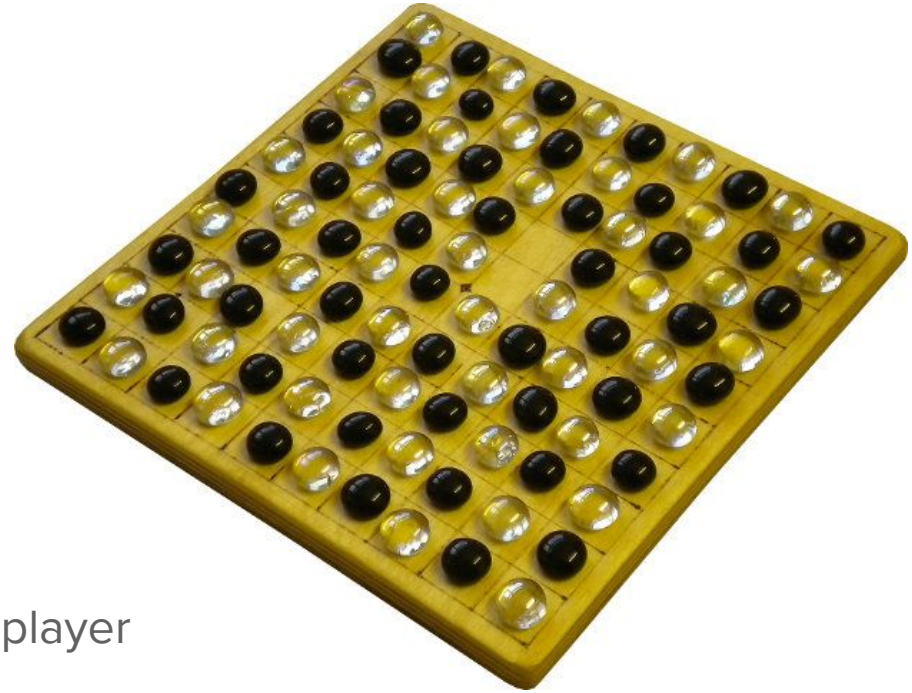
Classical Planning

willie
(some slides adapted from Stavros Vassos, University of Athens)

But first...

Konane



Hawaiian checkers

Take turns capturing pieces

Your task: Create an intelligent Konane player

Starter code: We provide implementation of the game, UI, sample players

Due in 2 weeks (11/12), but don’t wait because it is good practice for the exam

Pass the ball

Pass the ball to another person

The last letter of your (first) name must match the first letter of the other person’s name

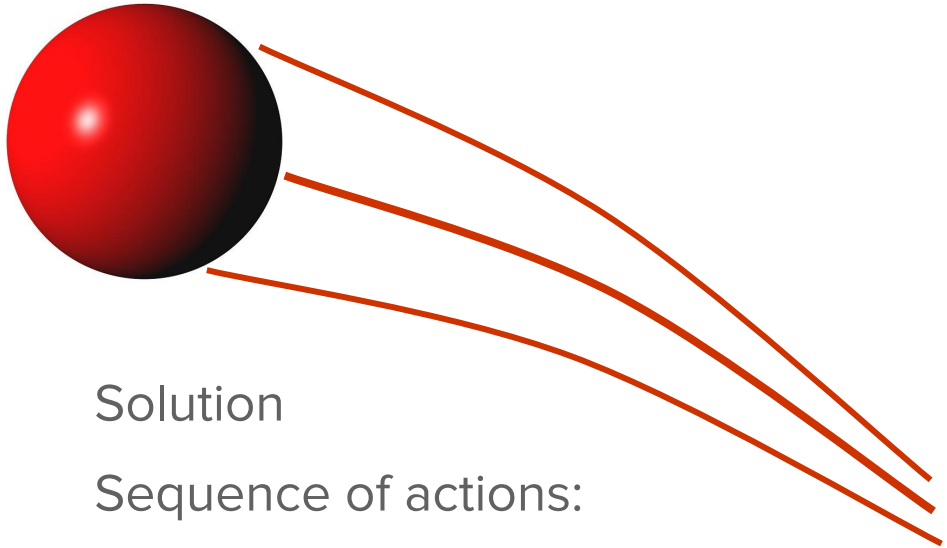
Get the ball to me (Willie)

Who has the ball?

Pass the ball

Define the problem:

- Initial state
  - A person has the ball
- Goal test
  - Does Willie have the ball?
- Actions/Successor Function
  - Previous person does not have ball
  - Next person does have ball
- Path Cost
  - All actions have same cost: 1



Solution

Sequence of actions:

1. Pass ball from a to b
2. Pass ball from b to c

   ...

n. Pass ball from v to willie

Defining another problem: Sokoban

Define the problem:

- Initial state

Sokoban

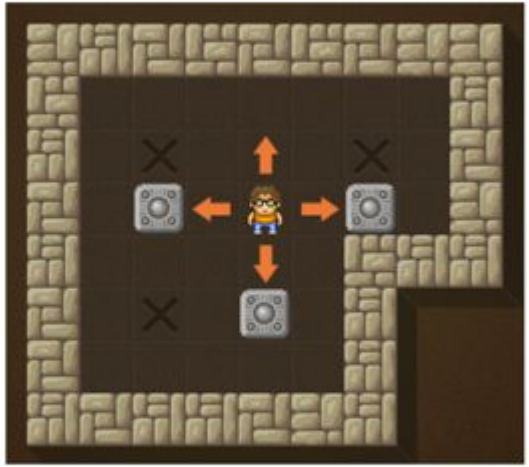
Define the problem:

- Initial state
- Goal

Sokoban

Define the problem:

- Initial state
- Goal
- Available actions



Sokoban

Define the problem:

- Initial state
- Goal
- Available actions
- Path cost: 1

Sokoban

Define the problem:

- Initial state
- Goal
- Available actions
- Path cost: 1

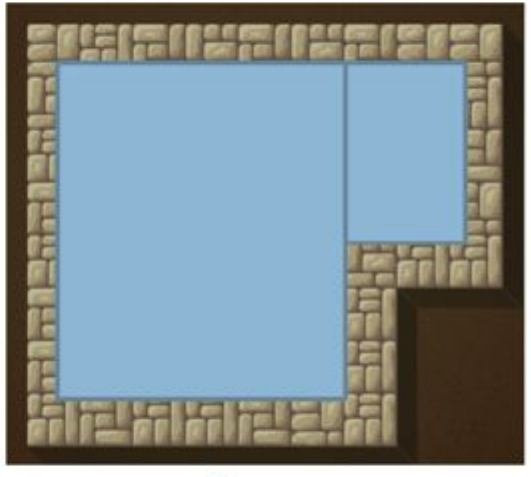
Find:

- Sequence of actions that achieves the goal (e.g, [Left, Down, Left, ...]

Sokoban

Define the problem:

- Initial state
- Goal
- Available actions
- Path cost: 1



Find:

- Sequence of actions that achieves the goal (e.g, [Left, Down, Left, ...]
- Method capable of finding a solution for every initial state and goal

Sokoban

Define the problem:

- Initial state
- Goal
- Available actions
- Path cost: 1

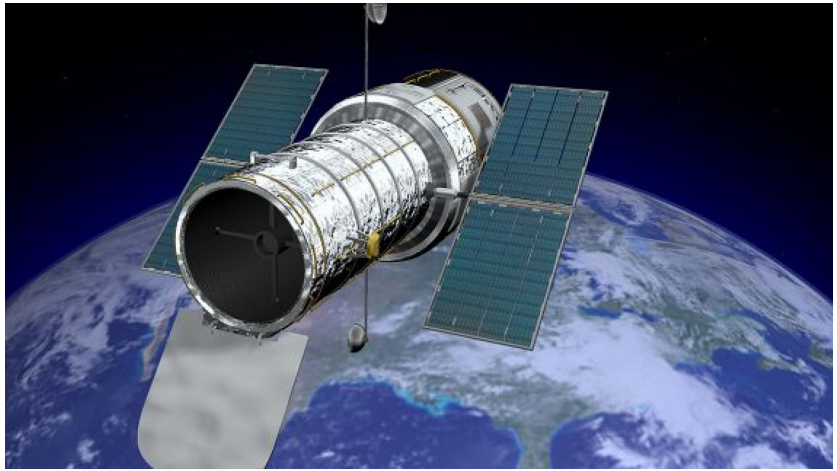
Find:

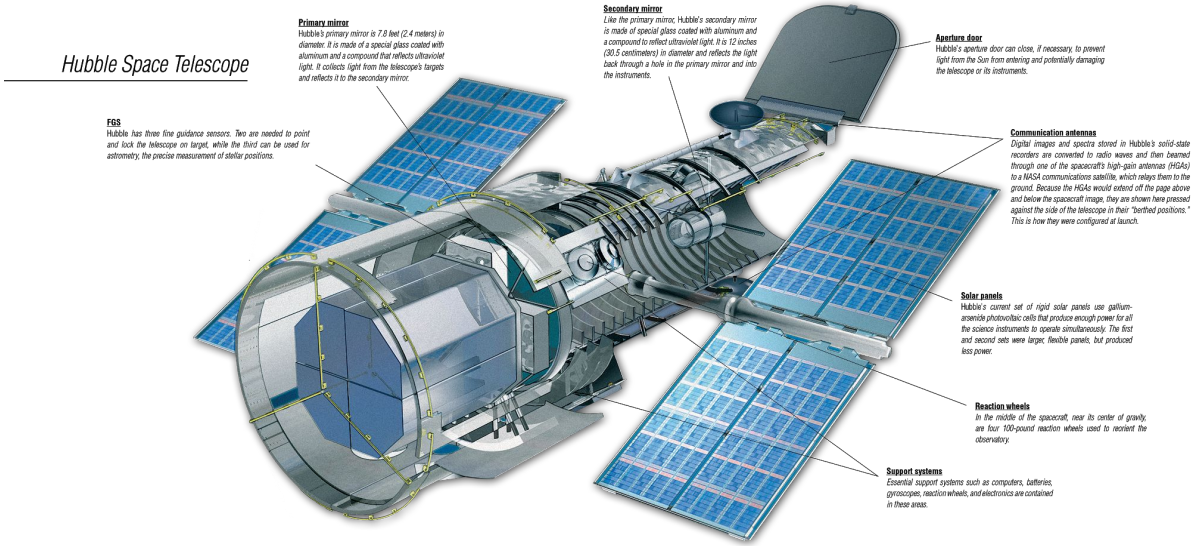
- Sequence of actions that achieves the goal (e.g, [Left, Down, Left, ...]
- Method capable of finding a solution for every application domain

Real world applications

Hubble telescope

- Planning and scheduling
- Short-term
- Long-term (1-2 years)





Amazon Robotics

Purchased Kiva in 2012

Warehouse robots

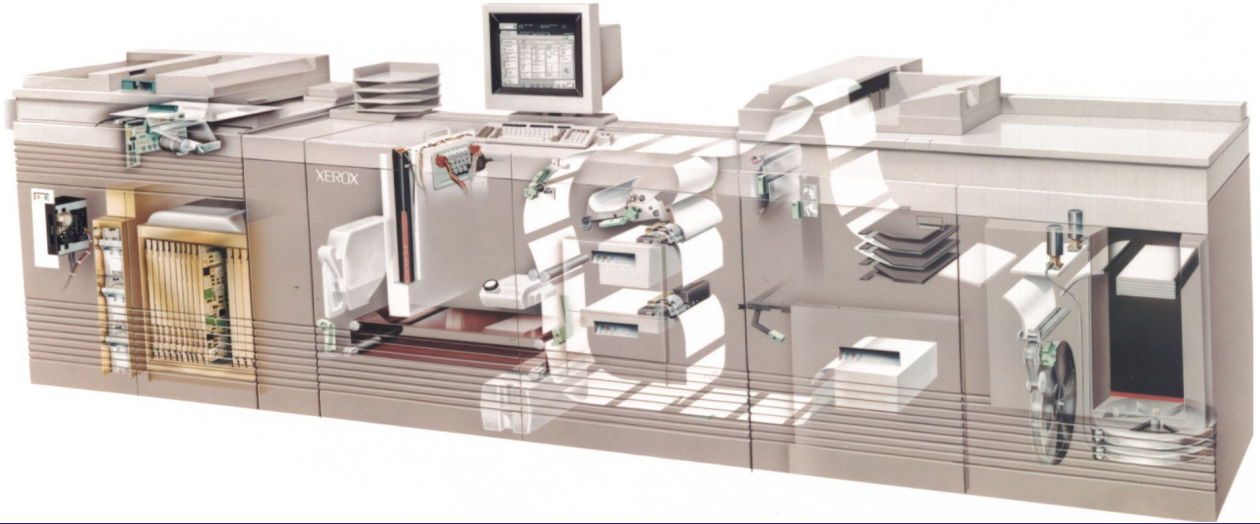
Planning to bring items to associates to pick customers orders

Xerox Printers

Print and bind a whole book in seconds

Model-based planning

Model-based diagnostics



Others

Autonomous vehicles (drones, cars, etc.)

Cognitive robots

Human-agent collaboration

Define the problem - Knowledge representation

The world is dynamic

- What's true now may not be true tomorrow
- Our actions trigger changes in the world

FOL in a dynamic world?

Reasoning in a dynamic world ↔ Planning

- Find a sequence of actions that lead to a goal state (of the world)
- This is just a complex search problem!

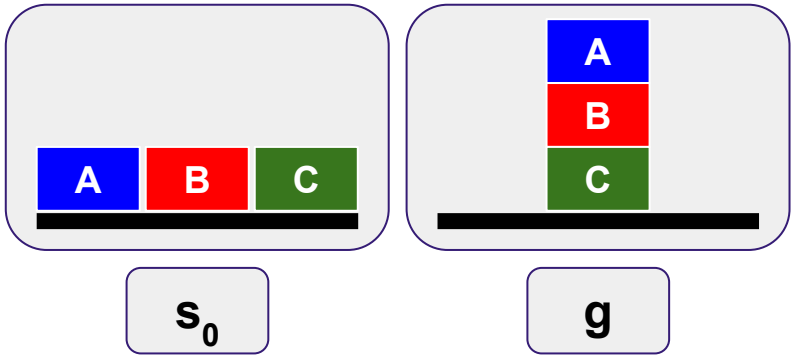
Situation Calculus

Logic for reasoning about changes in the state of the world

Prove: On(A,B) ∧ On(B,C)

The world is described by:

- Sequences of **situations** of the current state
- Changes from one situation to another are caused by **actions**



Fluents:

Vary from one situation to the next

On(A,Table) → On(A,Table,s_0)

Atemporal functions and predicates:

True in any situation

red(B)

Situation Calculus

Actions change a situation. They are described by stating their effects

Possibility Axiom: preconditions ⇒ Poss(a,s)

$\forall x\ \forall y\ \forall z\ \forall s\ On(x, y, s) \land Clear(x, s) \land Clear(z, s) \Rightarrow Poss(Move(x, y, z), s)$

Effect Axiom: Poss(a,s) Changes that result from action

$\forall x\ \forall y\ \forall z\ \forall s\ Poss(Move(x, y, z), s) \Rightarrow On(x, z, \mathbf{Result(Move(x, y, z), s)})$

Frame Problem

Problem: Actions don't specify what happens to objects not involved in the action, but the logic framework requires that information

$$\forall s\ \forall x\ Poss(Move(Table, x), s) \Rightarrow On(x, Table, Result(Move(Table, x), s))$$

Frame Axioms: Inform the system about preserved relations

$$\forall s\ \forall x\ \forall y\ \forall z\ [On(x, y, s) \land (x != z)] \Rightarrow On(x, y, Result(Move(Table, z), s)))$$

STRIPS

Reasoning about action and change

- Define initial state and goal
- Define actions with preconditions and the effects of the actions

Situation calculus [McCarthy, Hayes 1969] [Reiter 2001] is based on first-order logic allowing rich representations

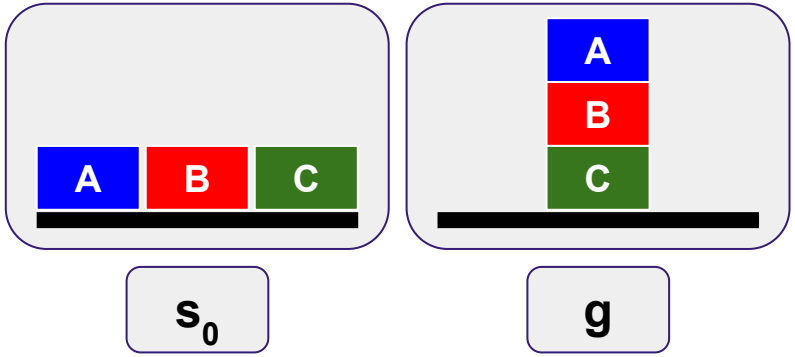
STRIPS planning [Fikes & N. J. Nilsson, 1971], more expressive planning formalisms

