SAPIENZA Università di Roma – MSc. in Engineering in Computer Science
Artificial Intelligence & Machine Learning – June 10, 2024

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

Consider the following scenario. An agent (☺) 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 (○) and a triangle (△). Cell D is the exit (⇨). The initial situation is depicted below:

A	B	C	D
☺	○	△	⇨

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. ϱ is executable in S_0;
2. ϱ results in a situation where cell C contains some item;
3. ϱ 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.

1. Formalize the following statements in FOL (by introducing suitable predicates, functions and constants):
   - (ϕ_1) Every sphere is a solid.
   - (ϕ_2) Every cube is a solid.
   - (ϕ_3) Every solid is a geometric shape.
   - (ϕ_4) Every square is a geometric shape.
2. Considering the KB $\mathcal{K} = \{\phi_1, \ldots, \phi_4\}$ 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:

1. Start with *Effect Axioms*:
   - $a = move(x) \supset (AgentAt(x, do(a, s)) \wedge \forall y.(AgentAt(y, s) \supset \neg AgentAt(y, do(a, s))) \wedge$
     $\forall z.(ItemAt(z, x, s) \supset (AgentHas(z, do(a, s)) \wedge \neg ItemAt(z, x, do(a, s)))))$
   - $a = drop() \supset ((\forall x.(AgentHas(x, s) \supset \neg agentHas(x, do(a, s)))) \wedge$
     $(\forall x \forall y.(AgentHas(x, s) \wedge AgentAt(y, s)) \supset ItemAt(x, y, do(a, s))))$
   - $a = exit() \supset \forall x.(AgentAt(x, s) \supset \neg AgentAt(x, do(a, s)))$
2. Normalize:
   - $a = move(x) \supset AgentAt(x, do(a, s))$ $\wedge x \neq y$
   - $(\exists x.a = move(x) \wedge AgentAt(y, s)) \supset \neg AgentAt(y, do(a, s))$
   - $(\exists x.a = move(x) \wedge ItemAt(z, x, s)) \supset AgentHas(z, do(a, s))$
   - $(a = move(x) \wedge ItemAt(z, x, s)) \supset \neg ItemAt(z, x, do(a, s))$
   - $(a = drop() \wedge AgentHas(x, s) \wedge AgentAt(y, s)) \supset ItemAt(x, y, do(a, s))$
   - $(a = drop() \wedge AgentHas(x, s)) \supset \neg agentHas(x, do(a, s))$
   - $(a = exit() \wedge AgentAt(x, s)) \supset \neg AgentAt(x, do(a, s))$
3. 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:

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

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 ϱ 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.

- $\mathcal{R}[Poss(move(B), S_0)] = \mathcal{R}[\exists y.(AgentAt(y, S_0) \wedge Right(y, B))] =$
  $\exists y.\mathcal{R}[AgentAt(y, S_0) \wedge Right(y, B)] =$
  $\exists y.\mathcal{R}[AgentAt(y, S_0)] \wedge Right(y, B) =$
  $\exists y.AgentAt(y, S_0) \wedge Right(y, B)$

  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.

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

  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.

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

  we have that:
  $\mathcal{R}[AgentAt(y, S_2)] =$
  $\mathcal{R}[move(C) = move(y) \vee (AgentAt(y, S_1) \wedge \neg((\exists z.move(C) = move(y)) \vee move(C) = exit()))] = \ldots =$
  $\mathcal{R}[move(C) = move(y) \vee false] = \mathcal{R}[move(C) = move(y)] = move(C) = move(y)$

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

  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 ϱ 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:

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

REGRESSION

$\rho =$ move (B); move (C); move (D);

1) ρ IS EXECUTABLE IN S_0 IF:

a move (B) IS EXECUTABLE IN S_0 $\rightarrow$ $D \models Poss(move(B), S_0)$

b move (C) IS EXECUTABLE IN S_1 $\rightarrow$ $D \models Poss(move(C), do(\underbrace{move(B), S_0}_{S_1}))$

c move (D) IS EXECUTABLE IN S_2 $\rightarrow$ $D \models Poss(move(D), do(\underbrace{move(C), do(move(B), S_0)}_{S_2})))$

a) $Poss(move(B), S_0) \leftrightarrow \exists y.\ AgentAt(y, S_0) \land Right(y, B)$

