
        )
      )
    )
  );
  END OF MOVE
  (: ACTION DROP()
    : PARAMETERS ()
    : PRECONDITION ()
    : EFFECT (FORALL (?x - ITEM ?y - LOCATION)
      (WHEN
        (AND(AGENTHAS ?x)(AGENTAT ?y))
        (AND(NOT(AGENTHAS ?x))(ITEMAT ?x ?y))
      )
    )
  );
  END OF DROP
)

```

```

(: ACTION EXIT()
: PARAMETERS ()
: PRECONDITION (AND
    (EXISTS (?y - LOCATION))(AND (AGENTAT ?y)(EXIT ?y)))
    (FORALL (?x - ITEM)
        (WHEN
            (ITEM ?x)
            (AGENTHAS ?x)
        )
    )
)
· EFFECT (FORALL (?y - LOCATION)
    (WHEN
        (AGENTAT ?y)
        (NOT (AGENTAT ?y)))
    )
)
);
END OF EXIT
); END OF DEFINE

```

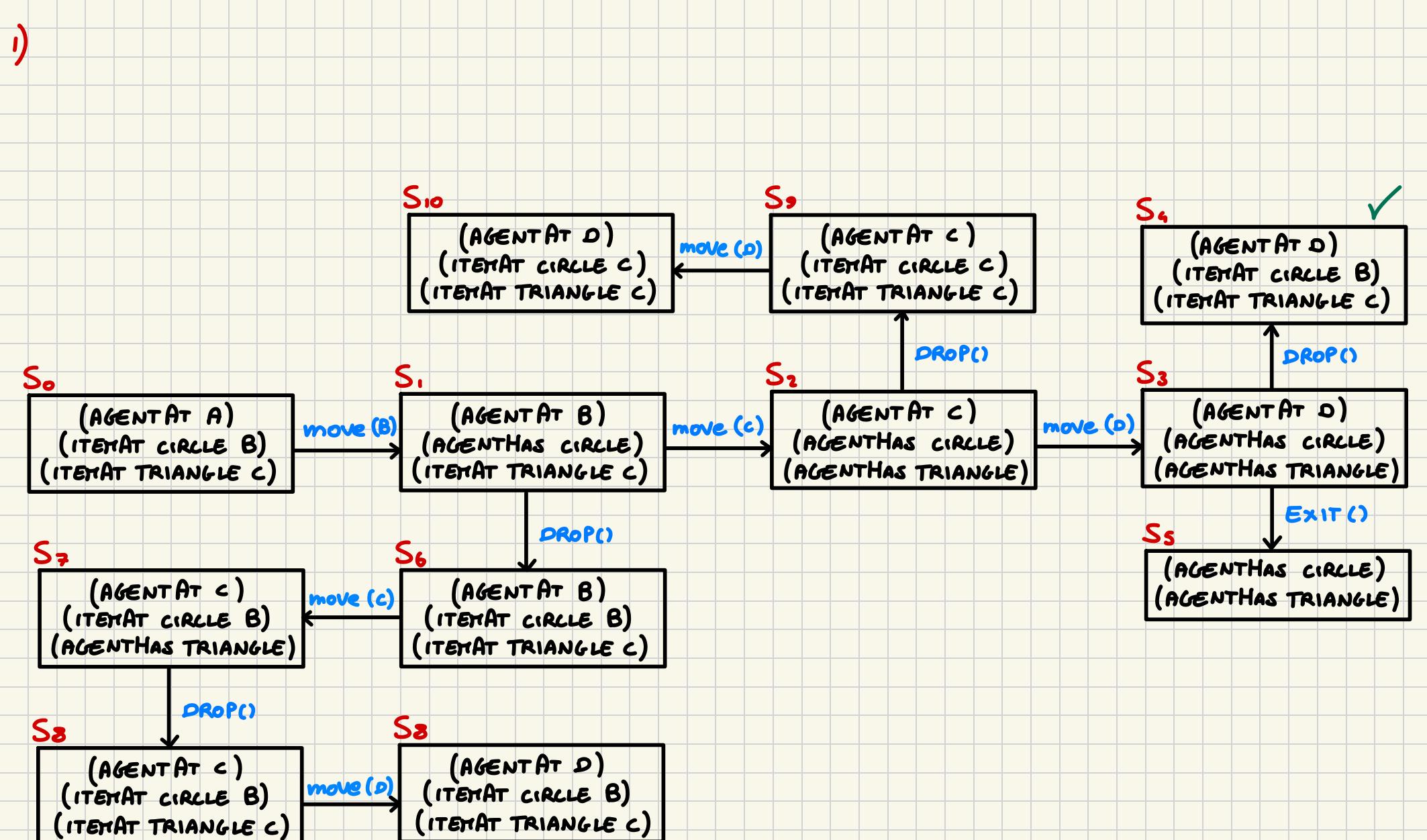
AFTER DEFINED THE DOMAIN FILE, WE DEFINE THE PROBLEM FILE.

goal:  $\text{ItemAt}(\text{circle}, D) \wedge \text{ItemAt}(\text{triangle}, D)$