Basis for PDDL, language still used today

Blocks World

Initial state: s_0

Goal: **g**



Available actions: moving a block

- From the table to the top of another block
- From the top of another block to the table
- From the top of one block to the top of another

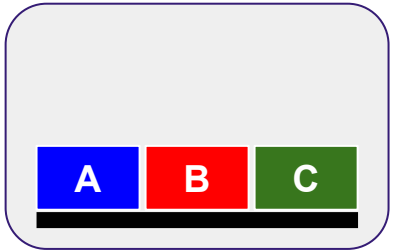
Blocks World

Initial state

Representation of the properties of the domain using first-order logic literals

Two relations:

- On(b,x):
  block b is on top of x, where x is another block or the table
- Clear(x):
  a block can be placed on top of x



s_0

On(A,Table)
On(B,Table)
On(C,Table)
Clear(A)
Clear(B)
Clear(C)

Blocks World

Initial state

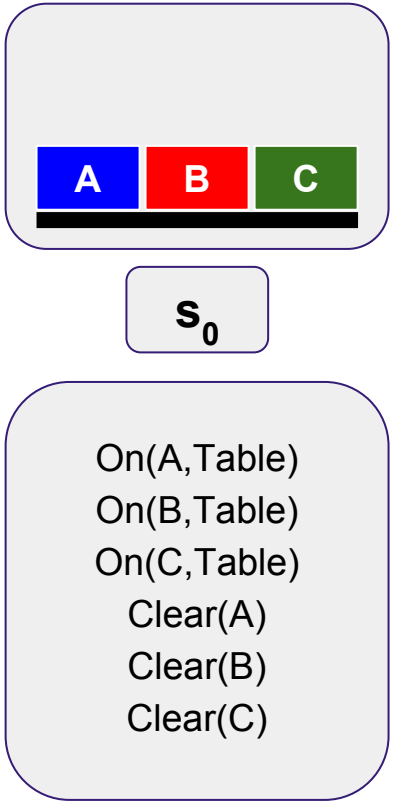
Representation of the properties of the domain using first-order logic literals

Ground and function-free

Completely specified

based on a **closed-world assumption**

^ whatever you don't know to be true, you assume false



Blocks World

Initial state

Closed-world assumption

Any literal not mentioned in the description of the state are assumed to be false!

¬On(A,A)
¬On(A,B)
¬On(A,C)
¬On(B,A)
¬On(B,B)
¬On(B,C)
¬On(C,A)
¬On(C,B)
¬On(C,C)
¬On(A,A)
¬On(A,B)
¬On(A,C)
¬On(Table,A)
¬On(Table,B)
¬On(Table,C)
¬On(Table,Table)
¬Clear(Table)

s_0

On(A,Table)
On(B,Table)
On(C,Table)
Clear(A)
Clear(B)
Clear(C)

Blocks World

Goal

Representation of the properties of the domain using first-order logic literals

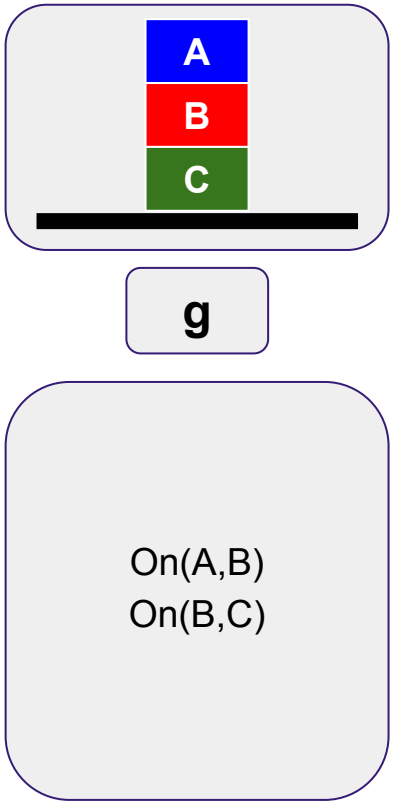
Ground and function-free

Partially specified

No **closed-world assumption**

A state **s** satisfies goal **g** if it contains all literals in **g**

(more literals may be in **s**)

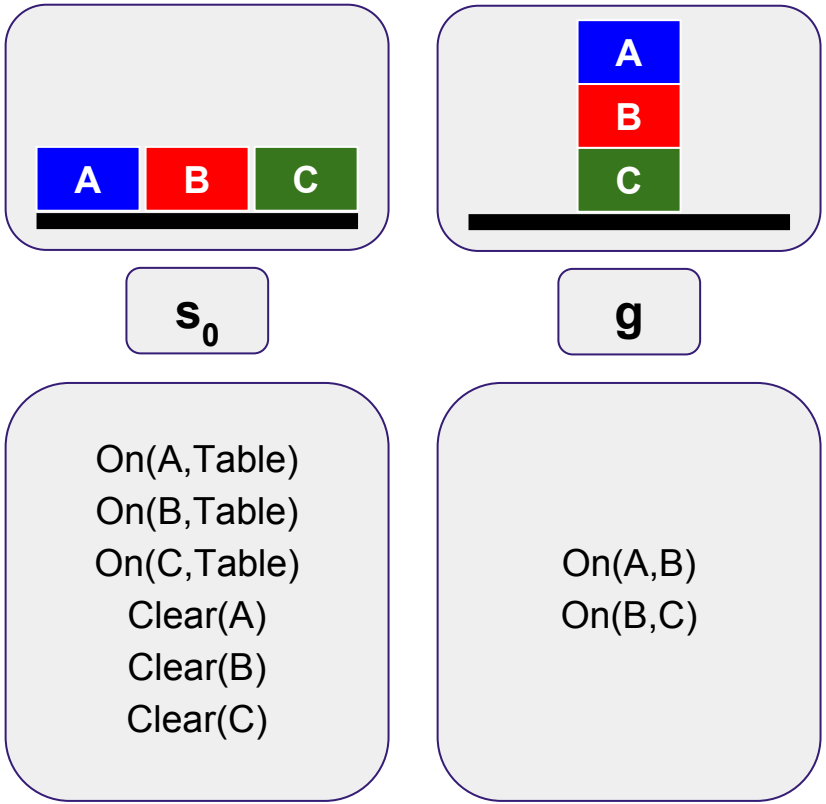


Blocks World

Initial state and goal condition are described using first-order logic literals

- Ground
- Function-free
- Positive

(list of literals forms a logical conjunction

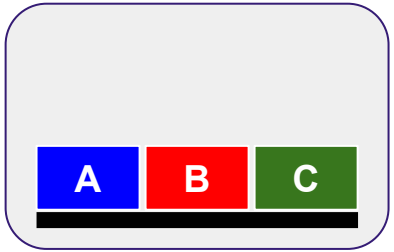


Blocks World

Available actions

Move:

- From the table to the top of another block
- From the top of another block to the table
- From the top of one block to the top of another block



s_0

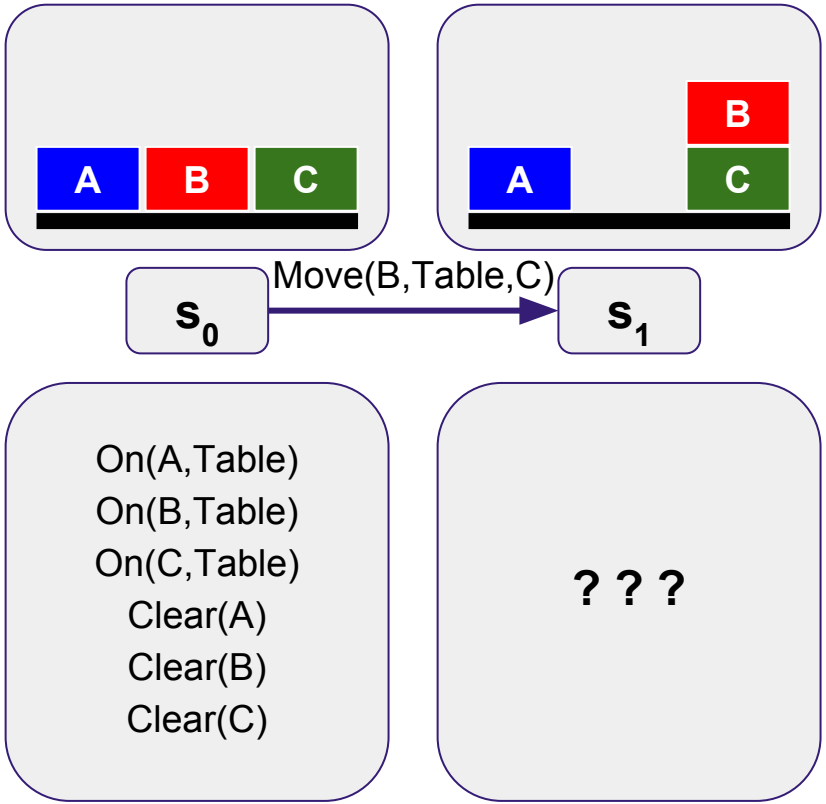
On(A,Table)
On(B,Table)
On(C,Table)
Clear(A)
Clear(B)
Clear(C)