$R[Poss(move(B), S_0)] = R[\exists y.\ AgentAt(y, S_0) \land Right(y, B)] =$
$= \exists y.\ R[AgentAt(y, S_0) \land Right(y, B)] =$
$= \exists y.\ R[AgentAt(y, S_0)] \land R[Right(y, B)] =$
$= \exists y.\ AgentAt(y, S_0) \land Right(y, B)$

REGRESSION THEOREM: $D \models Poss(move(B), S_0) \Leftrightarrow D_{S_0} \cup D_{una} \models R[Poss(move(B), S_0)]$

$D_{S_0} \cup D_{una} \models \exists y.\ AgentAt(y, S_0) \land Right(y, B)$

$\hookrightarrow$ YES, y EXISTS $(y = A)$

$\exists y.\ AgentAt(A, S_0) \land Right(A, B)$ TRUE

b) $D \models Poss(move(C), do(move(B), S_0))$?

$Poss(move(C), S_1) \leftrightarrow \exists y.\ AgentAt(y, S_1) \land Right(y, C)$

$Poss(move(C), do(move(B), S_0)) \leftrightarrow \exists y.\ AgentAt(y, do(move(B), S_0)) \land Right(y, C)$

$R[Poss(move(C), do(move(B), S_0))] =$

$= R[\exists y.\ AgentAt(y, do(move(B), S_0)) \land Right(y, C)] =$

$= \exists y.\ R[AgentAt(y, do(move(B), S_0)) \land Right(y, C)] =$

$= \exists y.\ \mathcal{R}[\mathrm{AGENTAT}(y, do(move(B), S_0))] \wedge \mathcal{R}[\mathrm{RIGHT}(y,c)] =$

$= \exists y.\ \mathcal{R}[\mathrm{AGENTAT}(y, do(move(B), S_0))] \wedge \mathrm{RIGHT}(y,c) =$

$F(\vec{t}, do(\alpha, \sigma))$

$\mathrm{AGENTAT}(x, do(a,s)) \leftrightarrow a = move(x) \vee \mathrm{AGENT}(x,s) \wedge$
$((\neg \exists z\ a = move(z) \wedge z \neq x) \vee a = \mathrm{EXIT}())$

$$= \exists y.\ \mathcal{R}[\underbrace{move(B) = move(y)}_{\phi} \vee \underbrace{\mathrm{AGENTAT}(y, S_0) \wedge ((\neg \exists z.\ move(B) = move(z) \wedge z \neq y) \vee move(B) = \mathrm{EXIT}())}_{\psi}] \wedge \mathrm{RIGHT}(y,C) =$$

$$= \exists y. \mathcal{R}[move(B) = move(y)] \vee \mathcal{R}[\mathrm{AGENTAT}(y,S_0) \wedge ((\neg \exists z.\ move(B) = move(z) \wedge z \neq y) \vee move(B) = \mathrm{EXIT}())] \wedge \mathrm{RIGHT}(y,C) =$$

$$= \exists y.\ move(B) = move(y) \vee (\mathcal{R}[\mathrm{AGENTAT}(y,S_0)] \wedge \mathcal{R}[((\neg \exists z.\ move(B) = move(z) \wedge z \neq y) \vee move(B) = \mathrm{EXIT}())]) \wedge \mathrm{RIGHT}(y,C) =$$

$$= \exists y.\ move(B) = move(y) \vee (\mathrm{AGENTAT}(y,S_0) \wedge ((\neg \exists z.\ move(B) = move(z) \wedge z \neq y) \vee \cancel{move(B) = \mathrm{EXIT}()})) \wedge \mathrm{RIGHT}(y,C) =$$

$$= \exists y.\ move(B) = move(y) \vee \mathrm{AGENTAT}(y,S_0) \wedge (\neg \exists z.\ move(B) = move(z) \wedge z \neq y) \wedge \mathrm{RIGHT}(y,C) =$$

$$D_{S_0} \cup D_{una} \models \exists y.\ move(B) = move(y) \vee \mathrm{AGENTAT}(y,S_0) \wedge (\neg \exists z.\ move(B) = move(z) \wedge z \neq y) \wedge \mathrm{RIGHT}(y,C) =$$