```

(DEFINE (PROBLEM x.PROBLEM)(: DOMAIN x.DOMAIN)
    (: OBJECTS A B C D - LOCATION
        CIRCLE TRIANGLE - ITEM
    )
    (: INIT (RIGHT A B)(RIGHT B C)(RIGHT C D)(EXIT D)(AGENTAT A)
        (ITEMAT CIRCLE B)(ITEMAT TRIANGLE C)
    )
    (: GOAL (FORALL (?x - ITEM)
        (ITEMAT ?x D)
    )
)
);
END OF DEFINE

```



2)

goal:  $\text{ItemAt}(\text{circle}, D) \wedge \text{ItemAt}(\text{triangle}, D)$

0.  $\mathcal{T} = [(S_0, \text{EMPTY})]$

$m = \{S_0\}$

1.  $(\text{STATE}, \text{PLAN}) = \mathcal{T}. \text{POP}()$      $(S_0, \text{EMPTY})$

m. ADD ( $S_1$ )

$\mathcal{T}. \text{PUSH}(S_1, \text{move}(B))$

$\mathcal{T} = [(S_1, \text{move}(B))]$      $m = \{S_0, S_1\}$

2.  $(\text{STATE}, \text{PLAN}) = \mathcal{T}. \text{POP}()$      $(S_1, \text{move}(B))$

m. ADD ( $S_6$ )

$\mathcal{T}. \text{PUSH}(S_6, \text{move}(B) \text{ } \text{DROP}())$

m. ADD ( $S_2$ )

$\mathcal{T}. \text{PUSH}(S_2, \text{move}(B) \text{ } \text{move}(C))$

$\mathcal{T} = [(S_2, \text{move}(B) \text{ } \text{move}(C)), \dots]$      $m = \{S_0, S_1, S_2, S_6\}$

3.  $(\text{STATE}, \text{PLAN}) = \mathcal{T}. \text{POP}()$      $(S_2, \text{move}(B) \text{ } \text{move}(C))$

m. ADD ( $S_9$ )

$\mathcal{T}. \text{PUSH}(S_9, \text{move}(B) \text{ } \text{move}(C) \text{ } \text{DROP}())$

m. ADD ( $S_3$ )

$\mathcal{T}. \text{PUSH}(S_3, \text{move}(B) \text{ } \text{move}(C) \text{ } \text{move}(D))$

$\mathcal{T} = [(S_3, \text{move}(B) \text{ } \text{move}(C) \text{ } \text{move}(D)), \dots]$      $m = \{S_0, S_1, S_2, S_3, S_6, S_9\}$

4.  $(\text{STATE}, \text{PLAN}) = \mathcal{T}. \text{POP}()$      $(S_3, \text{move}(B) \text{ } \text{move}(C) \text{ } \text{move}(D))$

m. ADD ( $S_5$ )

$\mathcal{T}. \text{PUSH}(S_5, \text{move}(B) \text{ } \text{move}(C) \text{ } \text{move}(D) \text{ } \text{EXIT}())$

m. ADD ( $S_4$ )

$\mathcal{T}. \text{PUSH}(S_4, \text{move}(B) \text{ } \text{move}(C) \text{ } \text{move}(D) \text{ } \text{DROP}())$

$\mathcal{T} = [(S_4, \text{move}(B) \text{ } \text{move}(C) \text{ } \text{move}(D) \text{ } \text{DROP}()), \dots]$      $m = \{S_0, S_1, S_2, S_3, S_4, S_5, S_6, S_9\}$

5.  $(\text{STATE}, \text{PLAN}) = \mathcal{T}. \text{POP}()$      $(S_4, \text{move}(B) \text{ } \text{move}(C) \text{ } \text{move}(D) \text{ } \text{DROP}())$

PLAN: move(B) move(C) move(D) DROP()

### Exercise 3.

1. Formalize the following statements in FOL (by introducing suitable predicates, functions and constants):

- $(\phi_1)$  Every sphere is a solid.
- $(\phi_2)$  Every cube is a solid.
- $(\phi_3)$  Every solid is a geometric shape.
- $(\phi_4)$  Every square is a geometric shape.

2. Considering the KB  $\mathcal{K} = \{\phi_1, \dots, \phi_4\}$  defined above, use the tableaux method to check whether it is possible that a square is also a sphere (you must define a suitable FOL formula to express this). If so, show a model  $I$  of  $\mathcal{K}$  where this holds.

## 1) PREDICATES

**SPHERE (x)**: x IS A SPHERE

**CUBE (x)**: x IS A CUBE

**SOLID (x)**: x IS A SOLID

**SQUARE (x)**: x IS A SQUARE

**SHAPE (x)**: x IS A GEOMETRIC SHAPE

## FORMULAS

$$\phi_1 = \forall x. \text{SPHERE}(x) \supset \text{SOLID}(x)$$

$$\phi_2 = \forall x. \text{CUBE}(x) \supset \text{SOLID}(x)$$

$$\phi_3 = \forall x. \text{SOLID}(x) \supset \text{SHAPE}(x)$$

$$\phi_4 = \forall x. \text{SQUARE}(x) \supset \text{SHAPE}(x)$$

$$\phi_5 = \exists x. \text{SQUARE}(x) \wedge \text{SPHERE}(x)$$

2)

$\alpha$ rules	$\frac{\phi \wedge \psi}{\begin{array}{c} \phi \\ \psi \end{array}}$	$\frac{\neg(\phi \vee \psi)}{\begin{array}{c} \neg\phi \\ \neg\psi \end{array}}$	$\frac{\neg\neg\phi}{\begin{array}{c} \phi \\ \neg\psi \end{array}}$	$\frac{\neg(\phi \supset \psi)}{\begin{array}{c} \phi \\ \neg\psi \end{array}}$