Blocks World

Available actions

Move:

- From the table to the top of another block
- From the top of another block to the table
- From the top of one block to the top of another block

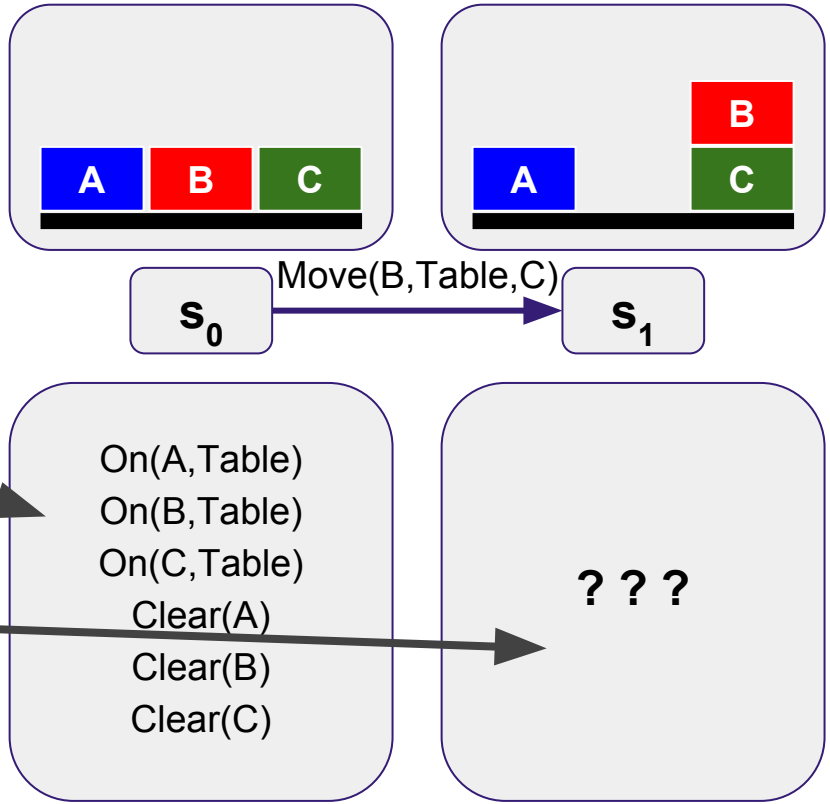


Blocks World

Available actions

`Move(b, x, y)`

- Preconditions
  - Literals denoting what needs to be in the state for the action to be applicable
- Effects
  - Literals denoting how the state is transformed when the action is applied



Blocks World

Available actions

`Move(b, x, y)`

Preconditions

On(b,x)

Clear(b)

Clear(y)

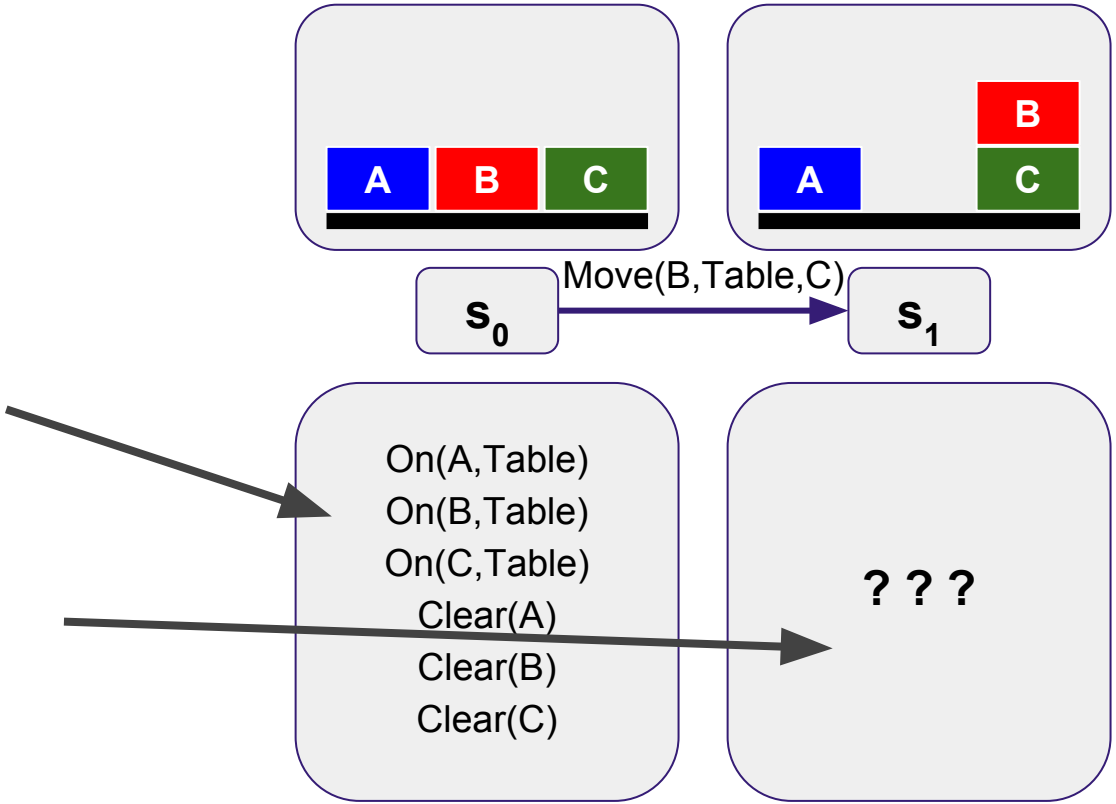
Effects

On(b,y)

Clear(x)

¬On(b,x)

¬Clear(y)

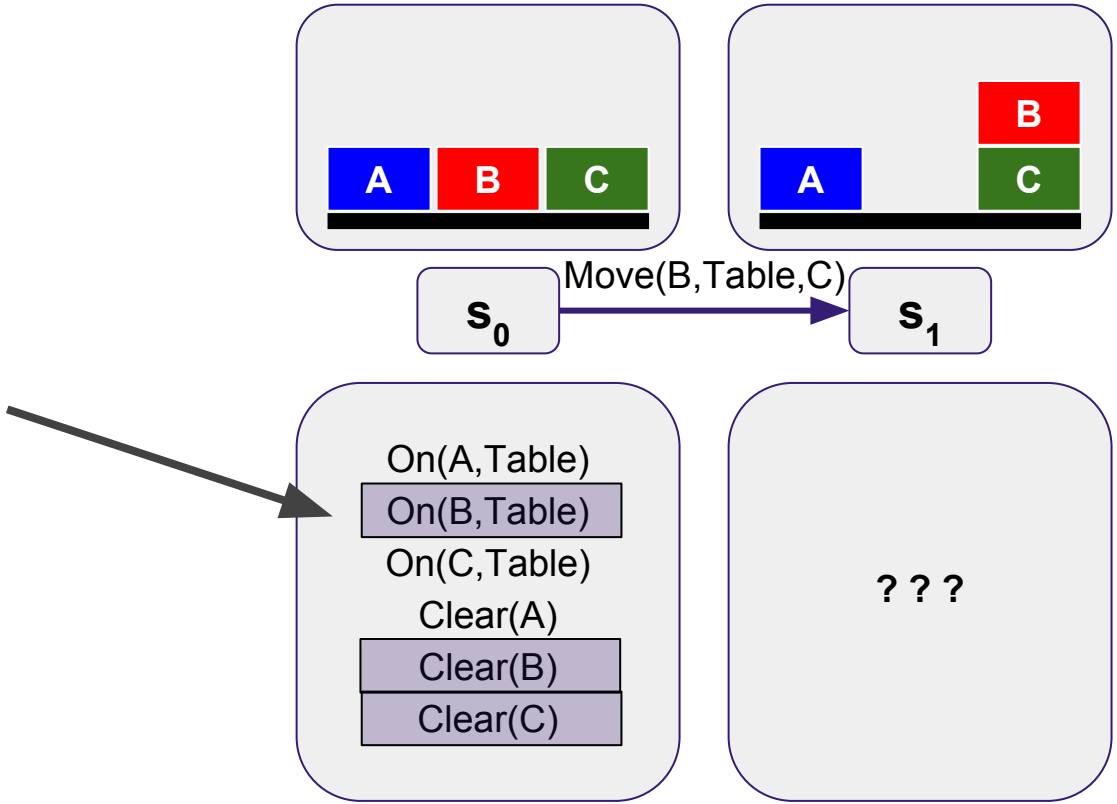


Blocks World

Available actions

`Move(B,Table,C)`

- Preconditions
  - On(B,Table) ✓
  - Clear(B) ✓
  - Clear(C) ✓
- Effects
  - On(b,y)
  - Clear(x)
  - ¬On(b,x)
  - ¬Clear(y)