FOR $y = B$
$$\left.\begin{array}{l} \exists y.\ move(B) = move(y) \vee \mathrm{AGENTAT}(y,S_0) \wedge \\ (\neg \exists z.\ move(B) = move(z) \wedge z \neq y) \\ \mathrm{RIGHT}(y,C) \end{array}\right\} \text{TRUE}$$

$D \models Poss(move(c), do(move(B), S_0))$ TRUE

c) $D \models Poss(move(D), do(move(C), do(move(B), S_0)))$?

$Poss(move(D), S_2) \leftrightarrow \exists y.\ \mathrm{AGENTAT}(y, S_2) \wedge \mathrm{RIGHT}(y, D)$

$Poss(move(D), do(move(C), do(move(B), S_0))) \leftrightarrow$
$\leftrightarrow \exists y.\ \mathrm{AGENTAT}(y, do(move(C), do(move(B), S_0))) \wedge \mathrm{RIGHT}(y, D)$

$D_{S_0} \cup D_{una} \models \mathcal{R}[Poss(move(D), do(move(C), do(move(B), S_0)))] =$

$= \mathcal{R}[\exists y.\ \mathrm{AGENTAT}(y, do(move(C), do(move(B), S_0))) \wedge \mathrm{RIGHT}(y, D)] =$

$= \exists y. \left(R\left[\text{AGENTAT}(y, do(move(C), do(move(B), S_0))) \right] \wedge R\left[\text{RIGHT}(y, D) \right] \right) =$

$= \exists y. \left(R\left[\text{AGENTAT}(y, do(move(C), do(move(B), S_0))) \right] \wedge \text{RIGHT}(y, D) \right) =$

$R[\text{AGENTAT}(y, do(\overset{a}{move(C)}, \overset{s}{S_1}))] =$

$\updownarrow$

$move(C) = move(y) \vee \left(\text{AGENTAT}(y, S_1) \wedge \neg\left((\exists z.\ z \neq y \wedge move(C) = move(z)) \vee \cancel{(move(C) = \text{EXIT}())} \right) \right)$

$= R\left[move(C) = move(y) \vee \underline{\left(\text{AGENTAT}(y, S_1) \wedge \neg(\exists z.\ z \neq y \wedge move(C) = move(z)) \right)} \right] =$

$= R[move(C) = move(y)] \vee R[\psi]$

$= \exists y. \left((move(C) = move(y)) \vee R[\psi] \right) \wedge \text{RIGHT}(y, D)$

$D_{S_0} \cup D_{una} = \exists y. \left((move(C) = move(y)) \vee R[\psi] \right) \wedge \text{RIGHT}(y, D)$

TRUE IF $y = C$

SINCE a, b AND c HOLD, p IS EXECUTABLE

2) CELL C CONTAINS SOME ITEM

$$\begin{cases} p = move(B);\ move(C);\ move(D); \\ \phi(s) = \exists y.\ \text{ITEMAT}(y, C, s) \end{cases}$$

$S_3 = do(move(D), do(move(C), do(move(B), S_0)))$ $\qquad D \models \phi$?