$\beta$ rules	$\frac{\phi \vee \psi}{\begin{array}{c} \phi \\ \psi \end{array}}$	$\frac{\phi \supset \psi}{\begin{array}{c} \neg\phi \\ \psi \end{array}}$	$\frac{\neg(\phi \wedge \psi)}{\begin{array}{c} \neg\phi \\ \neg\psi \end{array}}$	$\frac{\phi \equiv \psi}{\begin{array}{c} \phi \\ \psi \end{array}}$

$\gamma$ rules	$\frac{\forall x.\phi(x)}{\phi(t)}$	$\frac{\neg\exists x.\phi(x)}{\neg\phi(t)}$
$\delta$ rules	$\frac{\neg\forall x.\phi(x)}{\neg\phi(c)}$	$\frac{\exists x.\phi(x)}{\phi(c)}$

$\forall x. SPHERE(x) \supset SOLID(x)$  1

$\forall x. CUBE(x) \supset SOLID(x)$  2

$\forall x. SOLID(x) \supset SHAPE(x)$  3

$\forall x. SQUARE(x) \supset SHAPE(x)$  4

$\exists x. SQUARE(x) \wedge SPHERE(x)$  5

| δ ON 5

SQUARE(a)  $\wedge$  SPHERE(a) 6

| α ON 6

SQUARE(a) 7

SPHERE(a) 8

| δ ON 1

¬SPHERE(a)  $\vee$  SOLID(a) 9

| β ON 9

¬SPHERE(a) 10

SOLID(a) 11

CLASH WITH δ

¬CUBE(a)  $\vee$  SOLID(a) 12

| β ON 12

¬CUBE(a) 13

SOLID(a) 14

| γ ON 3

¬SOLID(a)  $\vee$  SHAPE(a) 15

| γ ON 3

¬SOLID(a)  $\vee$  SHAPE(a) 15

| β ON 15

¬SOLID(a) 16

SHAPE(a) 17

CLASH WITH 11

¬SQUARE(a)  $\vee$  SHAPE(a) 18

| β ON 18

¬SQUARE(a)

SHAPE(a)

CLASH WITH 7

OPEN

### MODEL I OF K:

$$\Delta^I = \{a\}$$

$$SPHERE^I = \{a\}$$

$$CUBE^I = \{\}$$

$$SOLID^I = \{a\}$$

$$SHAPE^I = \{a\}$$

$$SQUARE^I = \{a\}$$



| β ON 15

¬SOLID(a) 16

SHAPE(a) 17

| γ ON 4

¬SQUARE(a)  $\vee$  SHAPE(a) 18

| β ON 18

SHAPE(a)

CLASH WITH 7

OPEN

- $Item(x) \equiv (x = circle) \vee (x = triangle)$
- $Right(x, y) \equiv (x = A \wedge y = B) \vee (x = B \wedge y = C) \vee (x = C \wedge y = D)$
- $Exit(x) \equiv x = D$
- $AgentAt(x, S_0) \equiv x = A$
- $ItemAt(x, y, S_0) \equiv (x = circle \wedge y = B) \vee (x = triangle \wedge y = C)$

*Regression:*

For  $\varrho = move(B); move(C); move(D)$ , let:  $S_1 = do(move(B), S_0); S_2 = do(move(C), S_1); S_3 = do(move(D), S_2)$ .

To check whether  $\varrho$  is executable in  $S_0$ , we need to check whether:  $move(B)$  is executable in  $S_0$ ;  $move(C)$  is executable in  $S_1$ ;  $move(D)$  is executable in  $S_2$ .

These are equivalent to checking whether:

- $\mathcal{D}_0 \cup \mathcal{D}_{una} \models \mathcal{R}[Poss(move(B), S_0)];$
- $\mathcal{D}_0 \cup \mathcal{D}_{una} \models \mathcal{R}[Poss(move(C), S_1)];$
- $\mathcal{D}_0 \cup \mathcal{D}_{una} \models \mathcal{R}[Poss(move(D), S_2)].$

Let's regress the formulas.

$$\begin{aligned} \bullet \quad & \mathcal{R}[Poss(move(B), S_0)] = \mathcal{R}[\exists y.(AgentAt(y, S_0) \wedge Right(y, B))] = \\ & \exists y.\mathcal{R}[AgentAt(y, S_0) \wedge Right(y, B)] = \\ & \exists y.\mathcal{R}[AgentAt(y, S_0)] \wedge Right(y, B) = \\ & \exists y.AgentAt(y, S_0) \wedge Right(y, B) \end{aligned}$$

Since  $\mathcal{D}_0 \models AgentAt(A, S_0) \wedge Right(A, B)$ , we have that:  $\mathcal{D}_0 \cup \mathcal{D}_{una} \models \exists y.(AgentAt(y, S_0) \wedge Right(y, B))$ , thus  $move(B)$  is executable in  $S_0$ .

$$\begin{aligned} \bullet \quad & \mathcal{R}[Poss(move(C), S_1)] = \mathcal{R}[\exists y.(AgentAt(y, S_1) \wedge Right(y, C))] = \\ & \exists y.\mathcal{R}[(AgentAt(y, S_1) \wedge Right(y, C))] = \\ & \exists y.\mathcal{R}[AgentAt(y, S_1)] \wedge Right(y, C) = \\ & \exists y.\mathcal{R}[AgentAt(y, S_1)] \wedge Right(y, C) = \exists y.\mathcal{R}[AgentAt(y, do(move(B), S_0))] \wedge Right(y, C) = \\ & \exists y.\mathcal{R}[move(B) = move(y) \vee (AgentAt(y, S_0) \wedge \neg((\exists z.move(B) = move(z)) \vee move(B) = exit()))) \wedge Right(y, C)] = \\ & \exists y.\mathcal{R}[move(B) = move(y) \vee (AgentAt(y, S_0) \wedge \neg(true \vee false))] \wedge Right(y, C) = \\ & \exists y.\mathcal{R}[move(B) = move(y) \vee (AgentAt(y, S_0) \wedge false)] \wedge Right(y, C) = \\ & \exists y.\mathcal{R}[move(B) = move(y) \vee false] \wedge Right(y, C) = \\ & \exists y.\mathcal{R}[move(B) = move(y)] \wedge Right(y, C) = \\ & \exists y.move(B) = move(y) \wedge Right(y, C) \end{aligned}$$