Blocks World

Available actions

`Move(B,Table,C)`

Preconditions

On(B,Table)

Clear(B)

Clear(C)

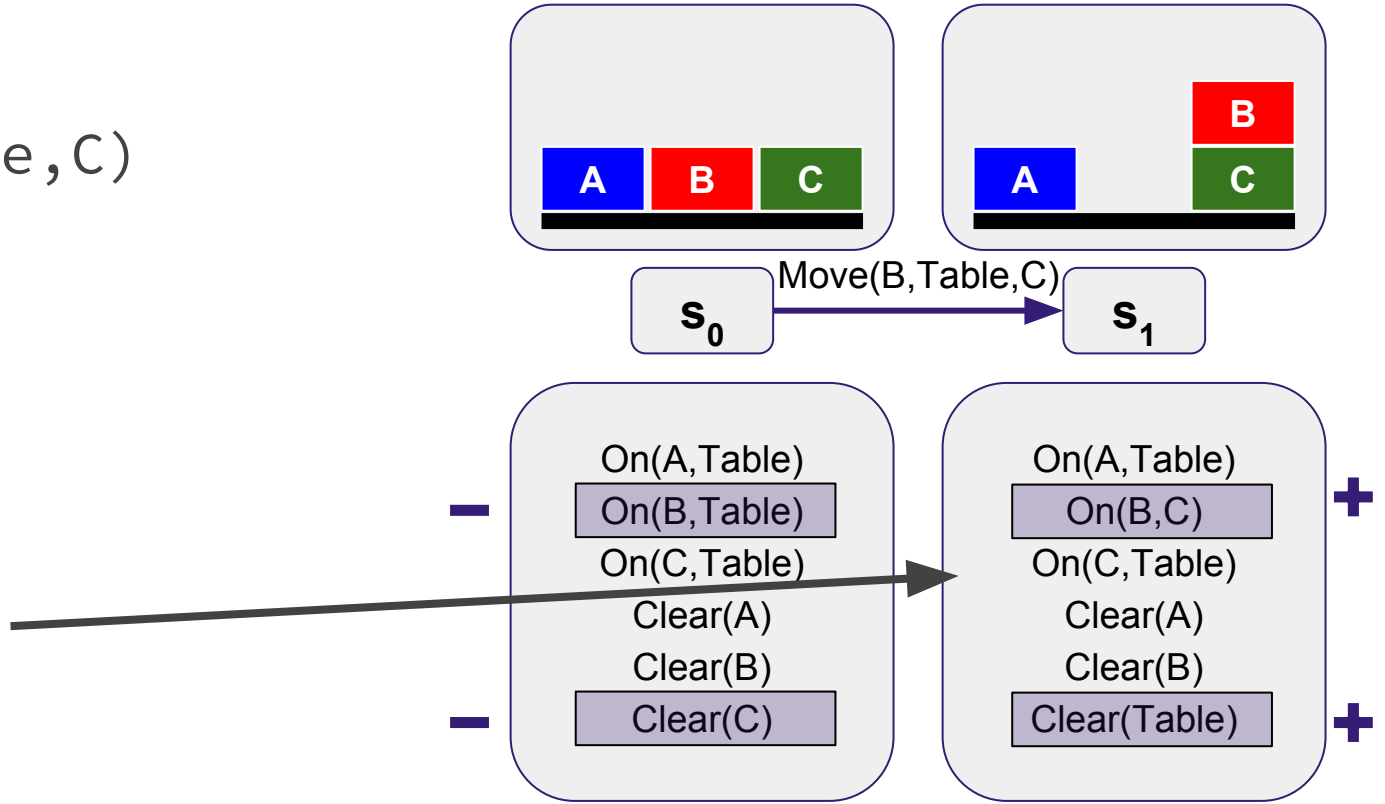
Effects

\+ On(b,y)

\+ Clear(x)

− ¬On(b,x)

− ¬Clear(y)



Blocks World

Available actions

`Move(b, x, y)`

Preconditions

On(b,x)

Clear(b)

Clear(y)

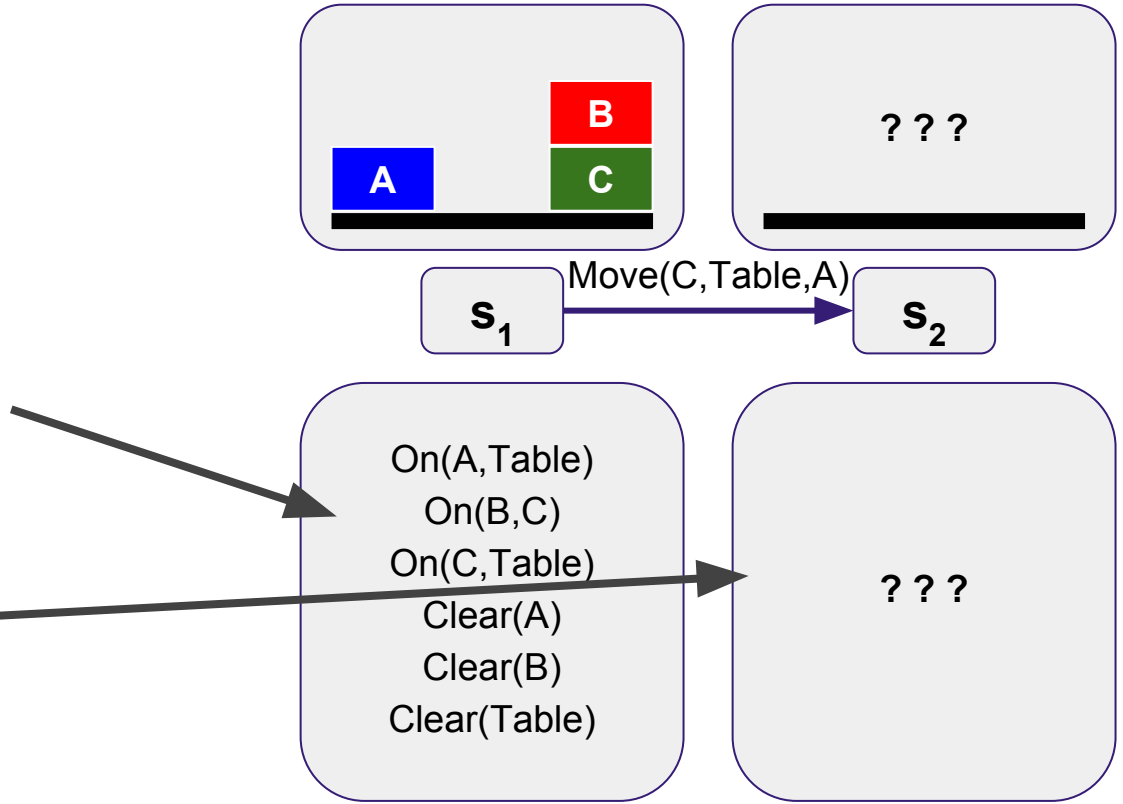
Effects

On(b,y)

Clear(x)

¬On(b,x)

¬Clear(y)