$D \models \phi \leftrightarrow D_{S_0} \cup D_{una} \models R[\phi] = R[\exists y.\ \text{ITEMAT}(y, C, S_3)] =$

$= \exists y.\ R[\text{ITEMAT}(y, C, S_3)] = \exists y.\ R[\text{ITEMAT}(y, C, do(move(D), S_2))] =$

SSA FOR ITEMAT:

$\text{ITEMAT}(y, x, do(a, s)) \leftrightarrow (a = \text{DROP}() \wedge \text{AGENTHAS}(y, s) \wedge \text{AGENTAT}(x, s)) \vee (\text{ITEMAT}(y, x, s) \wedge a \neq move(x))$

$\text{ITEMAT}(y, C, do(move(D), S_2)) \leftrightarrow (move(D) = \text{DROP}() \wedge \text{AGENTHAS}(y, S_2) \wedge \text{AGENTAT}(C, S_2)) \vee (\text{ITEMAT}(y, C, S_2) \wedge move(D) \neq move(C))$

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

$move(D) = \text{DROP}()$ IS FALSE, SO

$(move(D) = \text{DROP}() \wedge \text{AGENTHAS}(y, S_2) \wedge \text{AGENTAT}(C, S_2))$ IS FALSE

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

$= \exists y.\ R[\text{ITEMAT}(y, c, S_2)] = \exists y.\ R[\text{ITEMAT}(y, c, do(move(c), S_1))] =$

$= \exists y.\ R[\text{ITEMAT}(y, c, do(move(c), S_1))] =$

$\text{ITEMAT}(y, x, do(a, s)) \leftrightarrow (a = \text{DROP}() \wedge \text{AGENTHAS}(y, s) \wedge \text{AGENTAT}(x, s)) \vee$
$(\text{ITEMAT}(y, x, s) \wedge a \neq move(x))$

$\text{ITEMAT}(y, c, do(move(c), S_1)) \leftrightarrow (move(c) = \text{DROP}() \wedge \text{AGENTHAS}(y, S_1) \wedge$
$\text{AGENTAT}(c, S_1)) \vee (\text{ITEMAT}(y, c, S_1) \wedge move(c) \neq move(c))$

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

$move(c) \neq move(c)$ IS FALSE, SO
$(\text{ITEMAT}(y, c, S_1) \wedge move(c) \neq move(c))$ IS FALSE

$move(c) = \text{DROP}()$ IS FALSE, SO
$(move(c) = \text{DROP}() \wedge \text{AGENTHAS}(y, S_1) \wedge \text{AGENTAT}(c, S_1))$ IS FALSE

$= \exists y.\ R[\text{FALSE}] = \text{FALSE}$

SO $R[\phi] = \exists y.\ \perp (\text{FALSE}) = \perp$

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

Thus, ϱ does not result in a situation where cell C contains some item.

To check whether ϱ 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:
  $\mathcal{R}[AgentHas(x, S_3)] =$
  $\mathcal{R}[(\exists y.move(D) = move(y) \wedge ItemAt(x, y, S_2)) \vee (AgentHas(x, S_2) \wedge move(D) \neq drop())] =$
  $\mathcal{R}[(\exists y.move(D) = move(y) \wedge ItemAt(x, y, S_2)) \vee (AgentHas(x, S_2) \wedge true)] =$
  $\mathcal{R}[(\exists y.move(D) = move(y) \wedge ItemAt(x, y, S_2)) \vee (AgentHas(x, S_2))] =$
  $\mathcal{R}[\exists y.move(D) = move(y) \wedge ItemAt(x, y, S_2)] \vee \mathcal{R}[AgentHas(x, S_2)] =$
  $\exists y.\mathcal{R}[move(D) = move(y) \wedge ItemAt(x, y, S_2)] \vee \mathcal{R}[AgentHas(x, S_2)] =$
  $(\exists y.move(D) = move(y) \wedge \mathcal{R}[ItemAt(x, y, S_2)]) \vee \mathcal{R}[AgentHas(x, S_2)]$

  Let's regress $ItemAt(x, y, S_2)$:
  $\mathcal{R}[ItemAt(x, y, S_2)] =$
  $\mathcal{R}[(move(C) = drop() \wedge AgentHas(x, S_1) \wedge AgentAt(y, S_1)) \vee (ItemAt(x, y, S_1) \wedge move(C) \neq move(y))] =$
  $\mathcal{R}[(false \wedge AgentHas(x, S_1) \wedge AgentAt(y, S_1)) \vee (ItemAt(x, y, S_1) \wedge move(C) \neq move(y))] =$
  $\mathcal{R}[false \vee (ItemAt(x, y, S_1) \wedge move(C) \neq move(y))] =$
  $\mathcal{R}[ItemAt(x, y, S_1)] \wedge move(C) \neq move(y)$