Since  $\mathcal{D}_0 \cup \mathcal{D}_{una} \models move(B) = move(B) \wedge Right(B, C)$ , we have that:  $\mathcal{D}_0 \cup \mathcal{D}_{una} \models \exists y.move(B) = move(y) \wedge Right(y, C)$  and thus  $move(C)$  is executable in  $S_1$ .

$$\bullet \quad \mathcal{R}[Poss(move(D), S_2)] = \mathcal{R}[\exists y.(AgentAt(y, S_2) \wedge Right(y, D))] = \exists y.\mathcal{R}[AgentAt(y, S_2)] \wedge Right(y, D)$$

we have that:

$$\begin{aligned} & \mathcal{R}[AgentAt(y, S_2)] = \\ & \mathcal{R}[move(C) = move(y) \vee (AgentAt(y, S_1) \wedge \neg((\exists z.move(C) = move(y)) \vee move(C) = exit())))] = \dots = \\ & \mathcal{R}[move(C) = move(y) \vee false] = \mathcal{R}[move(C) = move(y)] = move(C) = move(y) \end{aligned}$$

thus:

$$\begin{aligned} & \mathcal{R}[Poss(move(D), S_2)] = \exists y.\mathcal{R}[AgentAt(y, S_2)] \wedge Right(y, D) = \\ & \exists y.move(C) = move(y) \wedge Right(y, D) \end{aligned}$$

and, again, since  $\mathcal{D}_0 \cup \mathcal{D}_{una} \models \exists y.move(C) = move(y) \wedge Right(y, D)$ ,  $move(D)$  is executable in  $S_2$ .

To check whether  $\varrho$  results in a situation where cell  $C$  contains some item, we need to check whether:

$$\mathcal{D}_0 \cup \mathcal{D}_{una} \models \mathcal{R}[\exists x.ItemAt(x, C, S_3)]$$

Let's regress the formula:

- $\mathcal{R}[\exists x.ItemAt(x, C, S_3)] = \exists x.\mathcal{R}[ItemAt(x, C, S_3)] =$   
 $\exists x.\mathcal{R}[(move(D) = drop() \wedge AgentHas(x, S_2) \wedge AgentAt(C, S_2)) \vee (ItemAt(x, C, S_2) \wedge move(D) \neq move(C))] =$   
 $\exists x.\mathcal{R}[false \vee (ItemAt(x, C, S_2) \wedge true)] =$   
 $\exists x.\mathcal{R}[ItemAt(x, C, S_2)] =$   
 $\exists x.\mathcal{R}[(move(C) = drop() \wedge AgentHas(x, S_1) \wedge AgentAt(C, S_1)) \vee (ItemAt(x, C, S_1) \wedge move(C) \neq move(C))] =$   
 $\exists x.\mathcal{R}[false \vee (ItemAt(x, C, S_1) \wedge false)] =$   
 $\exists x.\mathcal{R}[false] = false$

Thus,  $\varrho$  does not result in a situation where cell  $C$  contains some item.

To check whether  $\varrho$  results in a situation where the agent has all the items with itself, we need to check whether:

$$\mathcal{D}_0 \cup \mathcal{D}_{una} \models \mathcal{R}[\forall x.Item(x) \supset AgentHas(x, S_3)]$$

(Alternatively, the simpler formula;  $\neg AgentHas(circle, S_3) \wedge \neg AgentHas(triangle, S_3)$  could be checked)

Let's regress the formula:

- $\mathcal{R}[\forall x.Item(x) \supset AgentHas(x, S_3)] = \forall x.\mathcal{R}[Item(x) \supset AgentHas(x, S_3)] = \forall x.\mathcal{R}[Item(x)] \supset \mathcal{R}[AgentHas(x, S_3)] =$   
 $\forall x.Item(x) \supset \mathcal{R}[AgentHas(x, S_3)]$

We have:

$$\begin{aligned} \mathcal{R}[AgentHas(x, S_3)] &= \\ \mathcal{R}[(\exists y.move(D) = move(y) \wedge ItemAt(x, y, S_2)) \vee (AgentHas(x, S_2) \wedge move(D) \neq drop())] &= \\ \mathcal{R}[(\exists y.move(D) = move(y) \wedge ItemAt(x, y, S_2)) \vee (AgentHas(x, S_2) \wedge true)] &= \\ \mathcal{R}[(\exists y.move(D) = move(y) \wedge ItemAt(x, y, S_2)) \vee (AgentHas(x, S_2))] &= \\ \mathcal{R}[(\exists y.move(D) = move(y) \wedge ItemAt(x, y, S_2)] \vee \mathcal{R}[AgentHas(x, S_2)] &= \\ \exists y.\mathcal{R}[move(D) = move(y) \wedge ItemAt(x, y, S_2)] \vee \mathcal{R}[AgentHas(x, S_2)] &= \\ (\exists y.move(D) = move(y) \wedge \mathcal{R}[ItemAt(x, y, S_2)]) \vee \mathcal{R}[AgentHas(x, S_2)] &= \end{aligned}$$

Let's regress  $ItemAt(x, y, S_2)$ :

$$\begin{aligned} \mathcal{R}[ItemAt(x, y, S_2)] &= \\ \mathcal{R}[(move(C) = drop() \wedge AgentHas(x, S_1) \wedge AgentAt(y, S_1)) \vee (ItemAt(x, y, S_1) \wedge move(C) \neq move(y))] &= \\ \mathcal{R}[(false \wedge AgentHas(x, S_1) \wedge AgentAt(y, S_1)) \vee (ItemAt(x, y, S_1) \wedge move(C) \neq move(y))] &= \\ \mathcal{R}[false \vee (ItemAt(x, y, S_1) \wedge move(C) \neq move(y))] &= \\ \mathcal{R}[ItemAt(x, y, S_1)] \wedge move(C) \neq move(y) &= \end{aligned}$$

We have:

$$\begin{aligned} \mathcal{R}[ItemAt(x, y, S_1)] &= \\ \mathcal{R}[(move(B) = drop() \wedge AgentHas(x, S_0) \wedge AgentAt(y, S_0)) \vee (ItemAt(x, y, S_0) \wedge move(B) \neq move(y))] &= \\ \mathcal{R}[(false \wedge AgentHas(x, S_0) \wedge AgentAt(y, S_0)) \vee (ItemAt(x, y, S_0) \wedge move(B) \neq move(y))] &= \\ \mathcal{R}[ItemAt(x, y, S_0) \wedge move(B) \neq move(y)] &= \\ ItemAt(x, y, S_0) \wedge move(B) \neq move(y) &= \end{aligned}$$

Thus:

$$\mathcal{R}[ItemAt(x, y, S_2)] = ItemAt(x, y, S_0) \wedge move(B) \neq move(y) \wedge move(C) \neq move(y)$$

Let's regress  $AgentHas(x, S_2)$ :

$$\begin{aligned} \mathcal{R}[AgentHas(x, S_2)] &= \\ \mathcal{R}[(\exists y.move(C) = move(y) \wedge ItemAt(x, y, S_1)) \vee (AgentHas(x, S_1) \wedge move(C) \neq drop())] &= \\ \mathcal{R}[(\exists y.move(C) = move(y) \wedge ItemAt(x, y, S_1)) \vee (AgentHas(x, S_1) \wedge move(C) \neq drop())] &= \\ \mathcal{R}[(\exists y.move(C) = move(y) \wedge ItemAt(x, y, S_1)) \vee (AgentHas(x, S_1) \wedge true)] &= \\ \mathcal{R}[(\exists y.move(C) = move(y) \wedge ItemAt(x, y, S_1)) \vee AgentHas(x, S_1)] &= \\ (\exists y.move(C) = move(y) \wedge \mathcal{R}[ItemAt(x, y, S_1)]) \vee \mathcal{R}[AgentHas(x, S_1)] &= \end{aligned}$$

$\mathcal{R}[ItemAt(x, y, S_1)]$  was computed above.

For  $AgentHas(x, S_1)$ , we have:

$$\begin{aligned} \mathcal{R}[AgentHas(x, S_1)] &= \\ \mathcal{R}[(\exists y.move(B) = move(y) \wedge ItemAt(x, y, S_0)) \vee (AgentHas(x, S_0) \wedge move(B) \neq drop())] &= \\ \mathcal{R}[(\exists y.move(B) = move(y) \wedge ItemAt(x, y, S_0)) \vee (AgentHas(x, S_0) \wedge move(B) \neq drop())] &= \end{aligned}$$

$$\begin{aligned}\mathcal{R}[(\exists y.move(B) = move(y) \wedge ItemAt(x, y, S_0)) \vee (AgentHas(x, S_0) \wedge true)] &= \\ \mathcal{R}[(\exists y.move(B) = move(y) \wedge ItemAt(x, y, S_0)) \vee AgentHas(x, S_0)] &= \\ (\exists y.move(B) = move(y) \wedge ItemAt(x, y, S_0)) \vee AgentHas(x, S_0)\end{aligned}$$

Thus, we obtain:

$$\begin{aligned}\mathcal{R}[AgentHas(x, S_2)] &= \\ (\exists y.move(C) = move(y) \wedge ItemAt(x, y, S_0) \wedge move(B) \neq move(y)) \vee \\ (\exists y.move(B) = move(y) \wedge ItemAt(x, y, S_0)) \vee AgentHas(x, S_0)\end{aligned}$$

$$\begin{aligned}\text{Hence: } \mathcal{R}[AgentHas(x, S_3)] &= \\ (\exists y.move(D) = move(y) \wedge ItemAt(x, y, S_0) \wedge move(B) \neq move(y) \wedge move(C) \neq move(y)) \vee \\ (\exists y.move(C) = move(y) \wedge ItemAt(x, y, S_0) \wedge move(B) \neq move(y)) \vee \\ (\exists y.move(B) = move(y) \wedge ItemAt(x, y, S_0)) \vee AgentHas(x, S_0)\end{aligned}$$

Putting these results together, we obtain:

$$\begin{aligned}\mathcal{R}[\forall x.Item(x) \supset AgentHas(x, S_3)] &= \\ \forall x.Item(x) \supset \\ (\exists y.move(D) = move(y) \wedge ItemAt(x, y, S_0) \wedge \neg(move(B) = move(y)) \wedge move(C) \neq move(y)) \vee \\ (\exists y.move(C) = move(y) \wedge ItemAt(x, y, S_0) \wedge move(B) \neq move(y)) \vee \\ (\exists y.move(B) = move(y) \wedge ItemAt(x, y, S_0)) \vee AgentHas(x, S_0)\end{aligned}$$

Since we have that *circle* and *triangle* are the only items and:

$$\begin{aligned}\mathcal{D}_0 \cup \mathcal{D}_{una} \models Item(circle) \supset (move(B) = move(B) \wedge ItemAt(circle, B, S_0)) \text{ and} \\ \mathcal{D}_0 \cup \mathcal{D}_{una} \models Item(triangle) \supset (move(C) = move(C) \wedge ItemAt(triangle, C, S_0) \wedge move(B) \neq move(C))\end{aligned}$$

then:

$$\begin{aligned}\mathcal{D}_0 \cup \mathcal{D}_{una} \models \\ \forall x.Item(x) \supset \\ (\exists y.move(D) = move(y) \wedge ItemAt(x, y, S_0) \wedge move(B) \neq move(y) \wedge move(C) \neq move(y)) \vee \\ (\exists y.move(C) = move(y) \wedge ItemAt(x, y, S_0) \wedge move(B) \neq move(y)) \vee \\ (\exists y.move(B) = move(y) \wedge ItemAt(x, y, S_0)) \vee AgentHas(x, S_0)\end{aligned}$$

Thus,  $\varrho$  results in a situation where the agent has all the items with itself.

*PDDL*:

Domain file:

```
(define (domain grid_domain)
  (:requirements :adl)
  (:types location item)
  (:predicates
    (item ?x-item)
    (right ?x1 ?x2-location)
    (exit ?x-location)
    (agentAt ?x-location)
    (agentHas ?x-item)
    (itemAt ?x-item ?y-location)
  )
  (:action move
    :parameters (?x - location)
    :precondition ((exists ?y - location) (and (agentAt ?y) (right ?y ?x)))
    :effect (and
      (agentAt ?x)
      (forall (?y-location) (when (agentAt ?y) (not (agentAt ?y))))
      (forall (?y-item) (when (itemAt ?y ?x) (and (not (itemAt ?y ?x)) (agentHas (?y))))))
    )
  )
```

```

); end of move

(:action drop
:parameters ()
:precondition ()
:effect
  (forall (?x-location)
    (forall (?y-item)
      (when
        (and (agentAt ?x) (agentHas ?y))
        (and (itemAt ?y ?x) (not (agentHas ?y))))
      )
    )
  )
); end of drop

(:action exit
:parameters ()
:precondition (and
  (exists (?x - location) (and (agentAt ?x) (exit ?x)))
  (forall (?x - item) (agentHas ?x))
)
:effect (forall (?x-location) (not (agentAt ?x)))
); end of exit
); end of define

```

**Problem file:**