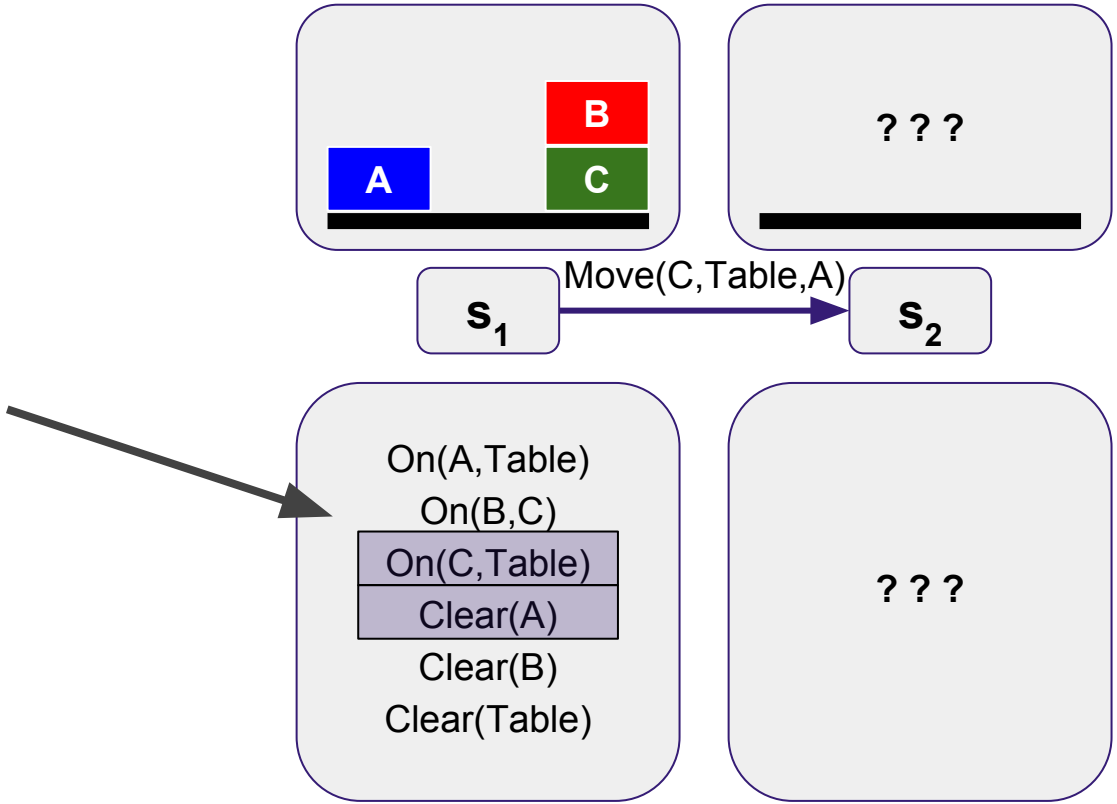


Blocks World

Available actions

`Move(C,Table,A)`

- Preconditions
  - On(C,Table) ✓
  - Clear(C) X
  - Clear(A) ✓
- Effects
  - On(C,A)
  - Clear(Table)
  - ¬On(C,Table)
  - ¬Clear(A)



Blocks World

Available actions

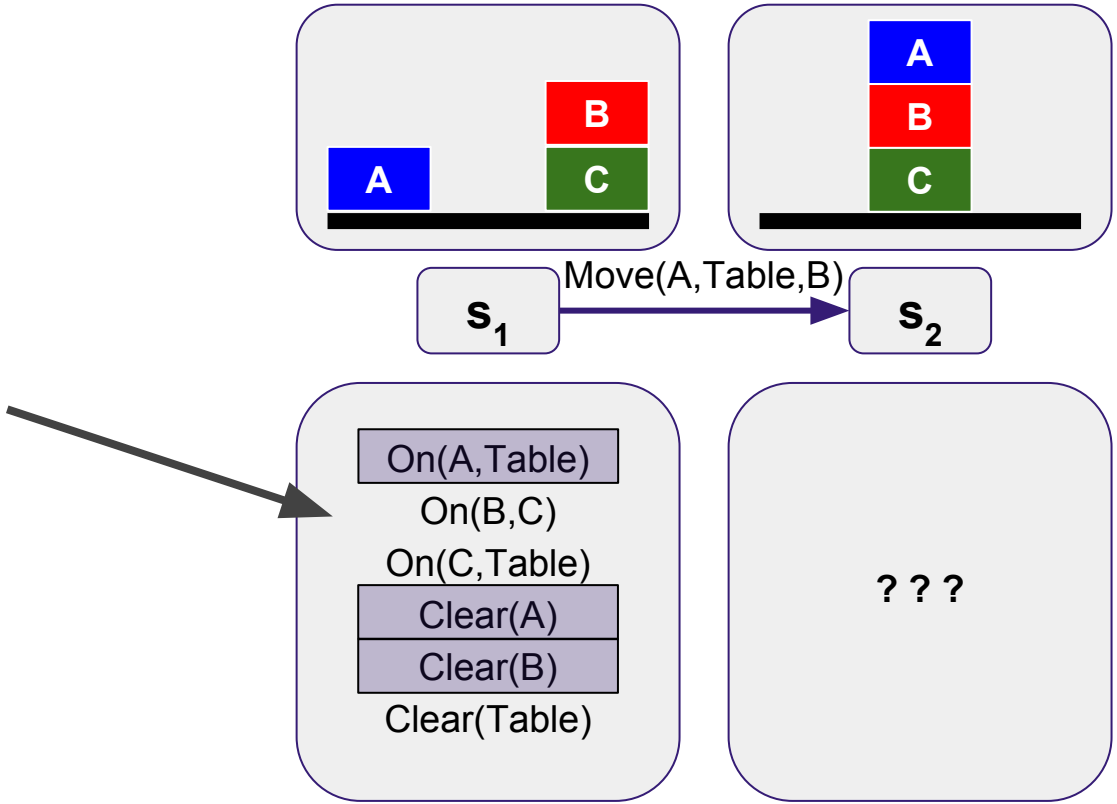
`Move(A,Table,B)`

Preconditions

- On(A,Table) ✓
- Clear(A) ✓
- Clear(B) ✓

Effects

- On(A,B)
- Clear(Table)
- ¬On(A,Table)
- ¬Clear(B)



Blocks World

Available actions

`Move(A,Table,B)`

Preconditions

On(A,Table)

Clear(A)

Clear(B)

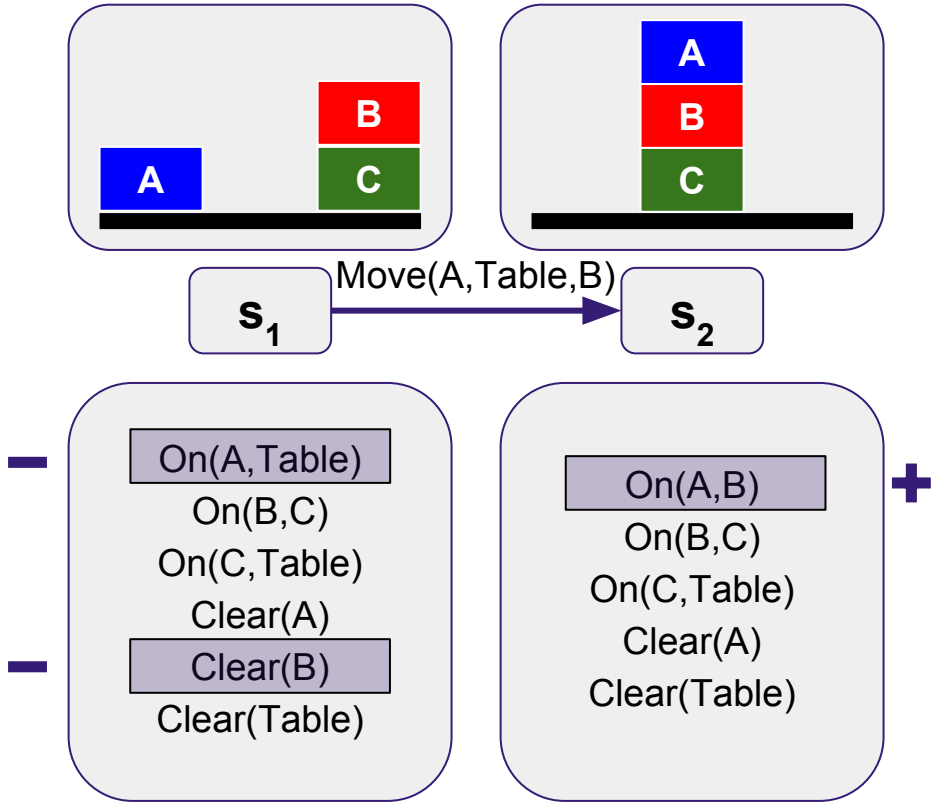
Effects

\+ On(A,B)