  We have:
  $\mathcal{R}[ItemAt(x, y, S_1)] =$
  $\mathcal{R}[(move(B) = drop() \wedge AgentHas(x, S_0) \wedge AgentAt(y, S_0)) \vee (ItemAt(x, y, S_0) \wedge move(B) \neq move(y))] =$
  $\mathcal{R}[(false \wedge AgentHas(x, S_0) \wedge AgentAt(y, S_0)) \vee (ItemAt(x, y, S_0) \wedge move(B) \neq move(y))] =$
  $\mathcal{R}[ItemAt(x, y, S_0) \wedge move(B) \neq move(y)] =$
  $ItemAt(x, y, S_0) \wedge move(B) \neq move(y)$

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

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

  For $AgentHas(x, S_1)$, we have:
  $\mathcal{R}[AgentHas(x, S_1)] =$
  $\mathcal{R}[(\exists y.move(B) = move(y) \wedge ItemAt(x, y, S_0)) \vee (AgentHas(x, S_0) \wedge move(B) \neq drop())] =$
  $\mathcal{R}[(\exists y.move(B) = move(y) \wedge ItemAt(x, y, S_0)) \vee (AgentHas(x, S_0) \wedge move(B) \neq drop())] =$

$\mathcal{R}[(\exists y.move(B) = move(y) \land ItemAt(x, y, S_0)) \lor (AgentHas(x, S_0) \land true)] =$
$\mathcal{R}[(\exists y.move(B) = move(y) \land ItemAt(x, y, S_0)) \lor AgentHas(x, S_0)] =$
$(\exists y.move(B) = move(y) \land ItemAt(x, y, S_0)) \lor AgentHas(x, S_0)$

Thus, we obtain:
$\mathcal{R}[AgentHas(x, S_2)] =$
$$(\exists y.move(C) = move(y) \land ItemAt(x, y, S_0) \land move(B) \neq move(y)) \lor$$
$$(\exists y.move(B) = move(y) \land ItemAt(x, y, S_0)) \lor AgentHas(x, S_0)$$

Hence: $\mathcal{R}[AgentHas(x, S_3)] =$
$$(\exists y.move(D) = move(y) \land ItemAt(x, y, S_0) \land move(B) \neq move(y) \land move(C) \neq move(y)) \lor$$
$$(\exists y.move(C) = move(y) \land ItemAt(x, y, S_0) \land move(B) \neq move(y)) \lor$$
$$(\exists y.move(B) = move(y) \land ItemAt(x, y, S_0)) \lor AgentHas(x, S_0))$$

Putting these results together, we obtain:
$\mathcal{R}[\forall x.Item(x) \supset AgentHas(x, S_3)] =$
$$\forall x.Item(x) \supset$$
$$(\exists y.move(D) = move(y) \land ItemAt(x, y, S_0) \land \neg(move(B) = move(y)) \land move(C) \neq move(y)) \lor$$
$$(\exists y.move(C) = move(y) \land ItemAt(x, y, S_0) \land move(B) \neq move(y)) \lor$$
$$(\exists y.move(B) = move(y) \land ItemAt(x, y, S_0)) \lor AgentHas(x, S_0)$$

Since we have that *circle* and *triangle* are the only items and:
$\mathcal{D}_0 \cup \mathcal{D}_{una} \models Item(circle) \supset (move(B) = move(B) \land ItemAt(circle, B, S_0))$ and
$\mathcal{D}_0 \cup \mathcal{D}_{una} \models Item(triangle) \supset (move(C) = move(C) \land ItemAt(triangle, C, S_0) \land move(B) \neq move(C))$

then:
$\mathcal{D}_0 \cup \mathcal{D}_{una} \models$
$$\forall x.Item(x) \supset$$
$$(\exists y.move(D) = move(y) \land ItemAt(x, y, S_0) \land move(B) \neq move(y) \land move(C) \neq move(y)) \lor$$
$$(\exists y.move(C) = move(y) \land ItemAt(x, y, S_0) \land move(B) \neq move(y)) \lor$$
$$(\exists y.move(B) = move(y) \land ItemAt(x, y, S_0)) \lor AgentHas(x, S_0)$$