```

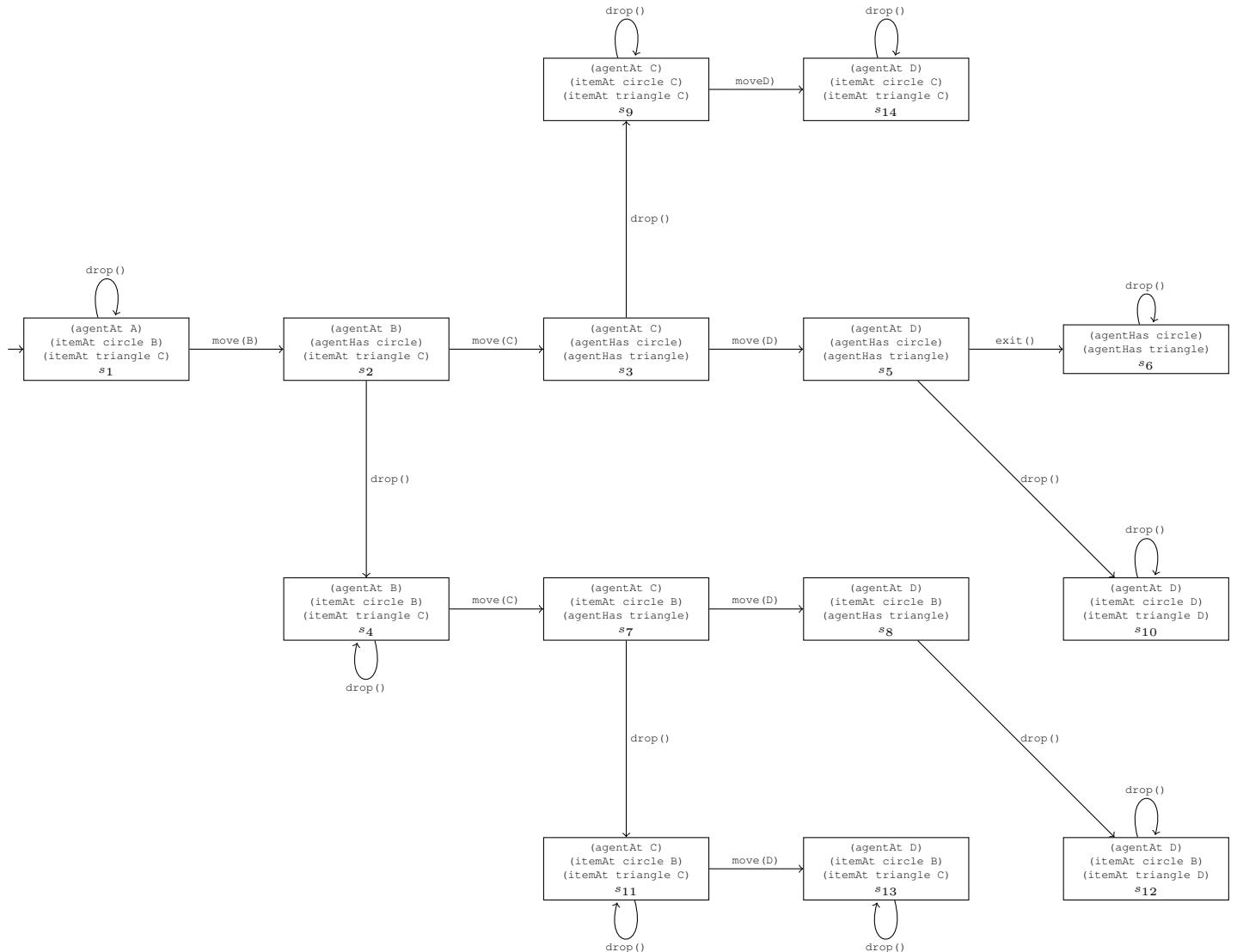
(define (problem grid_problem) (:domain grid_domain)
  (:objects
    A B C D - location
    circle triangle - item
  )

  (:init
    (right A B) (right B C) (right C D)
    (exit D)
    (agentAt A)
    (itemAt circle B) (itemAt triangle C)
  )

  (:goal (and (itemAt circle D) (itemAt triangle D)))
)

```

*Transition System:*



*DFS Algorithm:*