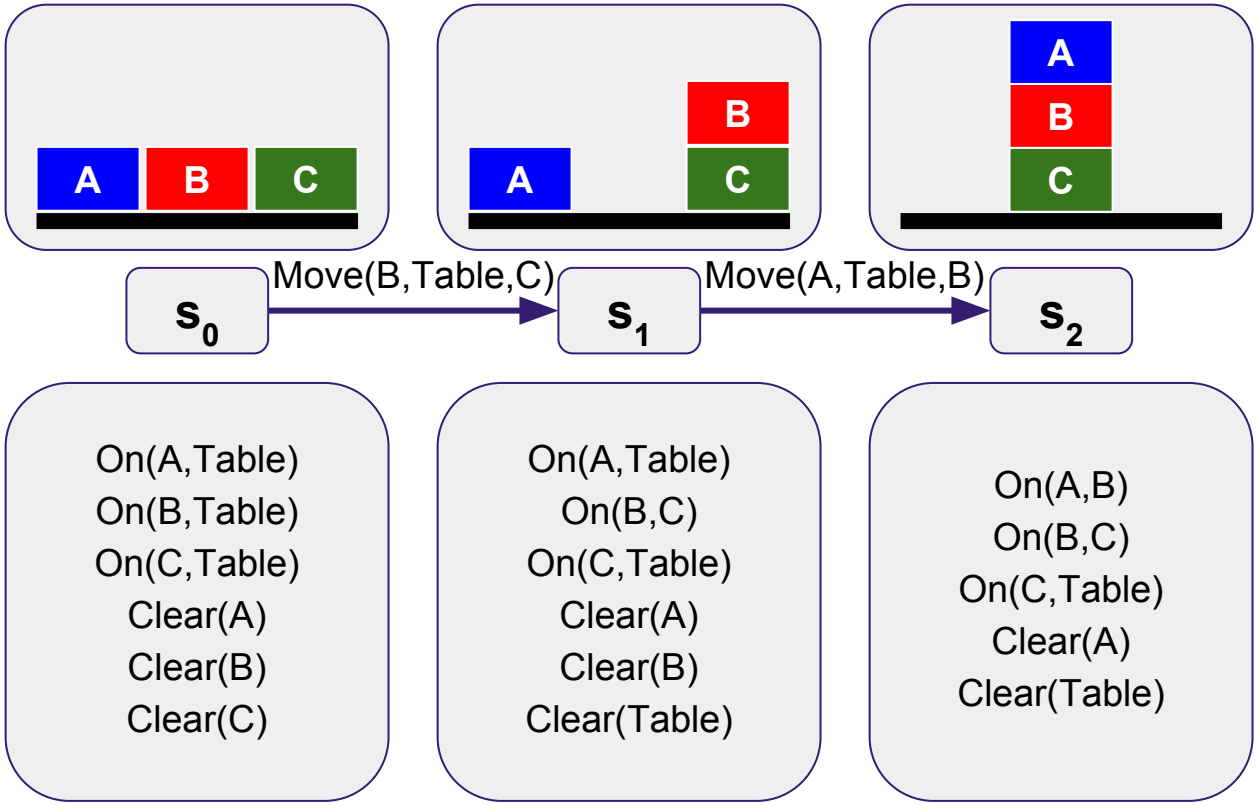
\+ Clear(Table)

− ¬On(A,Table)

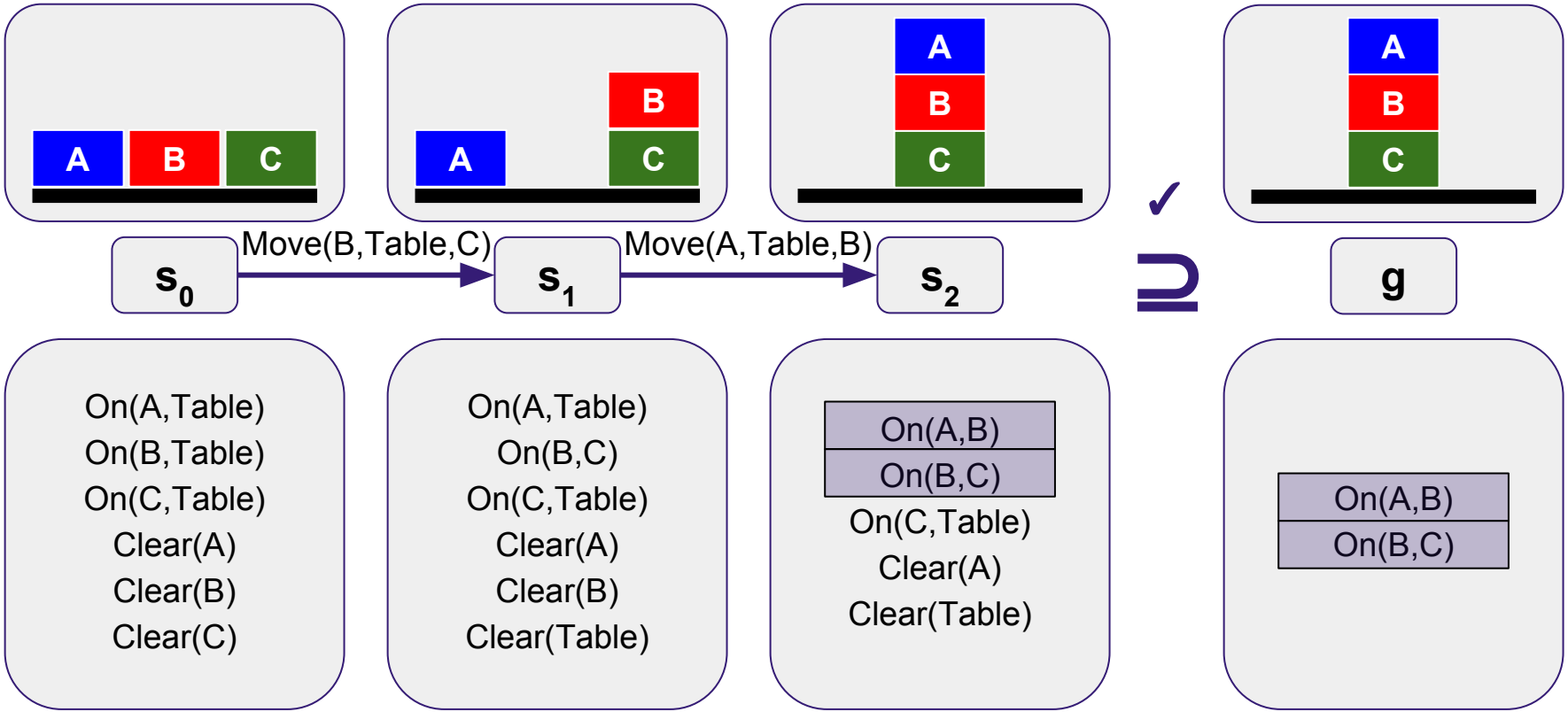
− ¬Clear(B)



Blocks World



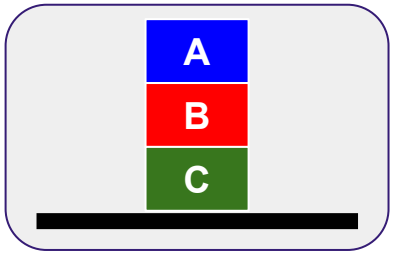
Blocks World



Blocks World

Is our representation of the blocks world correct?

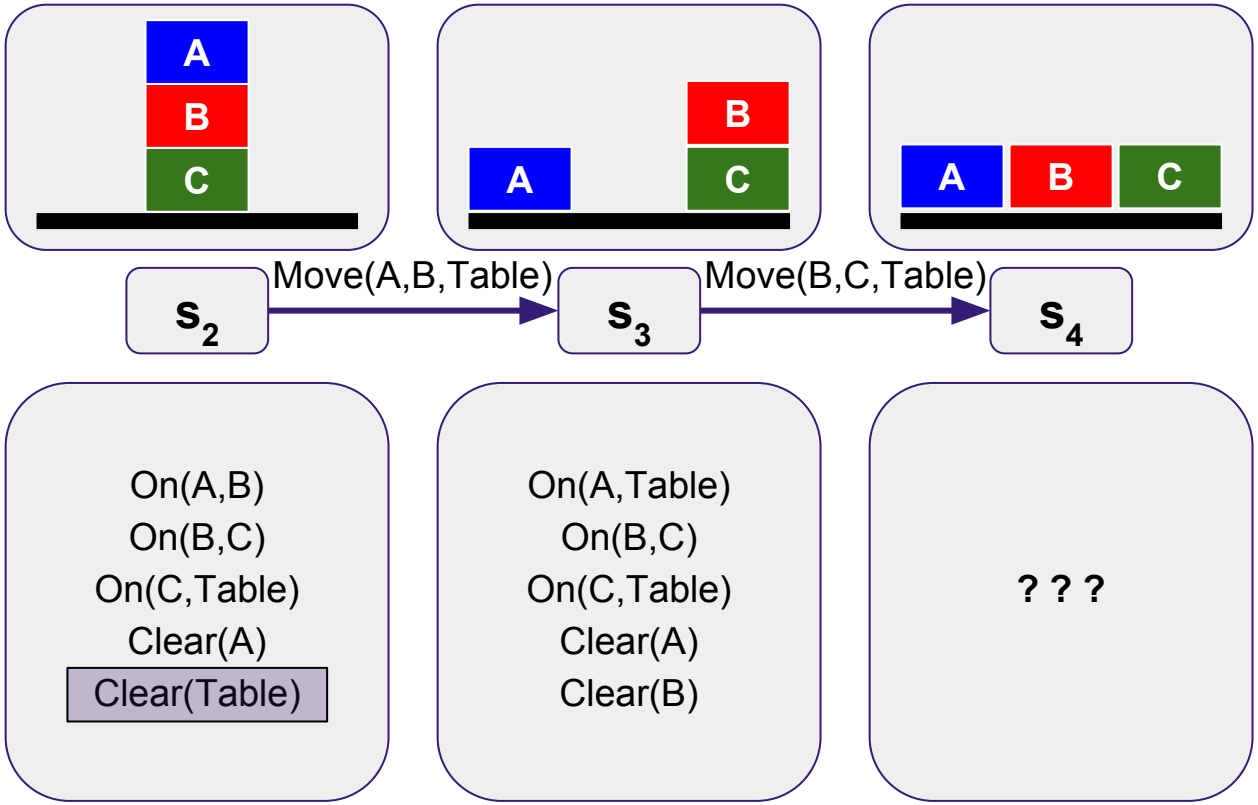
Consider moving back blocks A and B back to the table