Thus, ϱ 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
```

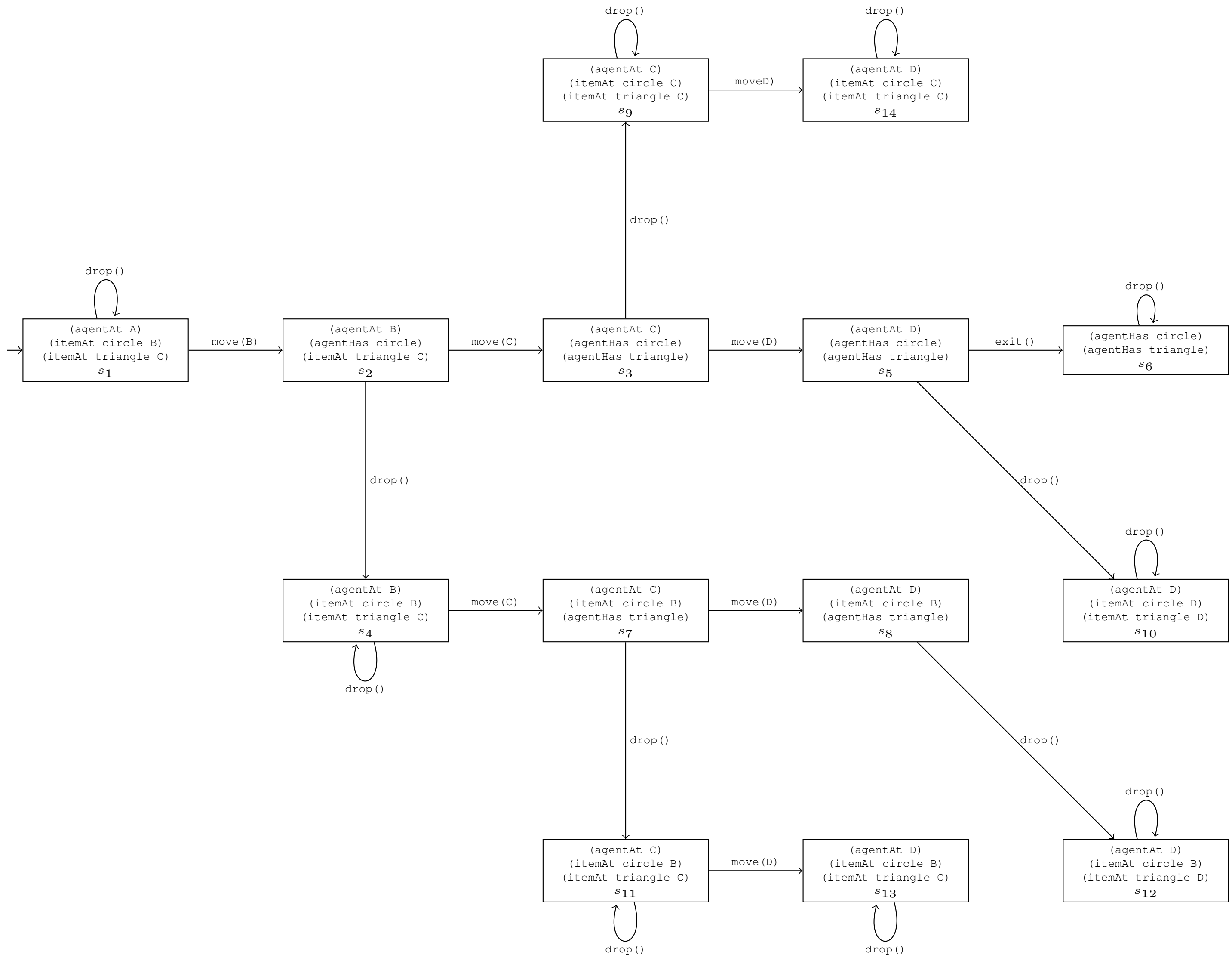
Problem file:

```
(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 satifies 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 noplan; // 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`, empty)]`
   `m = `$\{s_1\}$

1. `(state,plan)=(`s_1`, empty)`
   `t = [(`s_2`, move(B))]`
   `m = `$\{s_1, s_2\}$

2. `(state,plan)=(`s_2`, move(B))`
   `t = [(`s_3`, move(B) move(C)),`
   `      (`s_4`, move(B) drop())]`
   `m = `$\{s_1, s_2, s_3, s_4\}$

3. `(state,plan)=(`s_3`, move(B) move(C))`
   `t = [(`s_5`, move(B) move(C) move(D)),`
   `      (`s_9`, move(B) move(C) drop()),`
   `      (`s_4`, move(B) drop())]`
   `m = `$\{s_1, s_2, s_3, s_4, s_5, s_9\}$

4. `(state,plan)=(`s_5`, move(B) move(C) move(D))`
   `t = [(`s_{10}`, move(B) move(C) move(D) drop()),`
   `      (`s_6`, move(B) move(C) move(D) exit()),`
   `      (`s_9`, move(B) move(C) drop()),`
   `      (`s_4`, move(B) drop())]`
   `m = `$\{s_1, s_2, s_3, s_4, s_5, s_9, s_{10}\}$

5. `(state,plan)=((`s_{10}`, move(B) move(C) move(D) drop())`
   `return (move(B) move(C) move(D) drop())`

Solution plan is: `(move(B) move(C) move(D) drop())`

FOL:

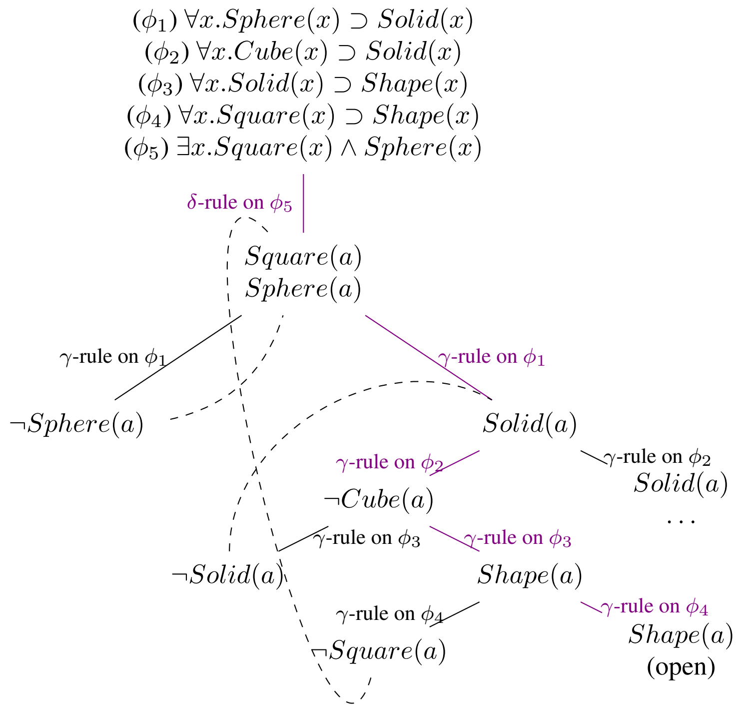
Predicates:

- $Sphere(x)$: 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:

- (ϕ_1) Every sphere is a solid: $\forall x.Sphere(x) \supset Solid(x)$
- (ϕ_2) Every cube is a solid: $\forall x.Cube(x) \supset Solid(x)$
- (ϕ_3) Every solid is a geometric shape: $\forall x.Solid(x) \supset Shape(x)$
- (ϕ_4) Every square is a geometric shape: $\forall x.Square(x) \supset Shape(x)$
- (ϕ_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 $\mathcal{K} \cup \{\phi_5\}$:



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

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