```

DFS(domain d, state init, formula goal){
    // d: input domain;
    // init: initial state;
    // goal: goal formula;
    Stack t = [(init, empty)]; // Open set (stack)
    Set m = {init}; // Marked states
    while (!t.empty()) {
        (state, plan) = t.pop(); // plan is action sequence followed to reach s
        if (s ⊨ goal) return plan; // if s satisfies goal state, return plan
        forall (actions a executable in s)
            s'=d.delta(s,a) // s' is successor of s under a (d.delta is transition function of d)
            if (s' ∉ m) { // if s' not marked
                m.add(s'); // mark s'
                t.push((s', plan·a)); // add s' with plan extended by a to open set t}
    }
    return noplans; // no plan found
}

```

*Current node, open set and marked states at end of every iteration of call*

DFS(grid\_domain,  $s_1$ , (itemAt circle D)  $\wedge$  (itemAt triangle D)):

0.  $t = [(s_1, \text{empty})]$   
 $m = \{s_1\}$
1.  $(\text{state}, \text{plan}) = (s_1, \text{empty})$   
 $t = [(s_2, \text{move(B)})]$   
 $m = \{s_1, s_2\}$
2.  $(\text{state}, \text{plan}) = (s_2, \text{move(B)})$   
 $t = [(s_3, \text{move(B)} \text{ move(C)}),$   
 $(s_4, \text{move(B)} \text{ drop()})]$   
 $m = \{s_1, s_2, s_3, s_4\}$
3.  $(\text{state}, \text{plan}) = (s_3, \text{move(B)} \text{ move(C)})$   
 $t = [(s_5, \text{move(B)} \text{ move(C)} \text{ move(D)}),$   
 $(s_9, \text{move(B)} \text{ move(C)} \text{ drop()}),$   
 $(s_4, \text{move(B)} \text{ drop()})]$   
 $m = \{s_1, s_2, s_3, s_4, s_5, s_9\}$
4.  $(\text{state}, \text{plan}) = (s_5, \text{move(B)} \text{ move(C)} \text{ move(D)})$   
 $t = [(s_{10}, \text{move(B)} \text{ move(C)} \text{ move(D)} \text{ drop()}),$   
 $(s_6, \text{move(B)} \text{ move(C)} \text{ move(D)} \text{ exit()}),$   
 $(s_9, \text{move(B)} \text{ move(C)} \text{ drop()}),$   
 $(s_4, \text{move(B)} \text{ drop()})]$   
 $m = \{s_1, s_2, s_3, s_4, s_5, s_9, s_{10}\}$
5.  $(\text{state}, \text{plan}) = ((s_{10}, \text{move(B)} \text{ move(C)} \text{ move(D)} \text{ drop()}))$   
return  $(\text{move(B)} \text{ move(C)} \text{ move(D)} \text{ drop()})$

Solution plan is:  $(\text{move(B)} \text{ move(C)} \text{ move(D)} \text{ drop()})$

FOL:

Predicates:

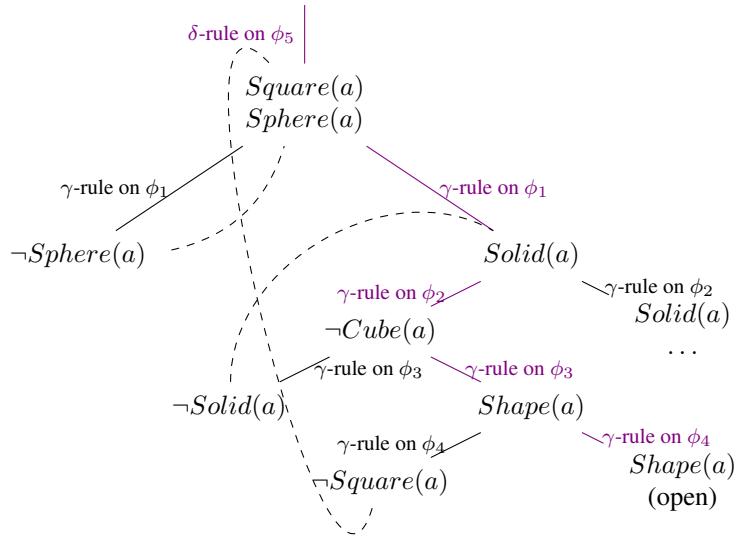
- $Sphere(x)$ :  $x$  is a sphere
- $Cube(x)$ :  $x$  is a cube
- $Solid(x)$ :  $x$  is a solid
- $Square(x)$ :  $x$  is a square
- $Shape(x)$ :  $x$  is a geometric shape

Formulas:

- $(\phi_1)$  Every sphere is a solid:  $\forall x. Sphere(x) \supset Solid(x)$
- $(\phi_2)$  Every cube is a solid:  $\forall x. Cube(x) \supset Solid(x)$
- $(\phi_3)$  Every solid is a geometric shape:  $\forall x. Solid(x) \supset Shape(x)$
- $(\phi_4)$  Every square is a geometric shape:  $\forall x. Square(x) \supset Shape(x)$
- $(\phi_5)$  It is possible that a square is also a sphere:  $\exists x. Square(x) \wedge Sphere(x)$

Let  $\mathcal{K} = \{\phi_1, \phi_2, \phi_3, \phi_4\}$ . We need to check whether  $\mathcal{K} \cup \{\phi_5\}$  is satisfiable. Let's construct the tableau for  $\mathcal{K} \cup \{\phi_5\}$ :

- $(\phi_1) \forall x. Sphere(x) \supset Solid(x)$
- $(\phi_2) \forall x. Cube(x) \supset Solid(x)$
- $(\phi_3) \forall x. Solid(x) \supset Shape(x)$
- $(\phi_4) \forall x. Square(x) \supset Shape(x)$
- $(\phi_5) \exists x. Square(x) \wedge Sphere(x)$



The coloured path gives us a model  $I$  of  $\mathcal{K}$ :

- $\Delta^I = \{a\}$
- $Sphere^I = \{a\}$
- $Cube^I = \{\}$
- $Solid^I = \{a\}$
- $Square^I = \{a\}$
- $Shape^I = \{a\}$