s_2

On(A,B)
On(B,C)
On(C,Table)
Clear(A)
Clear(Table)

Blocks World



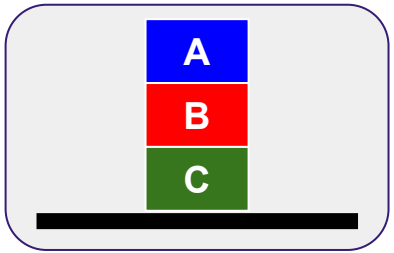
Blocks World

Is our representation of the blocks world correct?

Consider moving back blocks A and B back to the table

Clear(Table) needs to be treated differently

- Move(b,x,y)
- MoveToTable(b,x)



s_2

On(A,B)
On(B,C)
On(C,Table)
Clear(A)
Clear(Table)

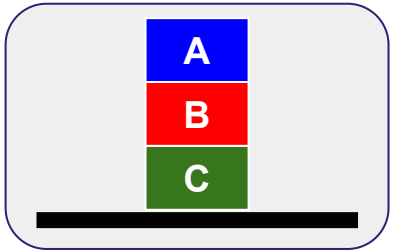
STRIPS planning

Init(On(A,Table) ∧ On(B,Table) ∧ On(C,Table) ∧ Clear(A) ∧
Clear(B) ∧ Clear(C))

Goal(On(A,B) ∧ On(B,C))

Action(Move(b,x,y),
PRECONDITIONS: On(b,x) ∧ Clear(b) ∧ Clear(y)
EFFECTS: On(b,y) ∧ Clear(x) ∧ ¬On(b,x) ∧ ¬Clear(y))

Action(MoveToTable(b,x),
PRECONDITIONS: On(b,x) ∧ Clear(b)
EFFECTS: On(b,Table) ∧ Clear(x) ∧ ¬On(b,x))



S_2

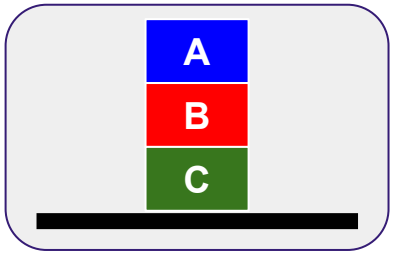
On(A,B)
On(B,C)
On(C,Table)
Clear(A)
Clear(Table)

STRIPS planning

Variables that appear in preconditions and effects need to be parameters of the action schema

Action(Move(b,x,y),
PRECONDITIONS: On(b,x) ∧ Clear(b) ∧ Clear(y)
EFFECTS: On(b,y) ∧ Clear(x) ∧ ¬On(b,x) ∧ ¬Clear(y))

Action(MoveToTable(b,x),
PRECONDITIONS: On(b,x) ∧ Clear(b)
EFFECTS: On(b,Table) ∧ Clear(x) ∧ ¬On(b,x))



S_2

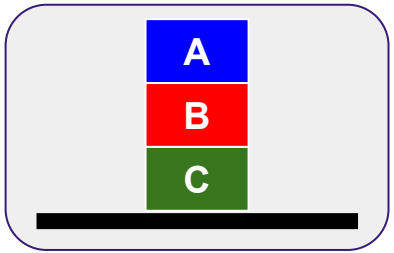
On(A,B)
On(B,C)
On(C,Table)
Clear(A)
Clear(Table)

STRIPS planning

Variables that appear in preconditions and effects need to be parameters of the action schema

Action(Move(b,x,y),
PRECONDITIONS: On(b,x) ∧ Clear(b) ∧ Clear(y)
EFFECTS: On(b,y) ∧ Clear(x) ∧ ¬On(b,x) ∧ ¬Clear(y))

Action(MoveToTable(b,x),
PRECONDITIONS: On(b,x) ∧ Clear(b)
EFFECTS: On(b,Table) ∧ Clear(x) ∧ ¬On(b,x))



s_2

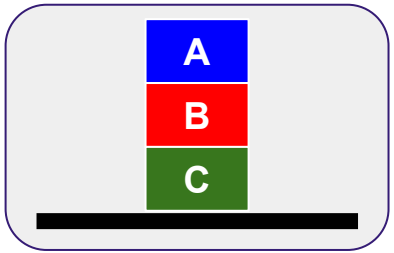
On(A,B)
On(B,C)
On(C,Table)
Clear(A)
Clear(Table)

STRIPS planning

Variables that appear in preconditions and effects need to be parameters of the action schema

Action(Move(b,x,y),
PRECONDITIONS: On(b,x) ∧ Clear(b) ∧ Clear(y)
EFFECTS: On(b,y) ∧ Clear(x) ∧ ¬On(b,x) ∧ ¬Clear(y))

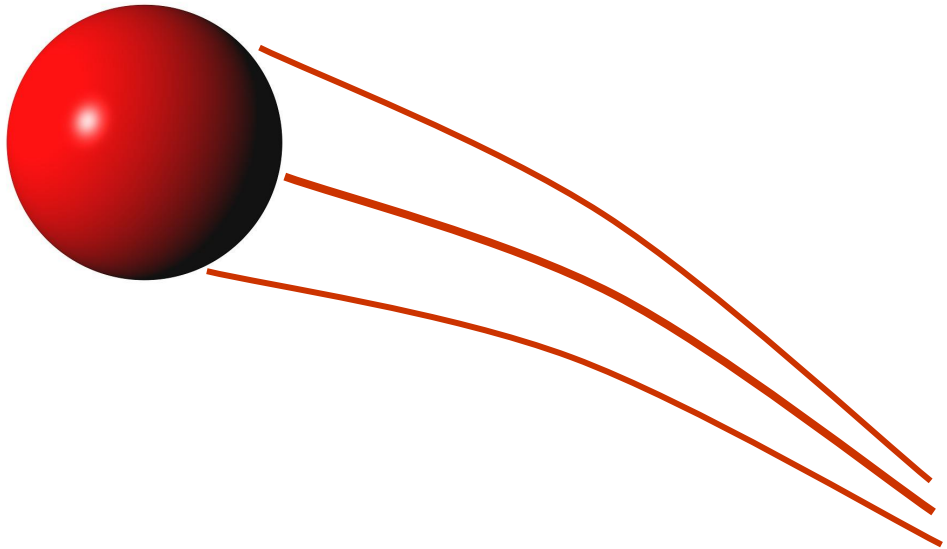
Action(MoveToTable(b,x),
PRECONDITIONS: On(b,x) ∧ Clear(b)
EFFECTS: On(b,Table) ∧ Clear(x) ∧ ¬On(b,x))



s_2

On(A,B)
On(B,C)
On(C,Table)
Clear(A)
Clear(Table)

Pass the ball - STRIPS



Init(HasBall(Ethan) ∧
LastLetter(Ethan, n))

Goal(HasBall(Willie) ∧
FirstLetter(Willie,w))

Action(PassBall(x,l,y,f)
PRECONDITIONS: LastLetter(x,l) ∧ FirstLetter(y,f) ∧
Same(l,f) ∧ HasBall(x)
EFFECTS: HasBall(y) ∧ ¬HasBall(x)

Next time

Problem

Domain

Description

Language